

# Date Science with R

R, RStudio & GitHub

Peng Zhang

School of Mathematical Sciences, Zhejiang University

2025/06/25

# R

## Objectives

In this lesson we will:

- get oriented to the RStudio interface
- work with R in the console
- be introduced to built-in R functions
- learn to use the help pages
- explore RMarkdown

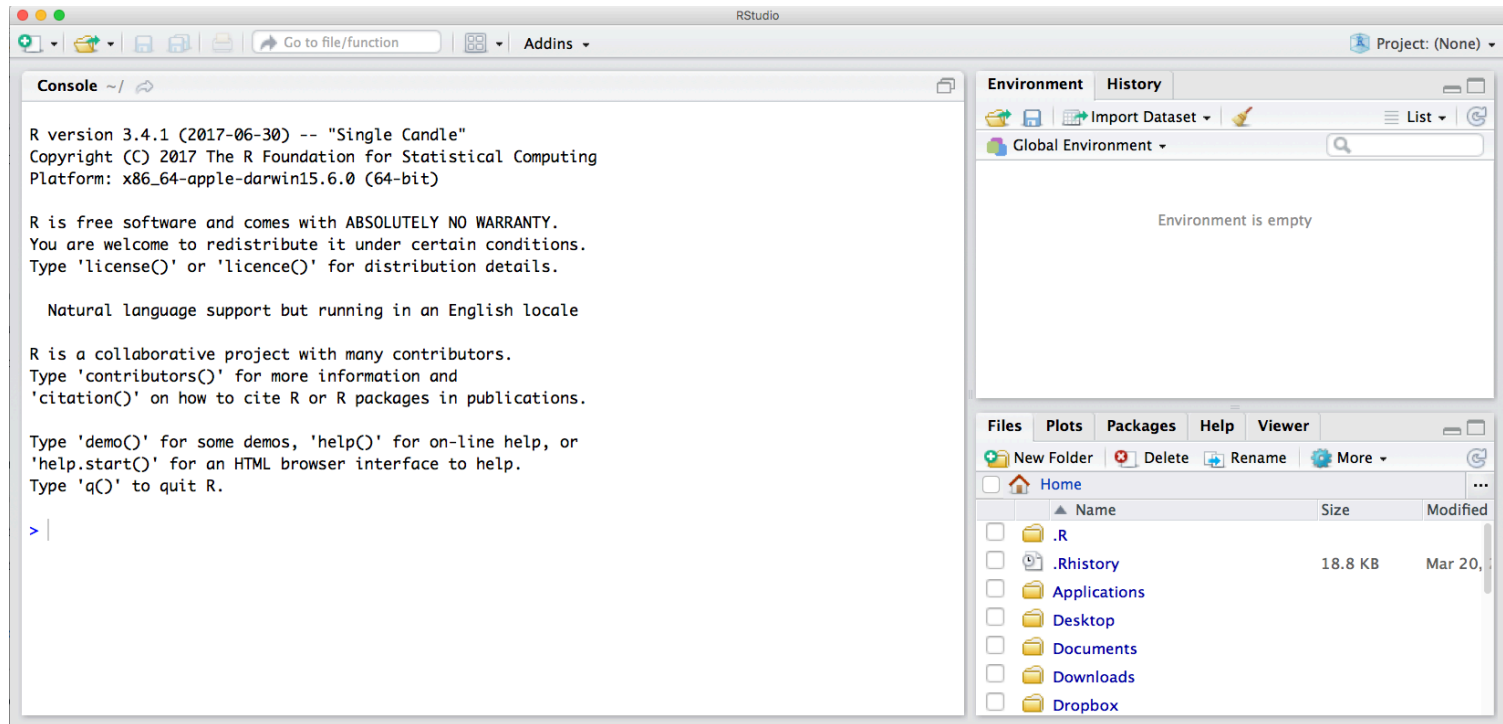
# Preparation

1. Download and install up-to-date versions of:
  - R: <https://cloud.r-project.org>
  - RStudio: <http://www.rstudio.com/download>
  - Git: <https://git-scm.com/downloads> *Note: open the download and follow normal install procedures on your computer but you won't see any software installed when you're done*
2. Create a GitHub account: <https://github.com> *Note! Shorter names that kind of identify you are better, and use your work email!*

## Why learn R with RStudio?

- Think of yourself as a pilot, and R is your airplane.
- And if R were an airplane, RStudio is the airport.

# R at the console



Notice the default panes:

- Console (entire left)
- Environment/History (tabbed in upper right)
- Files/Plots/Packages/Help (tabbed in lower right)

Let's go into the Console, where we interact with the live R process.

Make an assignment and then inspect the object you created by typing its name on its own.

```
x <- 3 * 4  
x
```

```
## [1] 12
```

All R statements where you create objects – “assignments” – have this form: *objectName <- value*.

Object names cannot start with a digit and cannot contain certain other characters such as a comma or a space. You will be wise to adopt a convention for demarcating words in names.

```
# i_use_snake_case  
# other.people.use.periods  
# evenOthersUseCamelCase
```

# Operators

- **Unary** – for arithmetic negation, ! for Boolean
- **Binary** usual arithmetic operators, plus ones for modulo and integer division; take two numbers and give a number

```
7+5
```

```
## [1] 12
```

```
7/5
```

```
## [1] 1.4
```

```
7 %% 5
```

```
## [1] 2
```

```
7 %/% 5
```

```
## [1] 1
```

# Logical operators and expressions

**Comparisons** are also binary operators; they take two objects, like numbers, and give a Boolean

- `==` means 'is equal to'
- `!=` means 'is not equal to'
- `<` means 'is less than'
- `>` means 'is greater than'
- `<=` means 'is less than or equal to'
- `>=` means 'is greater than or equal to'

```
7 > 5
```

```
## [1] TRUE
```

```
7 == 5
```

```
## [1] FALSE
```

```
7 != 5
```

```
## [1] TRUE
```

# Boolean operators

Basically "and" and "or":

```
(5 > 7) & (6*7 == 42)
```

```
## [1] FALSE
```

```
(5 > 7) | (6*7 == 42)
```

```
## [1] TRUE
```

(will see special doubled forms, && and ||, later)



# R functions, help pages

Let's try using `seq()` which makes regular sequences of numbers.

```
seq(1, 10)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

For help on `seq()` function, try

```
?seq  
help(seq) # same as ?seq
```

The help page tells the name of the package in the top left, and broken down into sections:

- **Description:** An extended description of what the function does.
- **Usage:** The arguments of the function and their default values.
- **Arguments:** An explanation of the data each argument is expecting.
- **Details:** Any important details to be aware of.
- **Value:** The data the function returns.
- **See Also:** Any related functions you might find useful.
- **Examples:** Some examples for how to use the function.

```
seq(from = 1, to = 10)
```

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

```
seq(from = 1, to = 10, by = 2)
```

```
## [1] 1 3 5 7 9
```

## Your turn

Exercise: Look up the help file for a function that you know or expect to exist. Here are some ideas: `?getwd()`, `?plot()`, `min()`, `max()`, `?mean()`, `?log()`.

# Packages

One of the amazing things about R is that a vast user community is always creating new functions and packages that expand R's capabilities.

- In R, the fundamental unit of shareable code is the package.
- A package bundles together code, data, documentation, and tests, and is easy to share with others.
- The traditional place to download packages is from CRAN, the Comprehensive R Archive Network, which is where you downloaded R. You can also install packages from GitHub, which we'll do later.
- You don't need to go to CRAN's website to install packages, this can be accomplished within R using the command `install.packages("package-name-in-quotes")`.

```
#install.packages("praise")  
library(praise)  
praise()
```

```
## [1] "You are dandy!"
```

# Clearing the environment

Now look at the objects in your environment (workspace) – in the upper right pane. The workspace is where user-defined objects accumulate.

You can also get a listing of these objects with a few different R commands:

```
objects()
```

```
## [1] "x"
```

```
ls()
```

```
## [1] "x"
```

```
rm(x)
```

To remove everything:

```
rm(list = ls())
```

or click the broom in RStudio's Environment pane.

For reproducibility, it is critical that you delete your objects and restart your R session frequently. Go to the top menus: Session > Restart R.

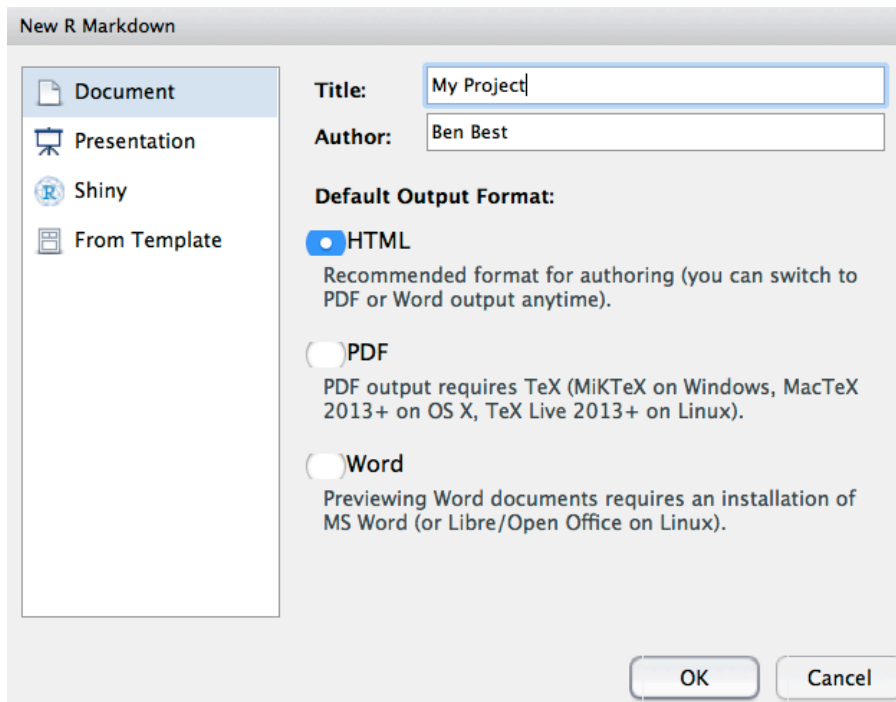
## **Your turn**

**Exercise:** Clear your workspace, then create a few new variables. Create a variable that is the mean of a sequence of 1-20. What's a good name for your variable? Does it matter what your 'by' argument is? Why?

# RMarkdown

An RMarkdown file will allow us to weave markdown text with chunks of R code to be evaluated and output content like tables and plots.

File -> New File -> RMarkdown... -> Document of output format HTML, OK.



This RMarkdown file has 2 different languages within it: **R** and **Markdown**.

There are some good cheatsheets to get you started, and here is one built into RStudio: Go to Help > Markdown Quick Reference

## Your Turn

1. In Markdown write some italic text, make a numbered list, and add a few subheaders. Use the Markdown Quick Reference (in the menu bar: Help > Markdown Quick Reference).
2. Reknit your html file.

## Code chunks

Create a new chunk in your RMarkdown first in one of these ways:

- click “Insert > R” at the top of the editor pane
  - type by hand markdown ````${r}````
  - if you haven’t deleted a chunk that came with the new file, edit that one
- Now, let’s write some R code.

```
x <- 1:15
```



To execute it, we need to get what we typed in the the R chunk (the grey R code) down into the console. How do we do it? There are several ways (let's do each of them):

1. copy-paste this line into the console.
2. select the line (or simply put the cursor there), and click 'Run'. This is available from
  - the bar above the file (green arrow)
  - the menu bar: Code > Run Selected Line(s)
  - keyboard shortcut: command-return
3. click the green arrow at the right of the code chunk

## **Your turn**

Add a few more commands to your file from this lecture. Execute them by trying the three ways above. Then, save your R Markdown file.

# GitHub

We will learn about version control using git and GitHub, and we will interface with this through RStudio. Why use version control? To save time when working with your most important collaborator: you.

git will track and version your files, GitHub stores this online and enables you to collaborate with others (and yourself).

We will interface with GitHub from our local computers using RStudio.

Here's what we'll do after we set up git on your computers:

1. create a repository on Github.com
2. clone locally using RStudio
3. learn the RStudio-GitHub workflow by syncing to Github.com: pull, stage, commit, push
4. explore github.com: files, commit history, file history
5. practice the RStudio-GitHub workflow by editing and adding files
6. practice R Markdown

# What are Git and Github?

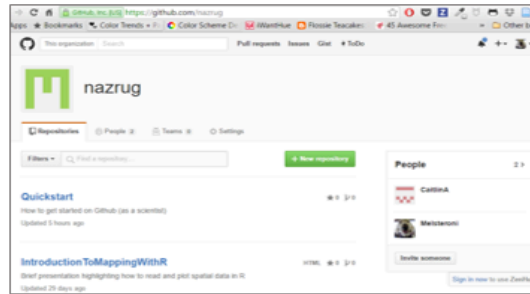
- **Git** is a version control system that lets you track changes to files over time. These files can be any kind of file (eg .doc, .pdf, .xls), but free text differences are most easily visible (eg txt, csv, md).
- **Github** is a website for storing your git versioned files remotely.

# Some Github terminology

- **User:** A Github account for you (e.g., jules32). **Organization:** The Github account for one or more user (e.g., datacarpentry).
- **Repository:** A folder within the organization that includes files dedicated to a project.
- **Local Github:** Copies of Github files located your computer.
- **Remote Github:** Github files located on the <https://github.com> website.
- **Clone:** Process of making a local copy of a remote Github repository. This only needs to be done once (unless you mess up your local copy).
- **Pull:** Copy changes on the remote Github repository to your local Github repository. This is useful if multiple people are making changes to a repository.
- **Push:** Save local changes to remote Github

# REMOTE

(aka Github website)



**Clone** (i.e., copy)  
repository to your  
computer (a one  
time event)

**Pull** remote  
changes

**Push** local  
changes



**LOCAL**  
(aka your computer)

# Setup Git & GitHub

We're going to switch gears from R for a moment and set up Git and GitHub. This set up is a one-time thing!

1. Create **Github** account at <http://github.com>, if you don't already have one. For username, I recommend all lower-case letters, short as you can. I recommend using your *zju.edu.cn* email.
2. You will use the `usethis` package to configure **git** with global commands, which means it will apply 'globally' to all files on your computer, rather than to a specific folder.

```
install.packages("usethis")  
library(usethis)  
use_git_config(user.name = "pengyan2000", user.email = "pengz@zju.edu.cn")
```

*BACKUP PLAN* If `usethis` fails, the following is the classic approach to configuring git. Open the Git Bash program (Windows) or the Terminal (Mac) and type the following:

```
# display your version of git
git --version
```

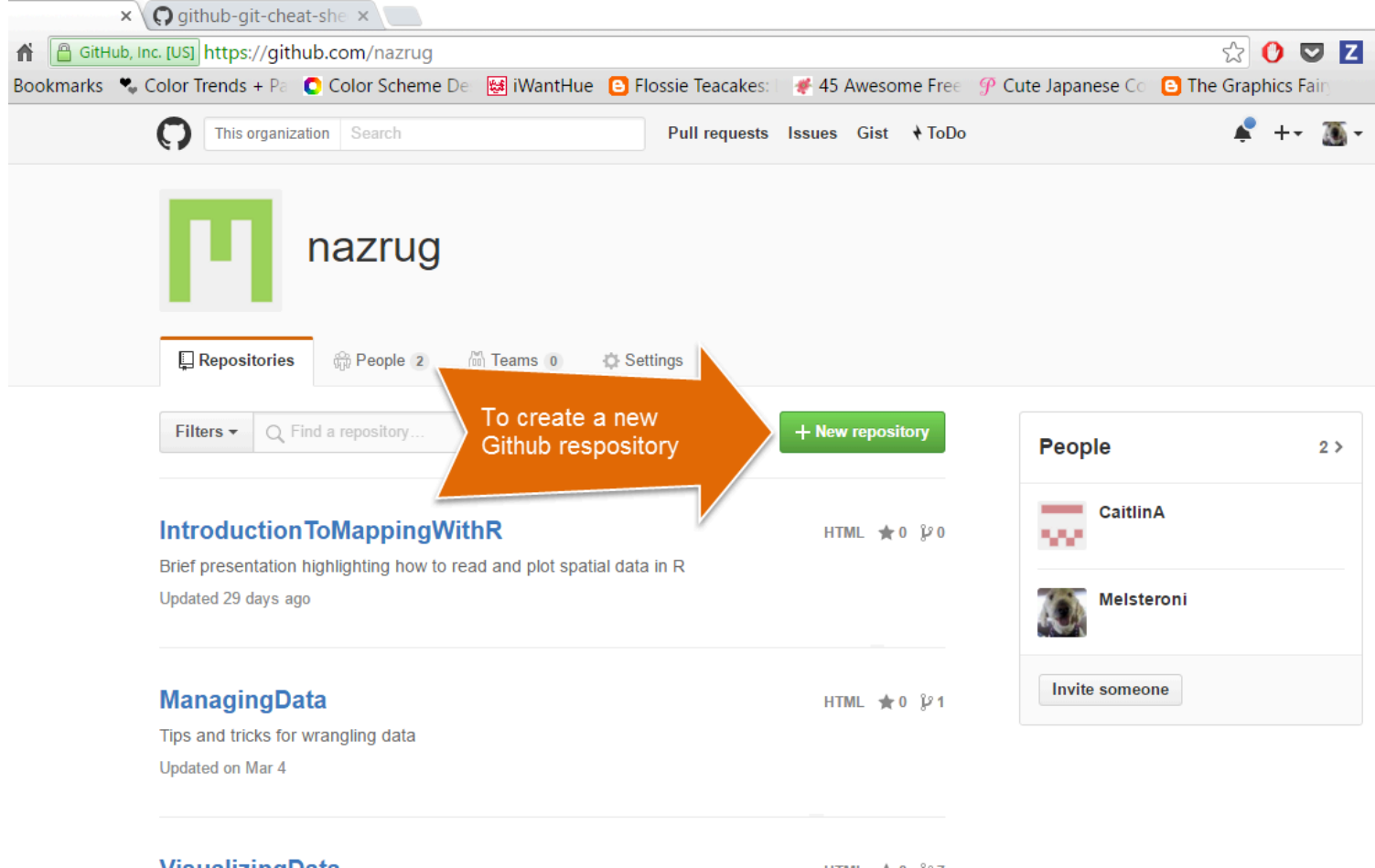
```
# replace USER with your Github user account
git config --global user.name USER
```

```
# replace NAME@EMAIL.EDU with the email you used to register with Github
git config --global user.email NAME@EMAIL.EDU
```

```
# list your config to confirm user.* variables set
git config --list
```

# Create a repository on Github.com

First, go to your account on github.com and click “New repository”.




The screenshot shows the GitHub profile page for the user 'nazrug'. The browser address bar displays 'https://github.com/nazrug'. The page header includes the GitHub logo, a search bar, and navigation links for 'Pull requests', 'Issues', 'Gist', and 'To Do'. The user's profile section shows a green square avatar and the name 'nazrug'. Below the profile, there are tabs for 'Repositories', 'People' (2), 'Teams' (0), and 'Settings'. The 'Repositories' tab is active, showing a list of repositories. An orange arrow points from the text 'To create a new Github repository' to the '+ New repository' button. The repository list includes 'IntroductionToMappingWithR' and 'ManagingData'. The right sidebar shows a list of people followed by the user, including 'CaitlinA' and 'Melsteroni', with an 'Invite someone' button below.

github-github-cheat-sheet x

GitHub, Inc. [US] https://github.com/nazrug

Bookmarks Color Trends + Pa Color Scheme De iWantHue Flossie Teacakes: 45 Awesome Free Cute Japanese Co The Graphics Fair

This organization Search Pull requests Issues Gist To Do

 nazrug

Repositories People 2 Teams 0 Settings

Filters Find a repository...

To create a new Github repository

+ New repository

**IntroductionToMappingWithR** HTML ★ 0 0


Brief presentation highlighting how to read and plot spatial data in R  
Updated 29 days ago


**ManagingData** HTML ★ 0 1

Tips and tricks for wrangling data  
Updated on Mar 4

**VisualizingData**

**People** 2 >

 CaitlinA

 Melsteroni

Invite someone



Choose a name. Call it whatever you want (the shorter the better), or follow me for convenience. I will call mine my-repo.

Also, add a description, make it public, create a README file, and create your repo!

The screenshot shows the GitHub 'Create a new repository' page. On the left, six blue arrows with white text point to specific parts of the form:

- 1. A simple name for the repository (points to the Repository name field)
- 2. A description (points to the Description field)
- 3. Public or private (private available with paid or education) (points to the visibility radio buttons)
- 4. Select this (points to the 'Initialize this repository with a README' checkbox)
- 5. I ignore these (points to the '.gitignore' dropdown)
- 6. Click here! (points to the 'Create repository' button)

The form fields and options are as follows:

- Create a new repository**  
A repository contains all the files for your project, including the revision history.
- Owner:** nazrug
- Repository name:** Quickstart ✓
- Description (optional):** How to get started on Github (as a scientist)
- Visibility:**
  - ☒ **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** (button)

The *Add gitignore* option adds a document where you can identify files or file-types you want Github to ignore. These files will stay in on the local Github folder (the one on your computer), but will not be uploaded onto the web version of Github.

The *Add a license* option adds a license that describes how other people can use your Github files (e.g., open source, but no one can profit from them, etc.). We won't worry about this today.

Check out our new repository!

Notice how the README.md file we created is automatically displayed at the bottom. The .md means that it is Markdown (remember how .Rmd was RMarkdown?) so the formatting we learned in the last lesson apply here.

The screenshot shows a web browser displaying the GitHub repository page for 'nazrug/Quickstart'. The browser's address bar shows the URL 'https://github.com/nazrug/Quickstart'. The repository page includes a search bar, navigation links for 'Pull requests', 'Issues', 'Gist', and 'ToDo', and a header for the repository 'nazrug / Quickstart'. Below the header, there are buttons for 'Unwatch', 'Star', and 'Fork'. The main content area shows the repository's structure with links for 'Code', 'Issues', 'Pull requests', 'Boards', 'Burndown', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected, showing the 'README.md' file. The file's commit history is displayed, showing an initial commit by 'Melsteroni' 2 hours ago. The commit message is 'Initial commit'.

# Clone your repository using RStudio

We'll start of by cloning to our local computer using RStudio. We are going to be cloning a copy of our Remote repository on Github.com to our local computers. Unlike downloading, cloning keeps all the version control and user information bundled with the files.

## **Step 0:** Create your `github` folder

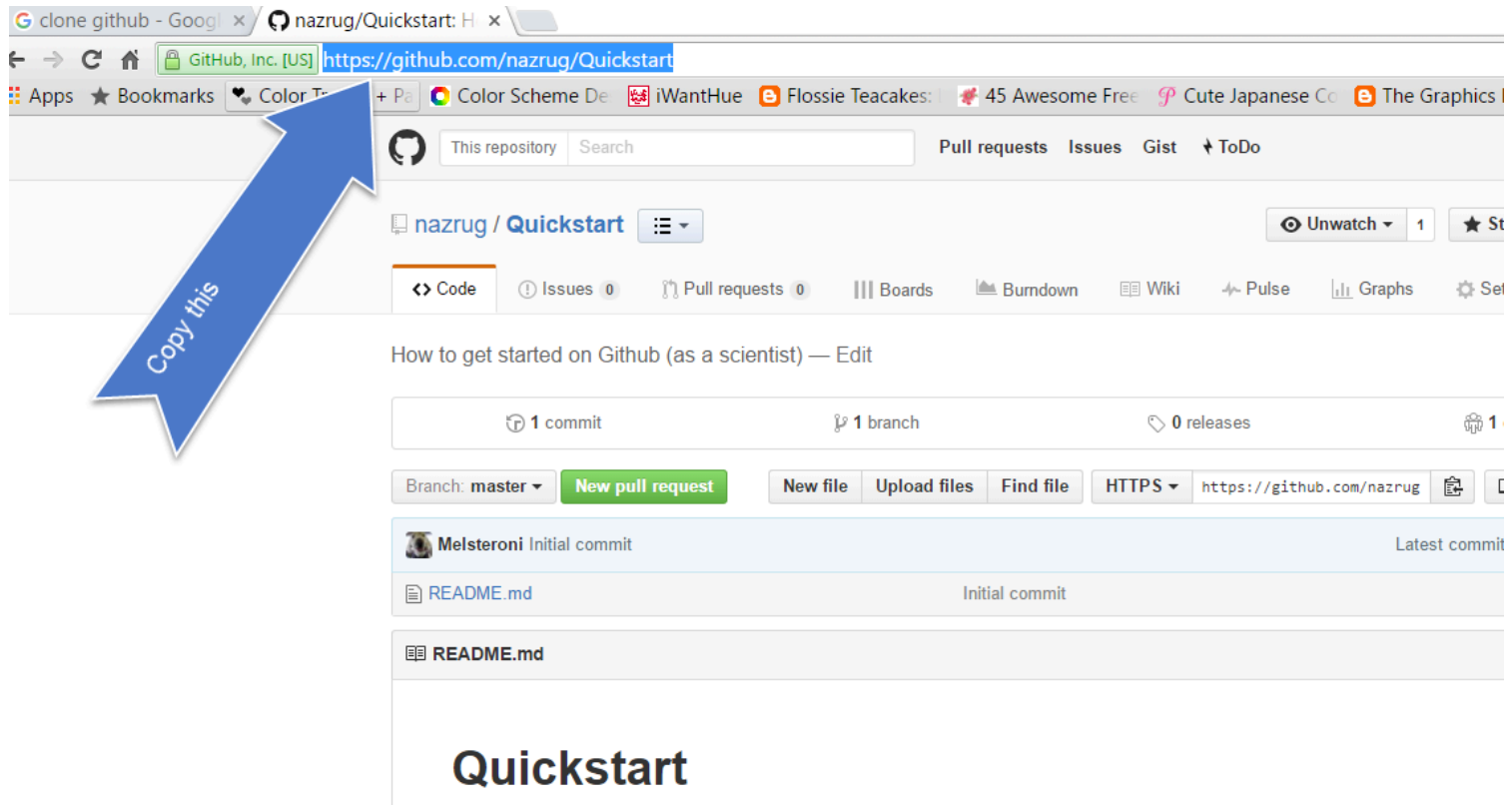
This is really important! We need to be organized and deliberate about where we want to keep all of our GitHub repositories (since this is the first of many in your career).

Let's all make a folder called `github` (all lowercase!) in our home directories. So it will look like this:

- Windows: `Users\[User]\Documents\github\`
- Mac: `Users/[User]/github/`

This will let us take advantage of something that is really key about GitHub.com: you can easily navigate through folders within repositories and the urls reflect this navigation. The greatness of this will be evident soon. So let's set ourselves up for easily translating (and remembering) those navigation paths by having a folder called `github` that will serve as our 'github.com'.

**Step 1:** Copy the web address of the repository you want to clone.



The screenshot shows a web browser window with the address bar displaying `https://github.com/nazrug/Quickstart`. A blue arrow with the text "Copy this" points to the URL. The repository page for "nazrug / Quickstart" is visible, showing the "Code" tab selected. The page includes a commit history table with one entry: "Melsteroni Initial commit". Below the table, the word "Quickstart" is displayed in a large font.

clone github - Google x nazrug/Quickstart: H x

GitHub, Inc. [US] `https://github.com/nazrug/Quickstart`

Apps ★ Bookmarks Color Theme + Pa Color Scheme De iWantHue Flossie Teacakes: 45 Awesome Free Cute Japanese Co The Graphics

This repository Search Pull requests Issues Gist ⚡ To Do

nazrug / Quickstart ⌵

Unwatch 1 ★ St

<> Code ! Issues 0 📦 Pull requests 0 ||| Boards 📊 Burndown 📖 Wiki 📈 Pulse 📊 Graphs ⚙ Set

How to get started on Github (as a scientist) — Edit

🕒 1 commit 🌿 1 branch 📦 0 releases 🏠 1

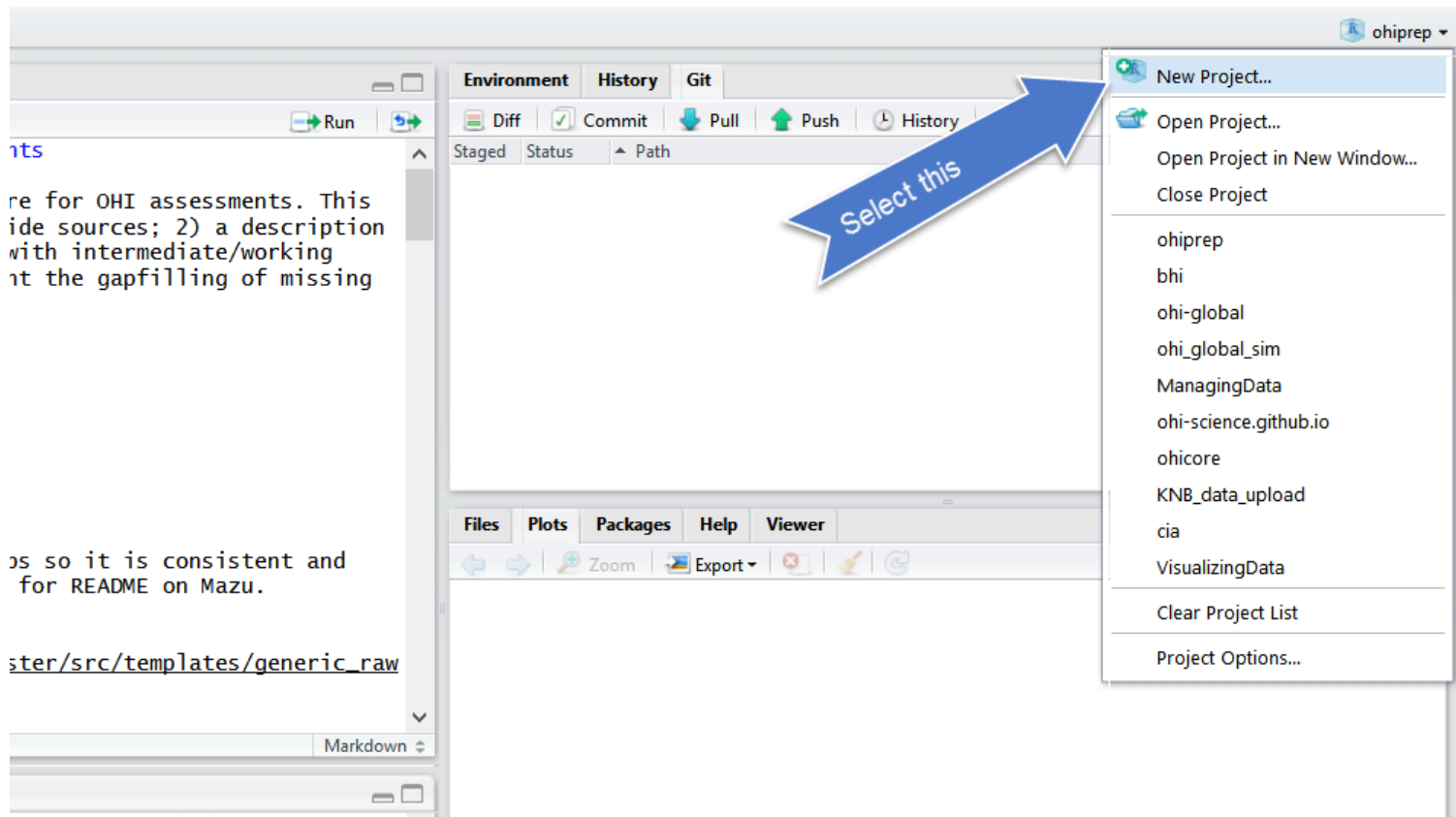
Branch: master ▾ New pull request New file Upload files Find file HTTPS ▾ `https://github.com/nazrug` 📄

👤 Melsteroni Initial commit	Latest commit
📄 README.md	Initial commit

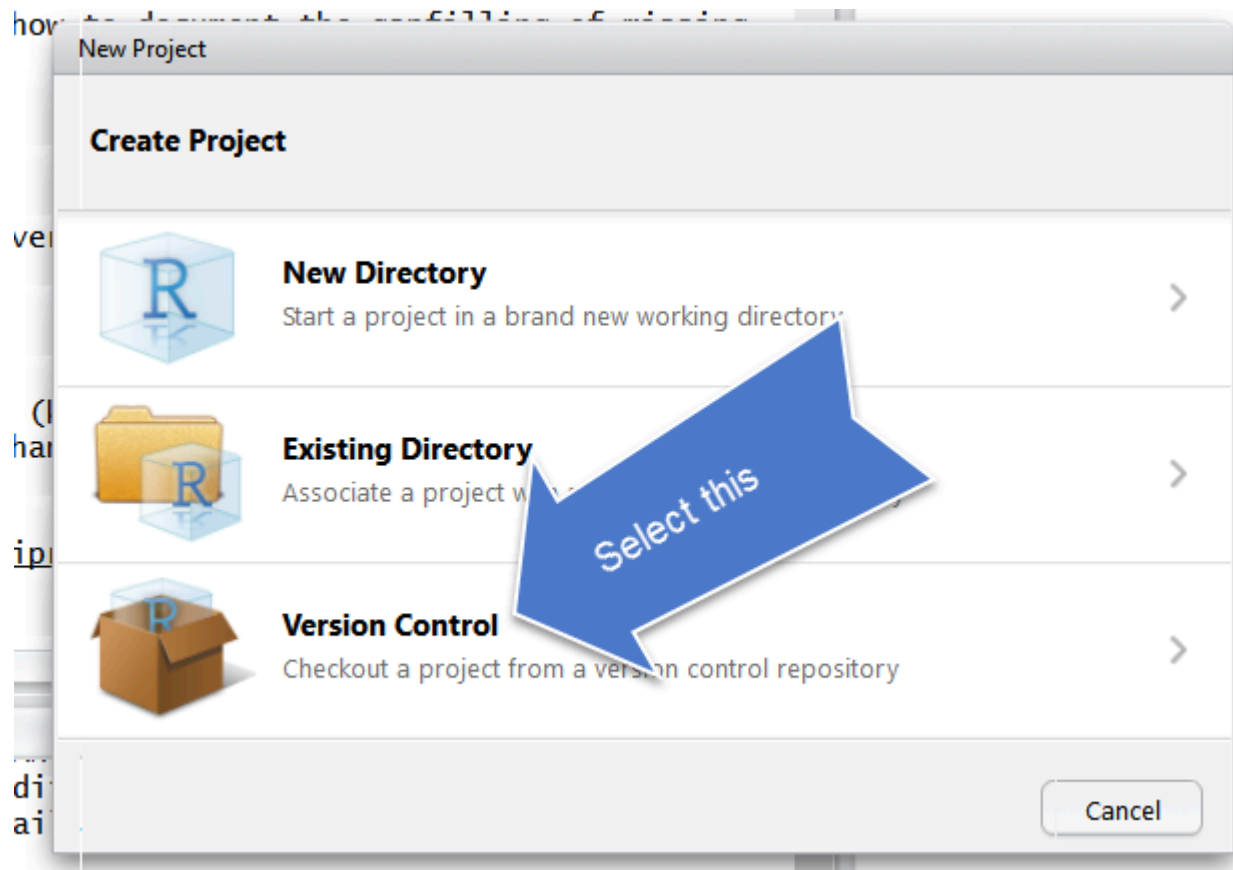
📖 README.md

# Quickstart

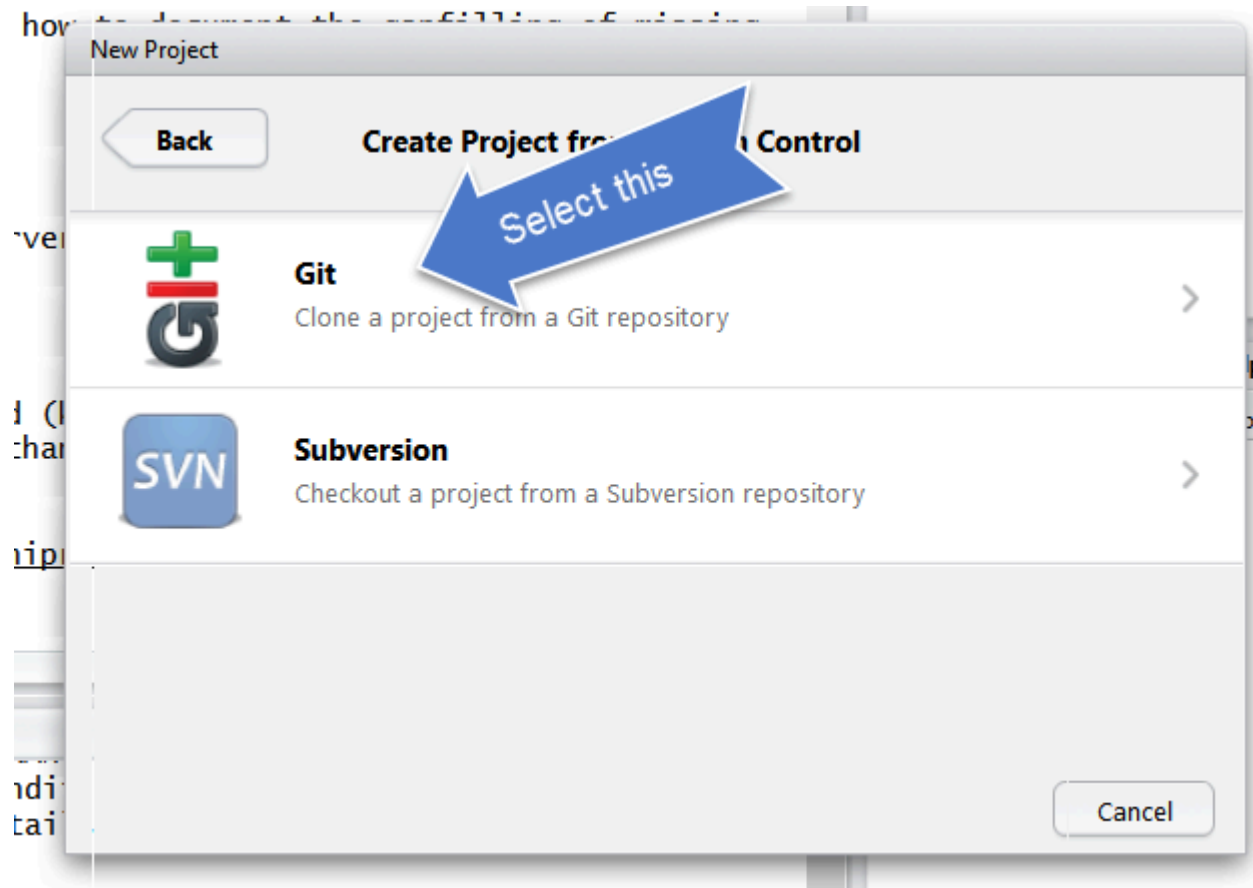
**Step 2:** from RStudio, go to New Project (also in the File menu).



### Step 3: Select Version Control

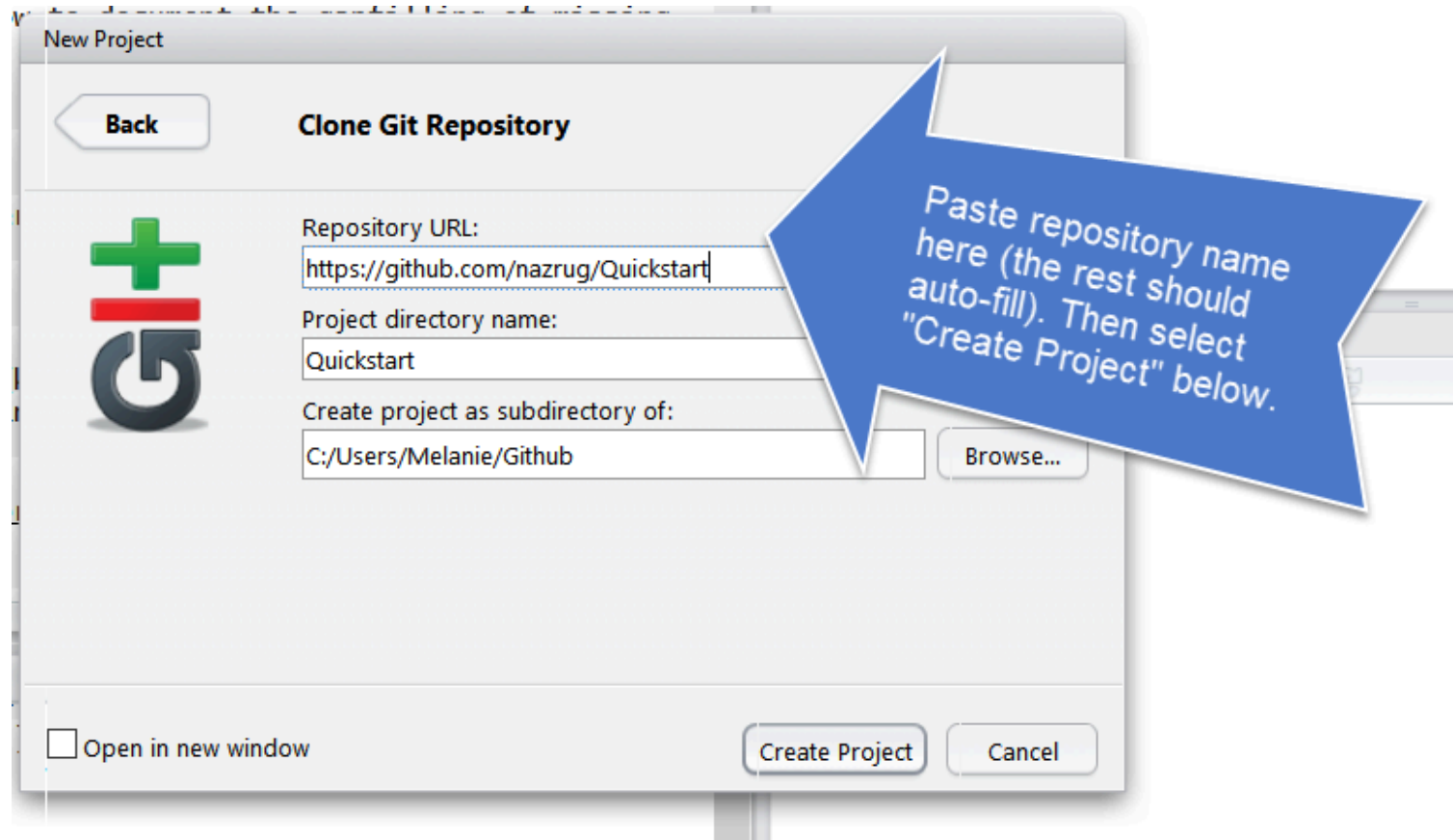


## Step 4: Select Git



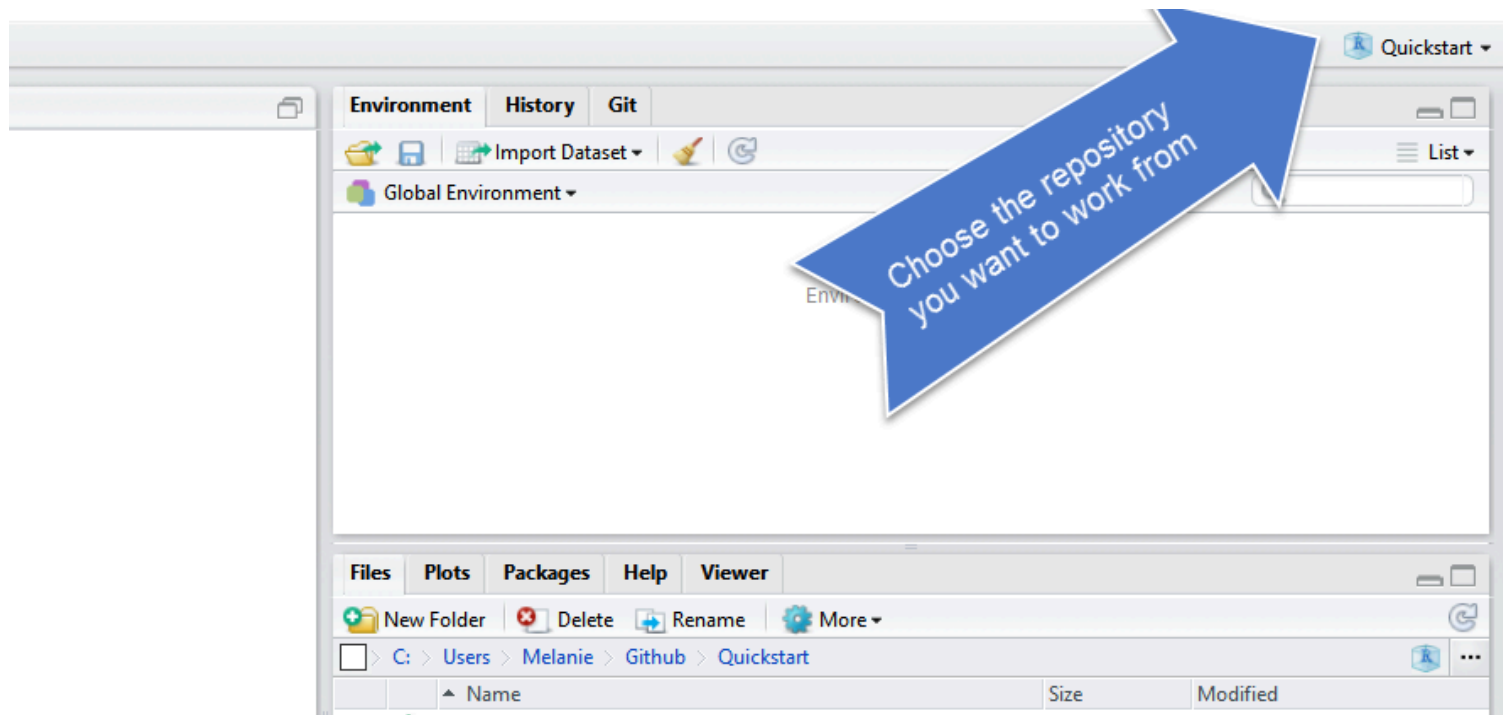
**Step 5:** Paste it in the Repository URL field, and type **tab** to autofill the Project Directory name. Make sure you keep the Project Directory Name **THE SAME** as the repository name from the URL.

Save it in your github folder (click on Browse) to do this.

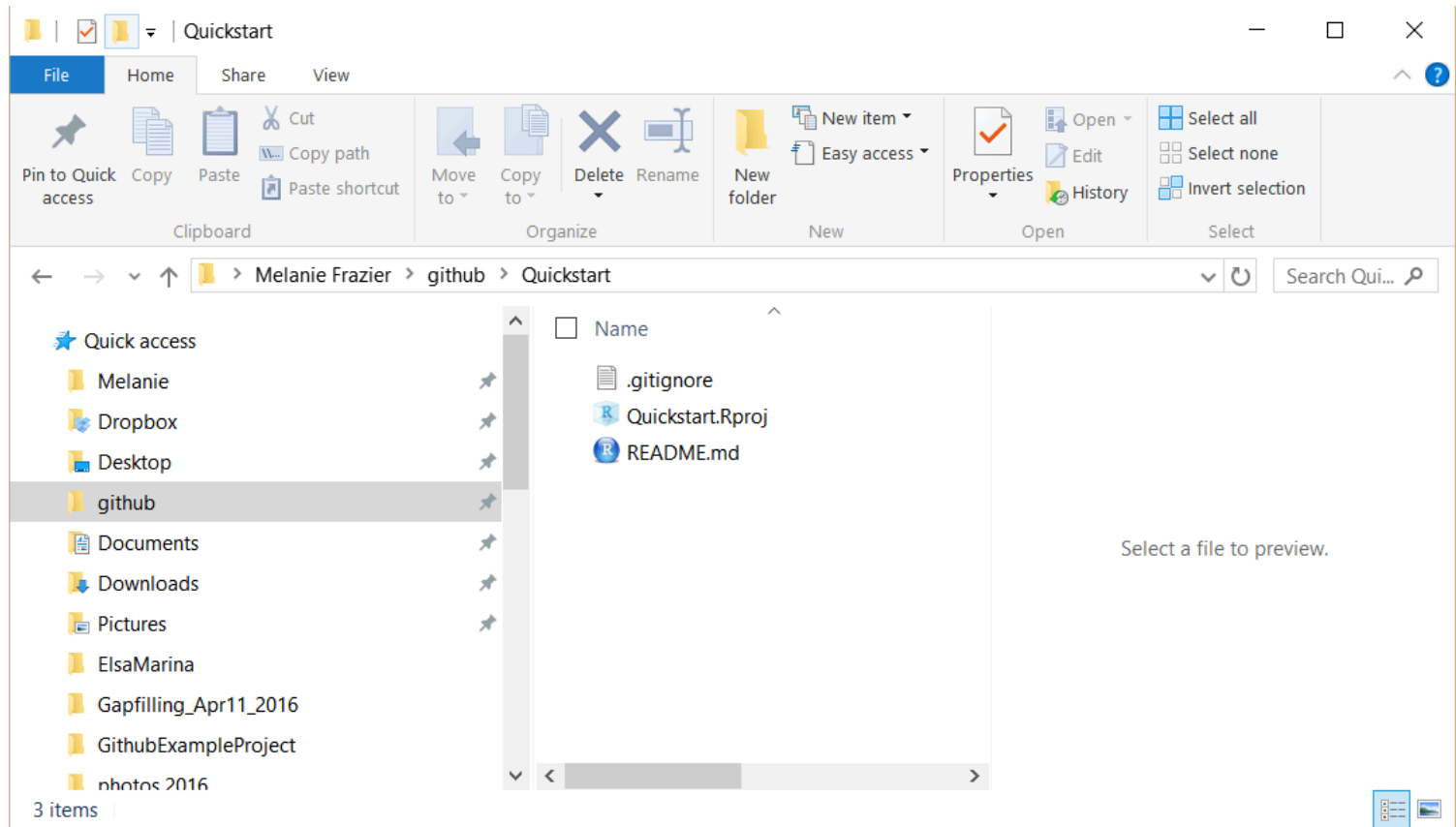




If everything went well, the repository will be added to the list located here:



And the repository will be saved to the Github folder on your computer:



# Inspect your repository

Notice a few things in our repo here:

1. Our working directory is set to `~/github/my-repo`. This means that I can start working with the files I have in here without setting the filepath. This is that when we cloned this from RStudio, it created an RStudio project, which you can tell because:
  - `.RProj` file, which you can see in the Files pane.
  - The project is named in the top right hand corner
2. We have a git tab! This is how we will interface directly to Github.com

~/github/my-repo - gh-pages - RStudio

Go to file/function

Addins

my-repo

Console

~/github/my-repo/

R version 3.4.1 (2017-06-30) -- "Single Candle"  
Copyright (C) 2017 The R Foundation for Statistical Computing  
Platform: x86\_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.

> |

Environment History Connections Git

Diff Commit

Staged Status Path

Files Plots Packages Help Viewer

New Folder Delete Rename More

Home > github > my-repo

	Name	Size	Modified
	..		
<input type="checkbox"/>	.gitignore	40 B	Jul 13, 2016, 6
<input type="checkbox"/>	.Rhistory	2.4 KB	Jul 15, 2016, 2
<input type="checkbox"/>	ggplot_intro.html	1.4 MB	Jul 15, 2016, 2
<input type="checkbox"/>	ggplot_intro.rmd	2 KB	Jul 15, 2016, 2
<input type="checkbox"/>	my-repo.Rproj	205 B	Nov 19, 2017,
<input type="checkbox"/>	README.md	475 B	Jul 13, 2016, 8
<input type="checkbox"/>	rmarkdown.html	774.9 KB	Jul 13, 2016, 6
<input type="checkbox"/>	rmarkdown.rmd	913 B	Jul 13, 2016, 6

When you first clone a repo through RStudio, RStudio will add an `.Rproj` file to your repo. And if you didn't add a `.gitignore` file when you originally created the repo on GitHub.com, RStudio will also add this for you. These will show up with little yellow ? icons in your git tab. This is GitHub's way of saying: "I am responsible for tracking everything that happens in this repo, but I haven't seen these files yet. Do you want me to track them too?" (We'll see that when you click the box to stage them, they will turn into `As` because they have been added to the repo.

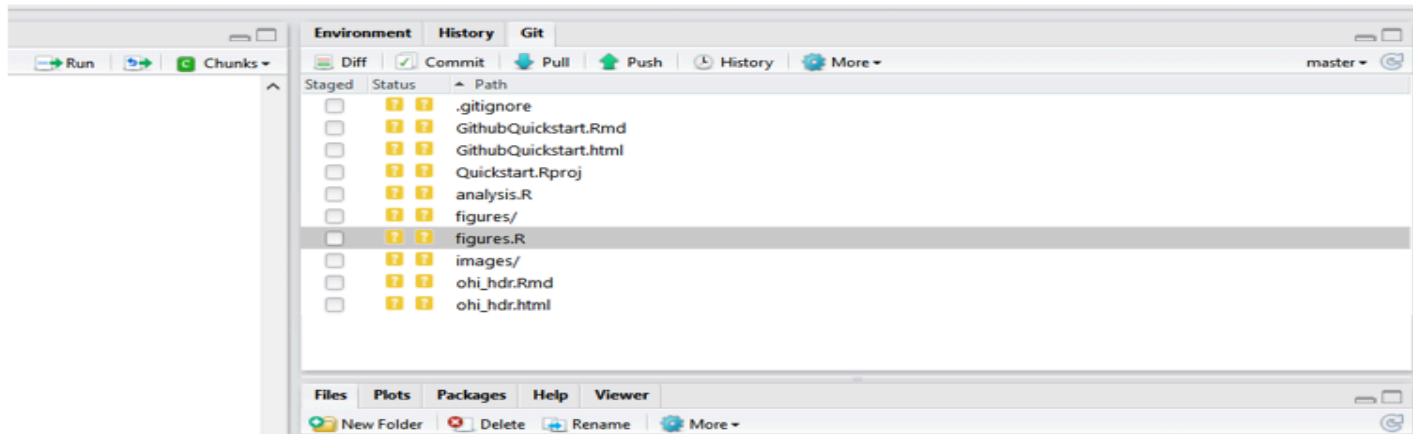
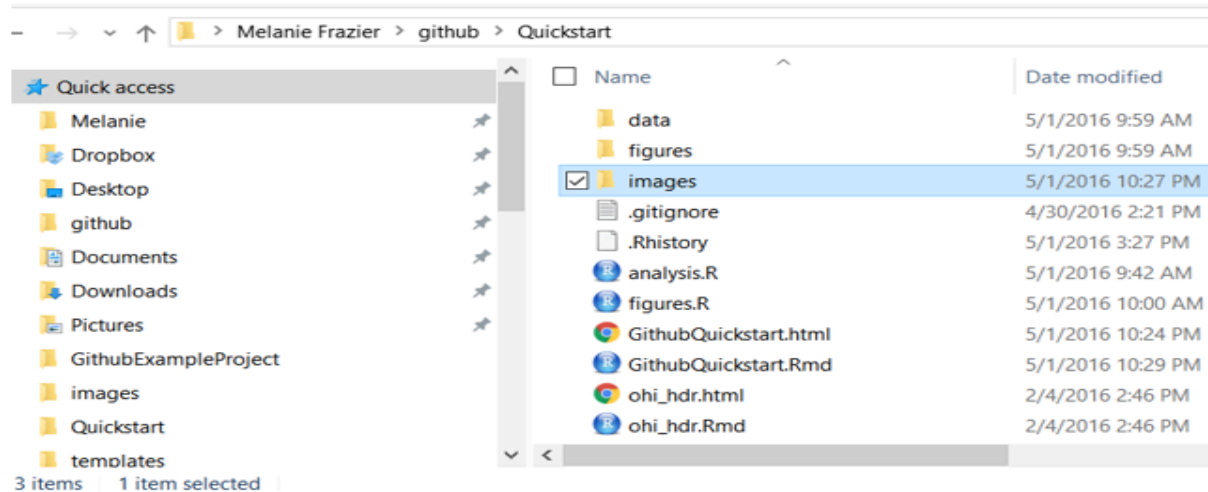
# Add files to our local repo

The repository will contain:

- `.gitignore` file
- `README.md`
- Rproj And, I typically create the following:
- folders for “data” and “figures”
- R scripts
- etc. I’m going to go to the Finder (Windows Explorer on a PC) and copy a file into my repository from there. And then I’m going to go back to RStudio – it shows up in the git tab! So the repository is being tracked, no matter how you make changes to it (changes do not have to be done only through RStudio).

To make changes to the repository, you will work from your computer (“local Github”).

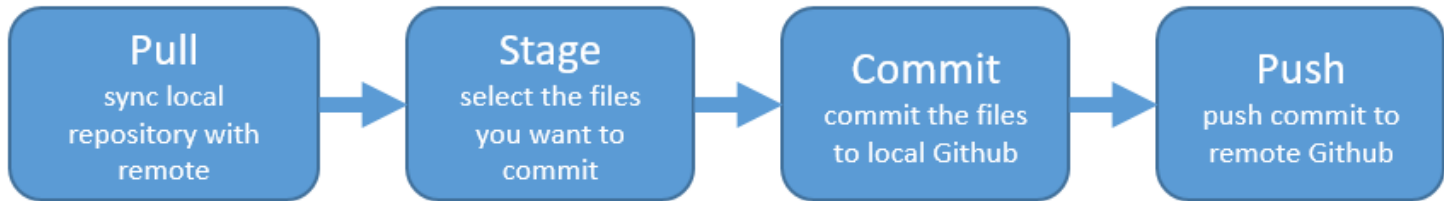
When files are changed in the local repository, these changes will be reflected in the Git tab of RStudio:





# Sync from RStudio to GitHub

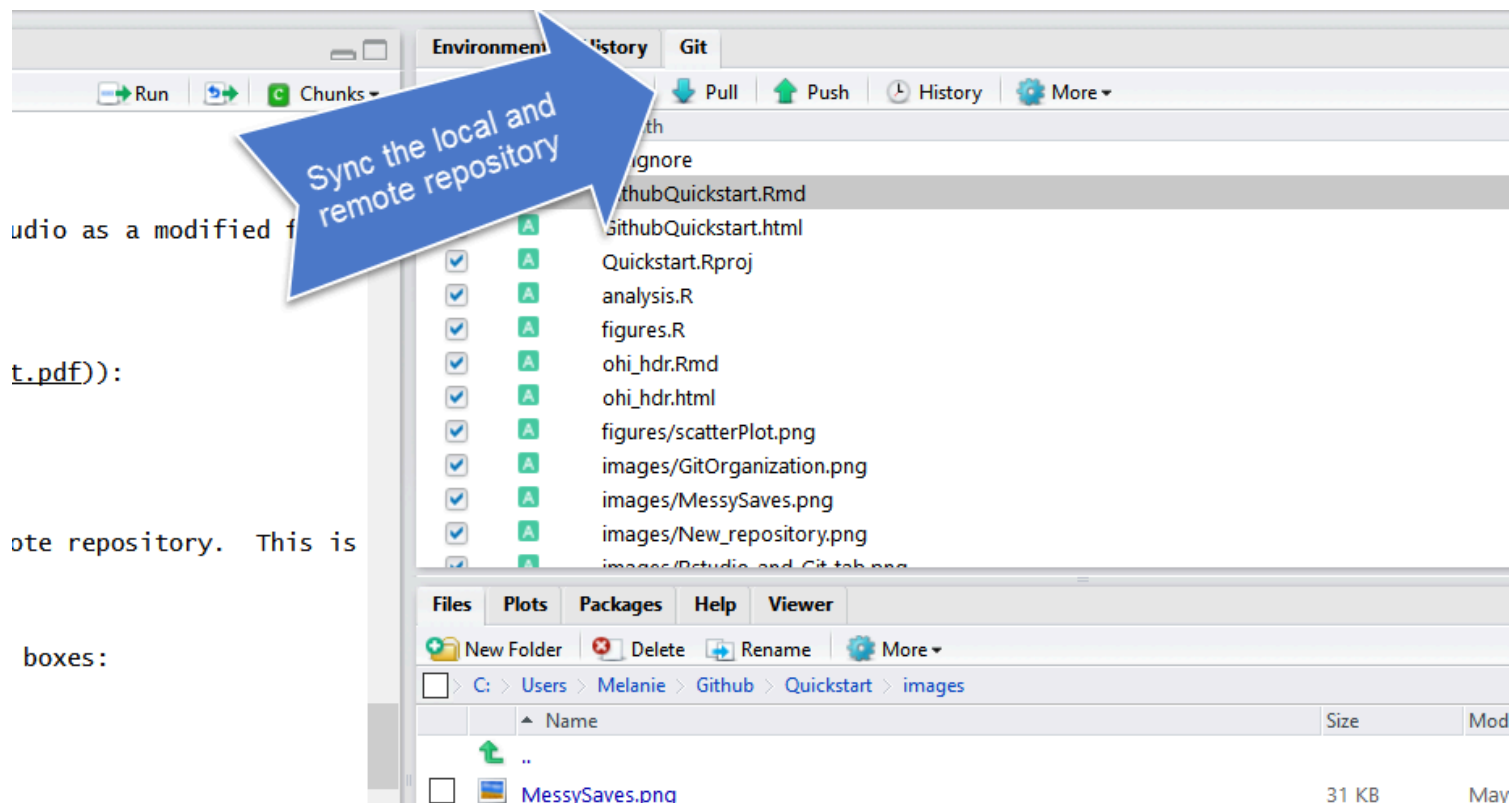
When you are ready to commit your changes, you follow these steps:



We walk through this process below:

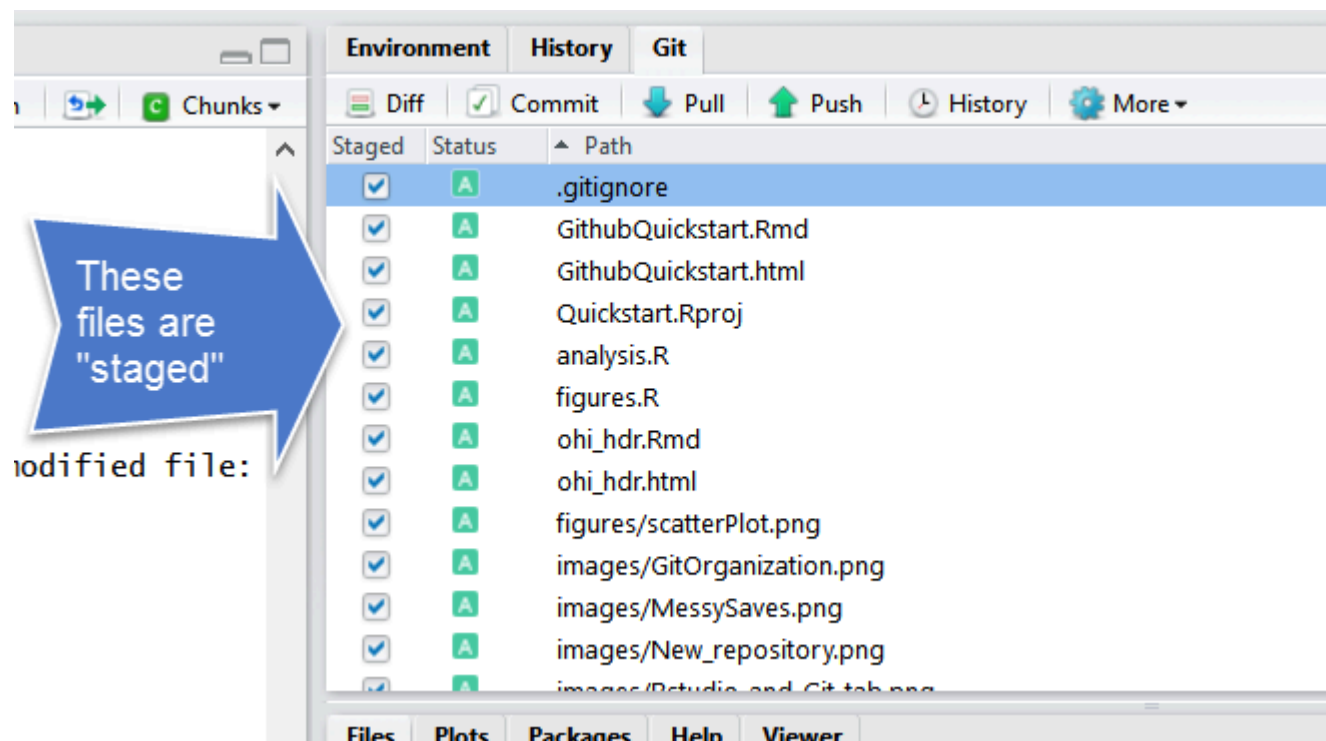
## Pull

From the Git tab, “Pull” the repository. This makes sure your local repository is synced with the remote repository. This is very important if other people are making changes to the repository or if you are working from multiple computers.



## Stage

Stage the files you want to commit. In RStudio, this involves checking the “Staged” boxes:



# Commit

The screenshot shows the RStudio Git interface. At the top, the 'Commit' tab is selected. Below it, a list of files to be committed is shown, including .gitignore, GithubQuickstart.Rmd, GithubQuickstart.html, Quickstart.Rproj, analysis.R, and figures.R. A blue arrow points to the 'Commit' button. Another blue arrow points to the 'Commit message' field, which contains the text 'Adding documents to Quickstart repo'. A third blue arrow points to the 'Commit' button at the bottom right. A fourth blue arrow points to the 'Staged' column in the file list. The bottom of the screenshot shows a preview of the commit message, which includes the title 'Github Quickstart for Scientists (e.g., not social coders)', the author information, and the output of the R script.

1. On Git tab select "Commit"

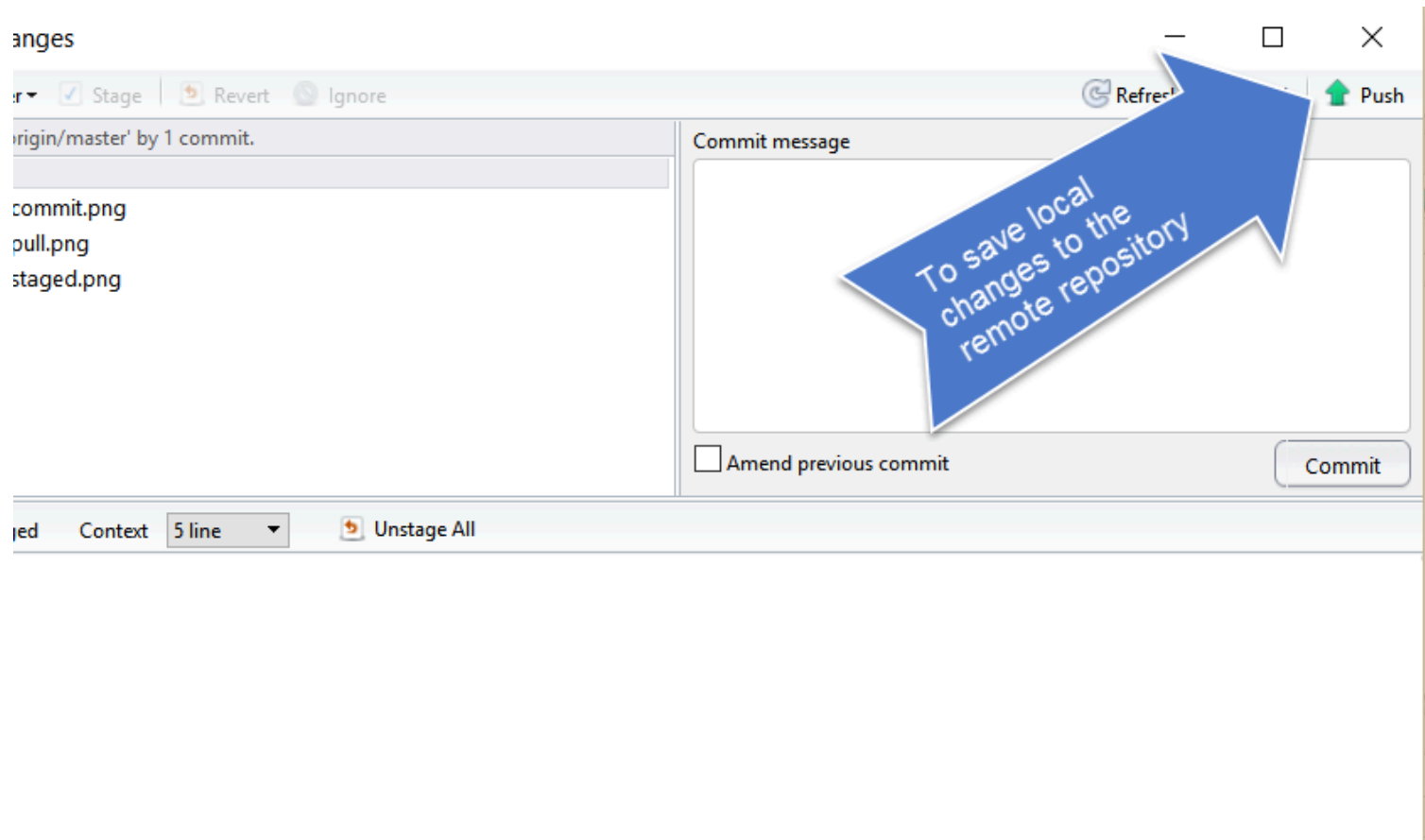
2. Can select files and see changes

3. Add a commit description

4. Click the "Commit" button

```
1 ---
2 title: "Github Quickstart for Scientists (e.g., not social coders)"
3 author: "Compiled on `r date()` by `r Sys.info()$user`"
4 output:
5   html_document:
6     toc: true
7     number_sections: true
8     theme: cerulean
9     highlight: haddock
10    includes:
11      in_header: ohi_hdr.html
12    pdf_document:
13      toc: true
14    ---
15
16 # What is Git/Github?
```

# Push



# Explore remote Github

The files you added should be on github.com:

The screenshot shows the GitHub repository page for 'nazrug / Quickstart'. The browser address bar shows 'https://github.com/nazrug/Quickstart'. The repository name 'nazrug / Quickstart' is displayed at the top, along with 'Unwatch', 'Star' (1), and 'Fork' (0) buttons. Below the repository name, there are tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Boards', 'Burndown', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected, showing the commit history. The commit history table lists the following commits:

Commit Message	Commit Hash	Time Ago
Melsteroni Adding Quickstart files	cad70a4	21 minutes ago
figures		Adding Quickstart files
images		Adding Quickstart files
.gitignore		Adding Quickstart files
GithubQuickstart.Rmd		Adding Quickstart files
GithubQuickstart.html		Adding Quickstart files
Quickstart.Rproj		Adding Quickstart files
README.md		Initial commit
analysis.R		Adding Quickstart files
figures.R		Adding Quickstart files
ohi_hdr.Rmd		Adding Quickstart files
ohi_hdr.html		Adding Quickstart files

## **Your turn!**

This time let's edit an existing file instead of adding something new. Open your README file by clicking on it in the Files pane (lower right corner). Write a few lines of text, save, and see what happens in your Git Tab. Sync it to your remote repository (Github.com).

Also, go to your Finder/Windows Explorer, and copy-paste something into your local GitHub repo. Then go back to RStudio and confirm that git tracked it. Remember, git will track anything within that folder (the way Dropbox does), it's not specific to RStudio!

# Create a new R Markdown file

Let's go back to RStudio, and get ourselves back into learning R. We are going to use R Markdown so that you can write notes to yourself in Markdown, and have a record of all your R code. Writing R commands in the console like we did this morning is great, but limited; it's hard to keep track of and hard to efficiently share with others. Plus, as your analyses get more complicated, you need to be able to see them all in one place.

Go to File > New File > R Markdown (or click the green plus in the top left corner).

Let's set up this file so we can use it for the rest of the day. I'm going to delete all the text that is already there and write some new text.

Now, let's save it. I'm going to call my file `ggplot2.Rmd`.

Then, sync your file to GitHub.

What if a file doesn't show up in the Git tab and you expect that it should? Check to make sure you've saved the file. If the filename is red with an asterix, there have been changes since it was saved. Remember to save before syncing to GitHub!



# Committing - how often? Tracking changes in your files

Whenever you make changes to the files in Github, you will walk through the Pull -> Stage -> Commit -> Push steps.

I tend to do this every time I finish a task (basically when I start getting nervous that I will lose my work). Once something is committed, it is very difficult to lose it.

One thing that I love about Github is that it is easy to see how files have changed over time.

Usually I compare commits through [github.com](https://github.com).

You can click on the commits to see how the files changed from the previous commit.