Lecture 2: R Markdown, Version Control with Git(Hub), and Other Productivity Tools

James Sears*
AFRE 891/991 SS 25
Michigan State University

*Parts of these slides are adapted from **"Advanced Data Analytics"** by Nick Hagerty and **"Data Science for Economists"** by Grant McDermott.

Table of Contents

Prologue

Prologue

Before we dive in, let's double check that we all have

- ☑ Installed **R**.
- ☑ Installed **RStudio**.
- ☑ Signed up for an account on **Github**
- ☑ Installed **Git** and **Github Desktop**
- ☑ Log into your Github account on Github Desktop

R Markdown

R Markdown

Before we dive into version control, let's chat about **R Markdown**.

R Markdown is a document type that allows for integration of R code and output into a Markdown document.

Resources:

- Website: **rmarkdown.rstudio.com**
- **R Markdown Cheatsheet**
- Book: **R Markdown: The Definitive Guide** (Yihui Xie, JJ Allaire, and Garrett Grolemund)

R Markdown

Before we dive into version control, let's chat about **R Markdown**.

R Markdown is a document type that allows for integration of R code and output into a Markdown document.

Other points:

- We'll be completing assignments using R Markdown.
- FWIW, my lecture slides and notes are all written in R Markdown too. (E.g. This slide deck is built using the **xaringan** package with the metropolis theme.)

R Markdown: Getting Started

- [x] Installed **R**.
- [x] Installed **RStudio**.
- [ ] Add the `rmarkdown` package

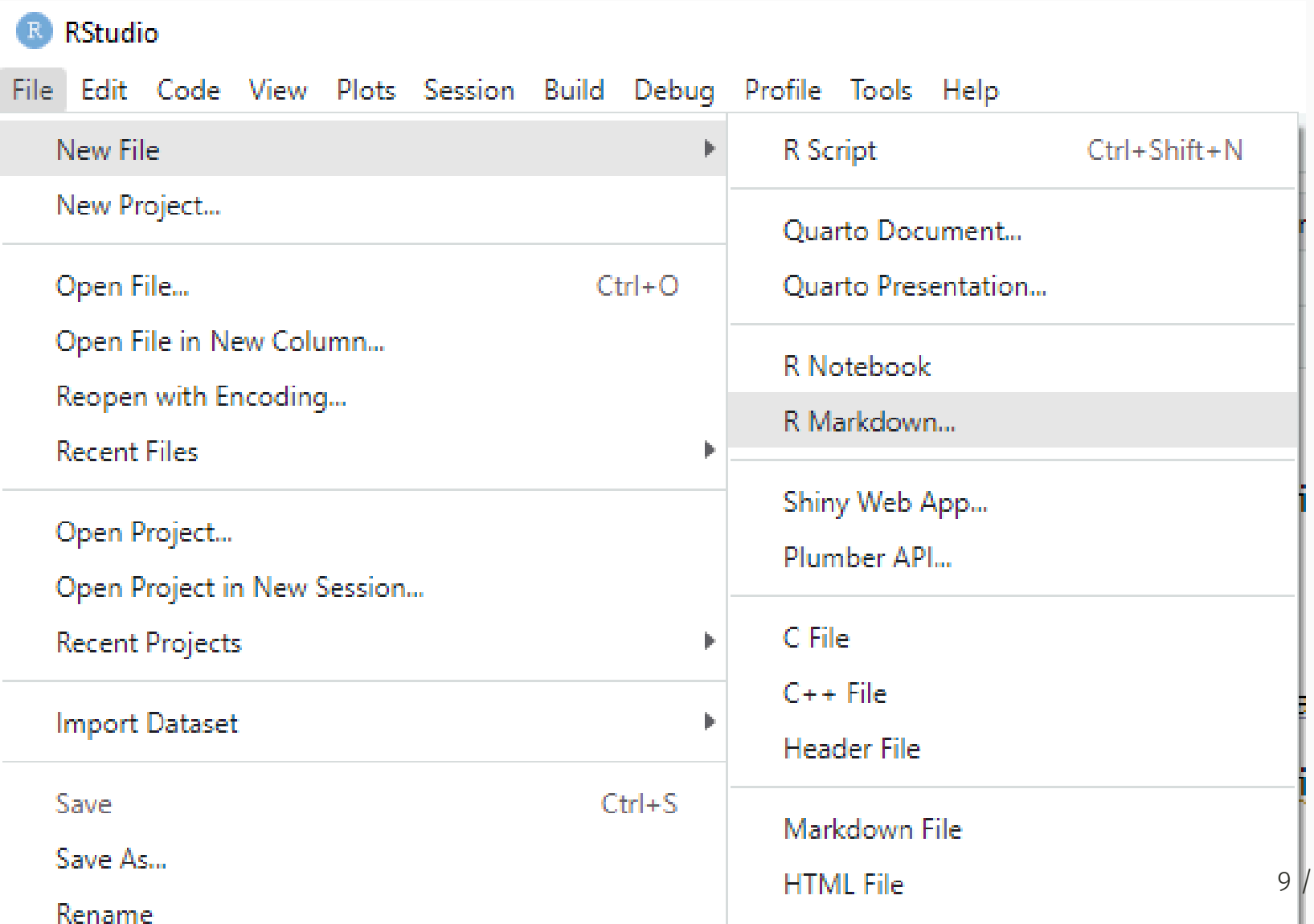
```
install.packages("rmarkdown")
```

- [ ] Install LaTeX

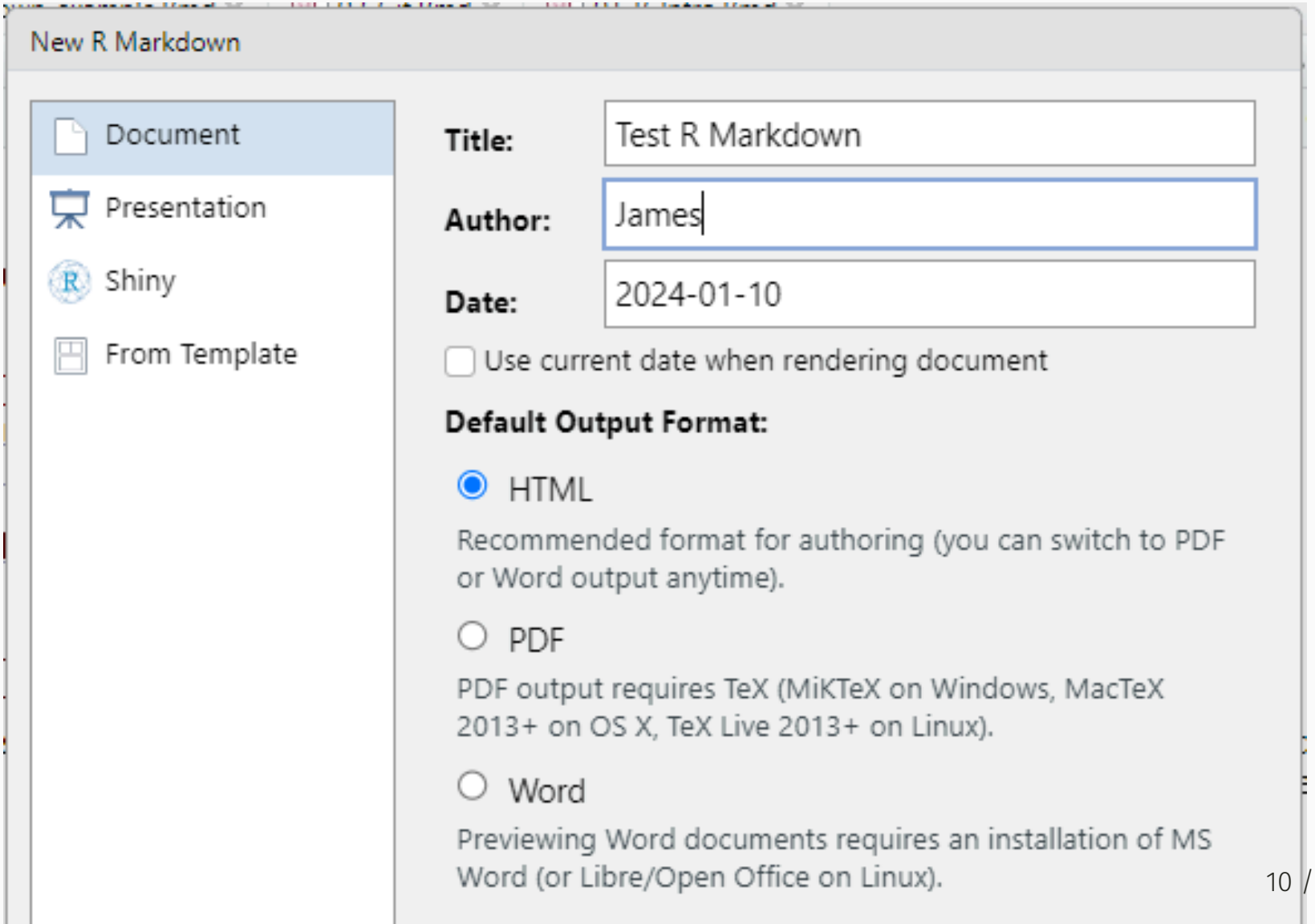
  - If just for this, can use **TinyTex**

```
# Install only if you don't have LaTeX already
install.packages("tinytex")
tinytex::install_tinytex()
```

R Markdown: Creating a New .Rmd File



R Markdown: Creating a New .Rmd File



R Markdown: Creating a New .Rmd File

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

02-Productivity.Rmd | Untitled1 | R-Markdown-example.Rmd | 02-Git.Rmd | 01-R-Intro.Rmd

Knit on Save | Knit | Run

Source | Visual | Outline

```
1  ---
2  title: "Test R Markdown"
3  author: "James"
4  date: "2024-01-10"
5  output: html_document
6  ---
7
8  ```{r setup, include=FALSE}
9  knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML,
   PDF, and MS Word documents. For more details on using R Markdown see
   <http://rmarkdown.rstudio.com>.
15
16 When you click the **Knit** button a document will be generated that includes both content
   as well as the output of any embedded R code chunks within the document. You can embed an R
   code chunk like this:
17
18 ```{r cars}
19 summary(cars)
20 ```
```

Modify Chu

R Markdown Components

R Markdown combines

1. **Markdown:** lightweight markup language
2. **LaTeX:** typesetting for math
3. **R:** include code and generate output

Let's do some practice: **open a new .Rmd file** and try adding content as we go

Markdown

Markdown allows for formatting text in a lightweight way

I highly recommend the handy **Markdown Guide** for more details

Markdown: Heading

Headings emphasize text and add chunks to your script

Largest headingwith one leading # (slide title above)

Second Largest (##)

Third Largest (###)

Getting Smaller... (####)

Normal Text for comparison

Markdown: Text Format

Bold text with **your text**

Italicize with *single asterisks*

Add `code text` with grave accents (the back tick symbol)

- `
- The other output of the tilde key `~` on keyboard

End a line with two spaces to start a new paragraph

- or leave a line space between sentences

Can also start a new line with backslash (\)

Markdown: Text Format

Add superscripts^{2} with ^carets^

Add ~~strikethroughs~~ with ~~double tildes~~

Add a line break (horizontal rule)

---

with ***

Markdown: Text Format

Draw **tables** using | and -

Col A	Col B	Col C
This	is	a
Table		wow

Markdown: Lists

Add an **ordered list** with **1.**

1. First Item
2. Second Item
3. No need to change the number - keep using 1. It will automatically update.

Add an **unordered list** with *** or -**

- A thing
- Another related thing
  - Indent to nest
    1. Can mix ordered and unordered

Markdown: Inputs

Add a **link** with []()

- [text label](URL)
- Add direct link with <link> **https://www.markdownguide.org**

Add an image with

- ![alt text](URL)

practice by adding `images/smile.png`:

R Markdown: LaTeX

Another advantage of R Markdown is that it integrates LaTeX functionality for typesetting math.

Add an **inline equation** with TeX

$$Var(X) = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n} \qquad Y_{it} = \beta_0 + \beta_1 X_{it} + \epsilon_{it}$$

Add multiple rows of LaTeX with

$$

LaTeX lines here

$$

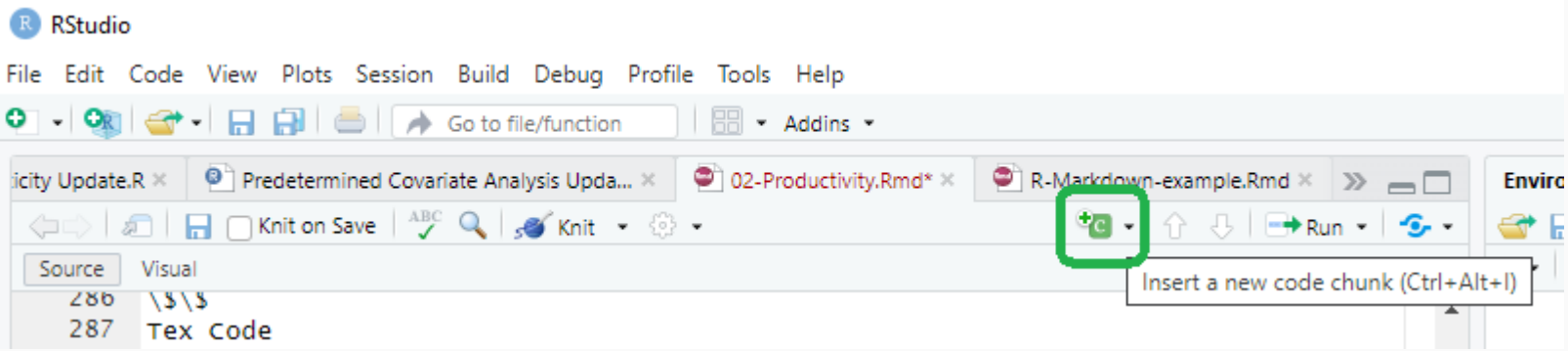
Use the **standard LaTeX commands** for symbols/characters

R Markdown: R Code

R code is primarily executed with **code chunks**

Add a chunk with

- `Cmd + Option + I (Ctrl + Alt + I on PC)`
- The `Insert` button in the UI
- Manually type



R Markdown: R Code Chunks

```
307
308 ```{r}
309
310 ```
311
```

Code chunks allow us to add as many lines of code as we want

- Output will appear underneath after executing the full chunk
- Can customize whether it runs, how output is displayed
- Can run manually
  - Line by line with `Cmd/Ctrl + Enter`
  - Entire chunk with `Run Entire Chunk` button

R Markdown: R Code Chunk Options

You can **add chunk options** in brackets after `r` and separated by commas.

Some commonly-used options include:

- **Chunk label** (`ex_chunk`)
- `include = FALSE` will run the chunk but hide it from the final document
- `eval = FALSE` will display code without evaluating it
- `results = 'hide'` runs code but hides output from the final document

```
52
53 ```{r sum, echo = FALSE, warning = FALSE}
54 2+2
55
56 ```
```

R Markdown: R Code Chunk Options

You can **add chunk options** after `r` and separated by commas.

Some commonly-used options include:

- `echo = FALSE` runs the code but hides the chunk from the final document
- `error = FALSE` (`warning: FALSE`) will hide error (warning) messages generated by the code

- LOTS of options for output figures: figure size (fig.width, fig.height, fig.dim), output document scale (out.width, out.height), alignment (fig.align), caption (fig.cap)

  Learn more **about chunk options here**

R Markdown: R Code

You can call R objects from earlier chunks **inline** with

`` `r ` ``

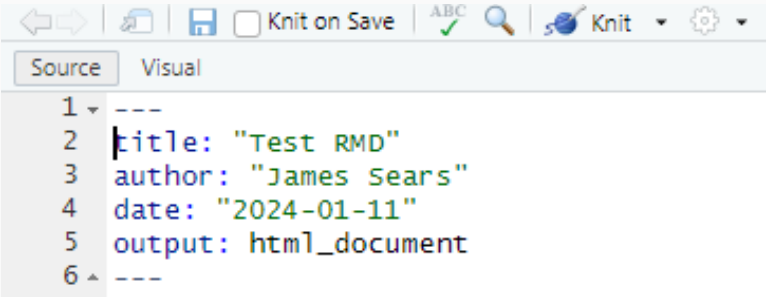
```
four = 2+2
```

This can output in line with text: 2 + 2 = 4

R Markdown File Organization

1. Header

RStudio automatically builds the R Markdown file from a template, which begins with a **header**



- Title
- Author
- Date
- Output Format
  - Main options[1]: HTML (`html_document`), PDF (`pdf_document`), LaTeX (`latex_document`), or Word (`word_document`)

1: See **CH 3 of "R Markdown: The Definitive Guide" for more on how to customize output formats**

2. R Setup

By default, RStudio adds a **setup** code chunk next.

```
8  ```{r setup, include=FALSE}
9  knitr::opts_chunk$set(echo = TRUE)
10 ```
11
```

- Can set global options
- Useful as your preamble
- For **R Notebooks**, this will automatically be run and is the only place where you can change your working directory

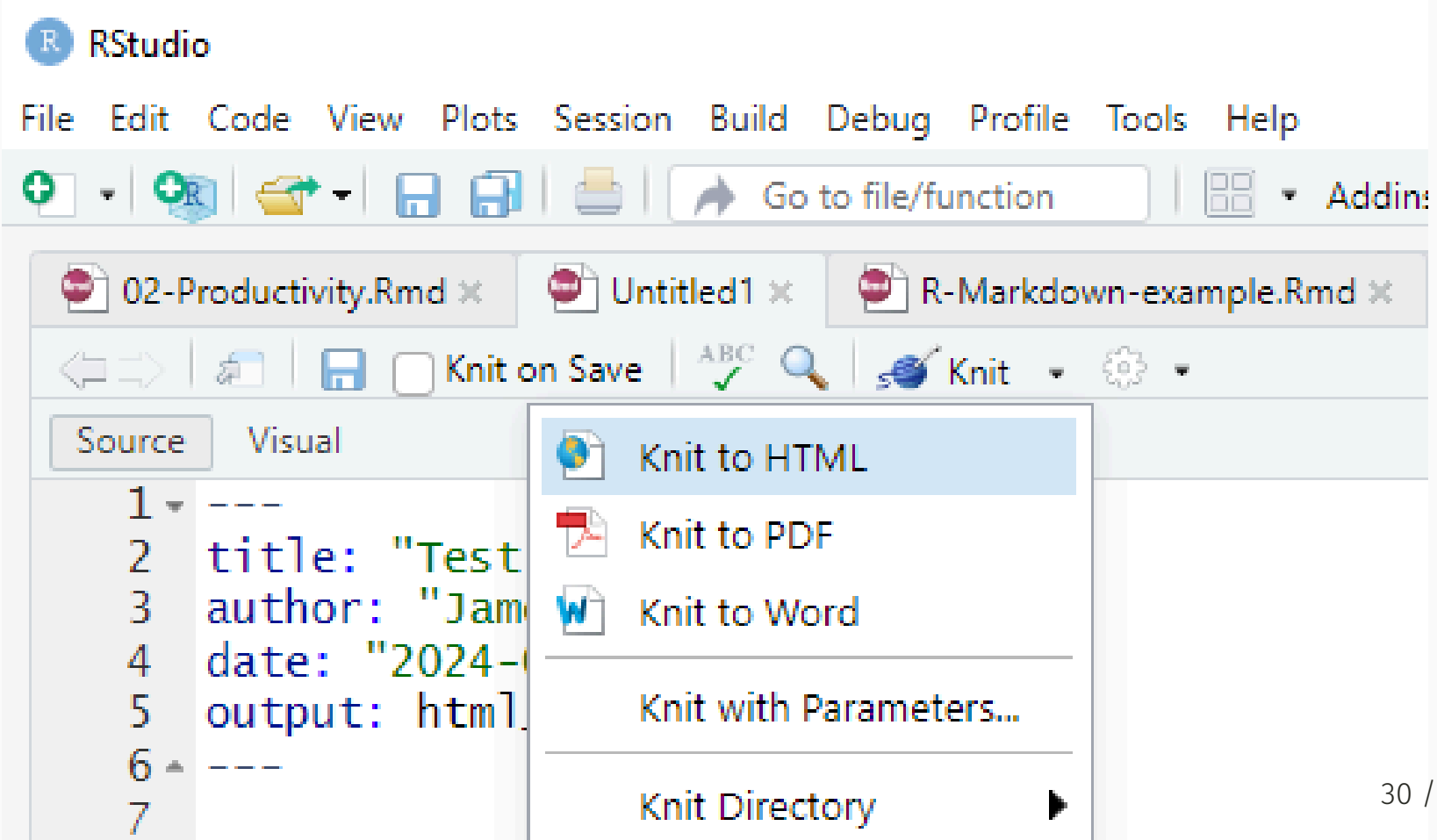
3. Contents

From here on you can build the report/notebook as needed for the task.

- Add any writing and outside graphics or **bibTeX citations**
- Add code chunks to carry out desired analysis
- Employ sections and formatting to structure the document as desired

Compiling/Knitting

When you are ready to compile your final document, use the `Knit` button or `Ctrl/Cmd + Shift + K`



R Markdown: Knit to Compile Output

F:/OneDrive - Michigan State University/Teaching/MSU 2023-2024/AFRE 891 SS23/Lecture-Slides/02-Productivity/output/R-Markdown-example.html

R-Markdown-example.html | Open in Browser | Find | Publish

Test R Markdown

James

2024-01-10

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Markdown Practice!

Markdown Practice

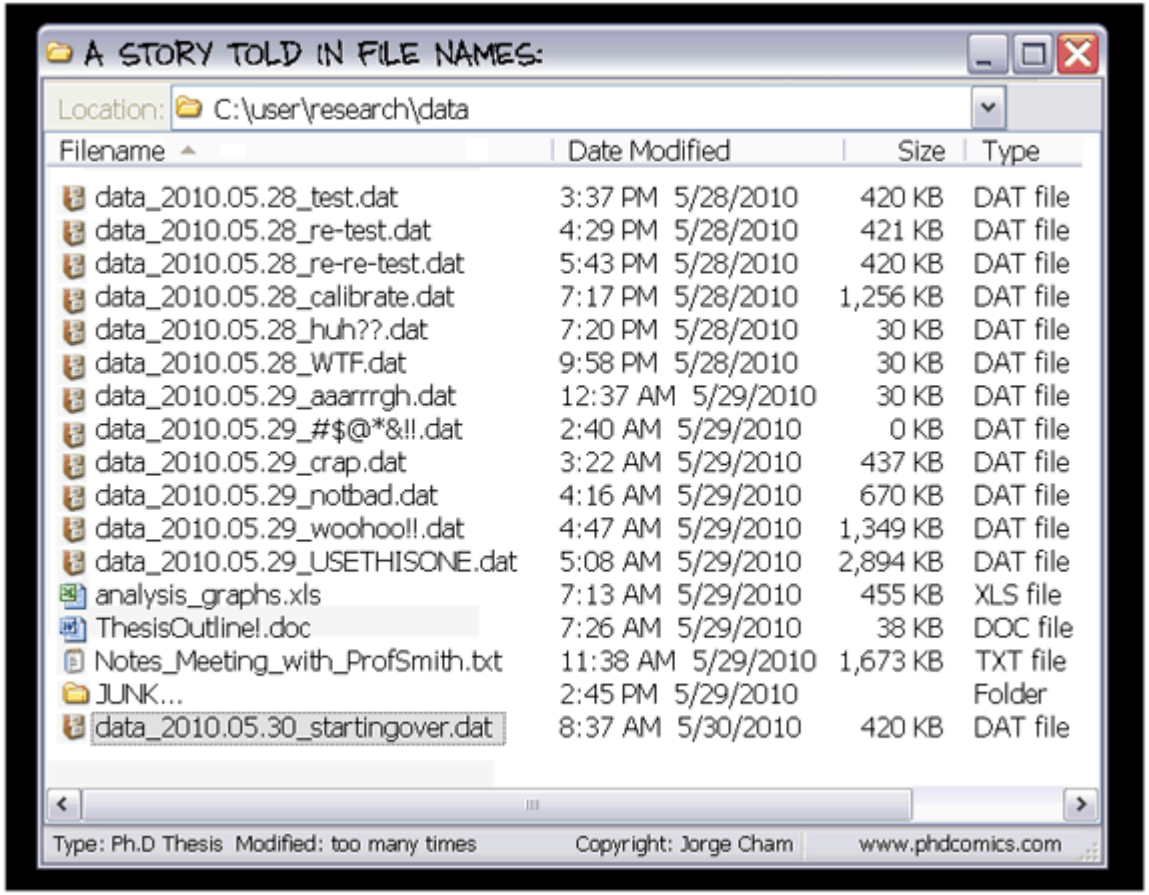
1. Create a new R Markdown file named "R-Markdown-Ex.RMD"
2. In the setup chunk, load the `dslabs` and `tidyverse` packages
   - Use the `data()` function to read in the `divorce_margarine` dataset
3. Add a header labeled "Correlation vs. Causation" and a text explanation below for why we often want to differentiate between the two
4. Add a code chunk with the label `plot`
   - Type the following code:

```
ggplot(divorce_margarine) +
geom_point(aes(x = margarine_consumption_per_capita,
           y = divorce_rate_maine)) +
labs(title = "Relationship between Margarine Consumption and
          Divorce Rates in Maine",
 subtitle = "2000-2009",
 x = "Margarine Consumption per Capita",
y = "Divorce Rate")`
```

5. Knit and save a PDF/HTML copy of the file to the "output" folder

Version Control

Why Use Version Control



Goals of Version Control

While building project folders with the above naming conventions is *fun*, a good **version control system** can solve this problem.

- Save each set of changes sequentially
- Keep track of different versions of a file
- Merge changes from multiple versions/sources

Git(Hub) Solves this Problem

Git

- **Git** is a **distributed version control system**
  - Each team member has a **local copy** of files on their computer
- Imagine if Dropbox and the "Track changes" feature in MS Word had a baby. Git would be that baby.
- In fact, it's even better than that because Git is optimised for the things that economists and data scientists spend a lot of time working on (e.g. code).
- There is a learning curve, but I promise you it's worth it.

Git(Hub) Solves this Problem

GitHub

- It's important to realise that **Git** and **GitHub** are distinct things.
- **GitHub** is an **online hosting platform** that provides an array of services built on top of the Git system. (Similar platforms include Bitbucket and GitLab.)
- Just like we don't *need* Rstudio to run R code, we don't *need* GitHub to use Git... But it will make our lives so much easier.

Git(Hub) for Scientific Research

From software development...

- Git and GitHub's role in global software development is not in question.
- There's a high probability that your favourite app, program or package is built using Git-based tools. (RStudio is a case in point.)

... to scientific research

- Benefits of VC and collaboration tools aside, Git(Hub) helps to operationalise the ideals of open science and reproducibility.[2]
- Journals have increasingly strict requirements regarding reproducibility and data access. GH makes this easy (DOI integration, off-the-shelf licenses, etc.)
- I host **teaching materials** on GH. I even use it to host and maintain my **website**.

2: **Democratic databases: Science on GitHub (Nature)** (Perkel, 2016).

Git(Hub) and RStudio

Seamless Integration

One of the (many) great features of RStudio is how well it integrates version control into your everyday workflow.

- Even though Git is a completely separate program to R, they feel like part of the same "thing" in RStudio.
- This next section is about learning the basic Git(Hub) commands and the recipe for successful project integration with RStudio.

I also want to bookmark a general point that we'll revisit many times during this course:

- The tools that we're using all form part of a coherent **data science ecosystem**.
- Greatly reduces the cognitive overhead ("aggregation") associated with traditional workflows, where you have to juggle multiple programs and languages at the same time.

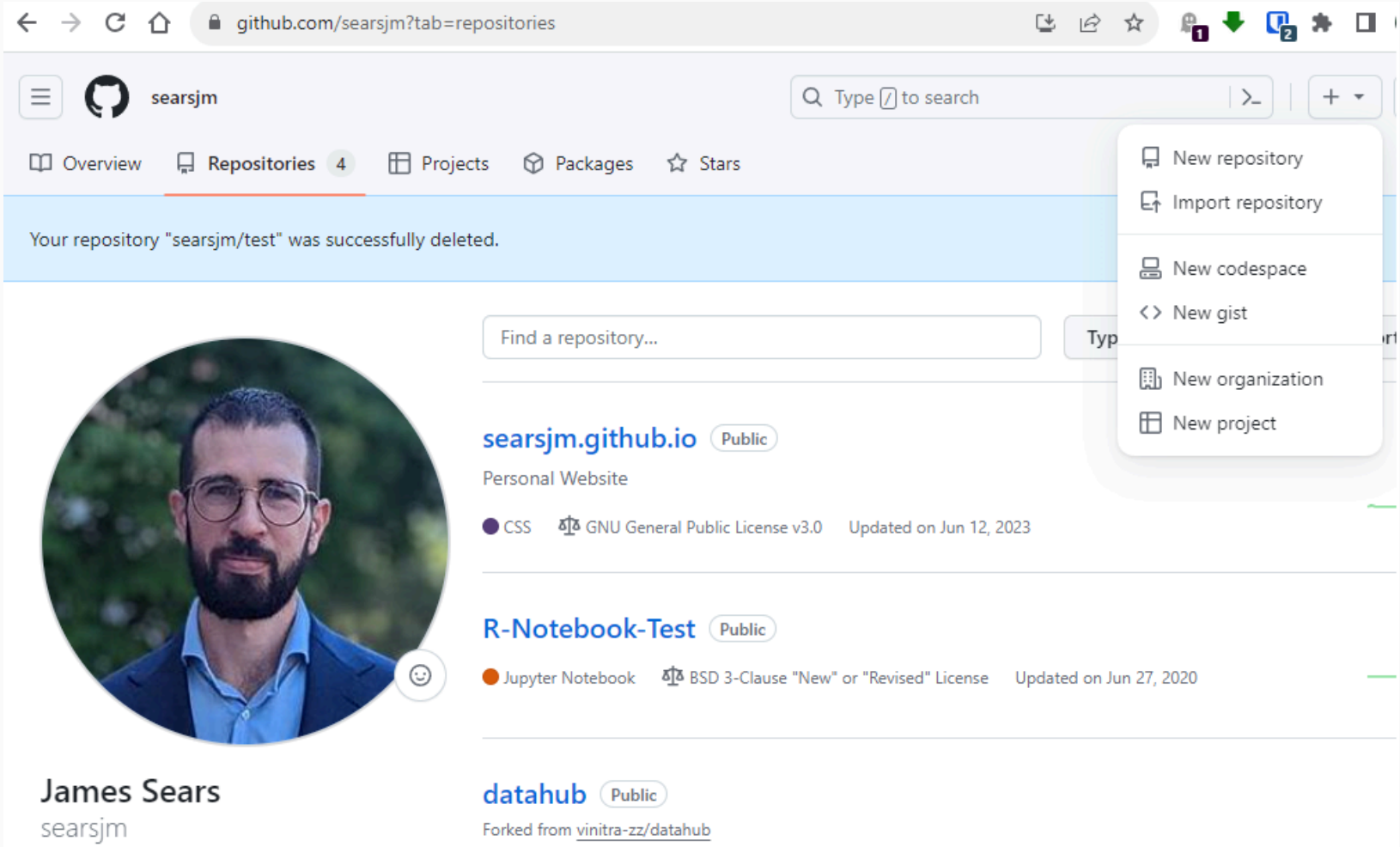
Link a GitHub Repo to an RStudio

The starting point for our workflow is to link a GitHub repository (i.e. "repo") to an RStudio Project. Here are the steps we're going to follow:

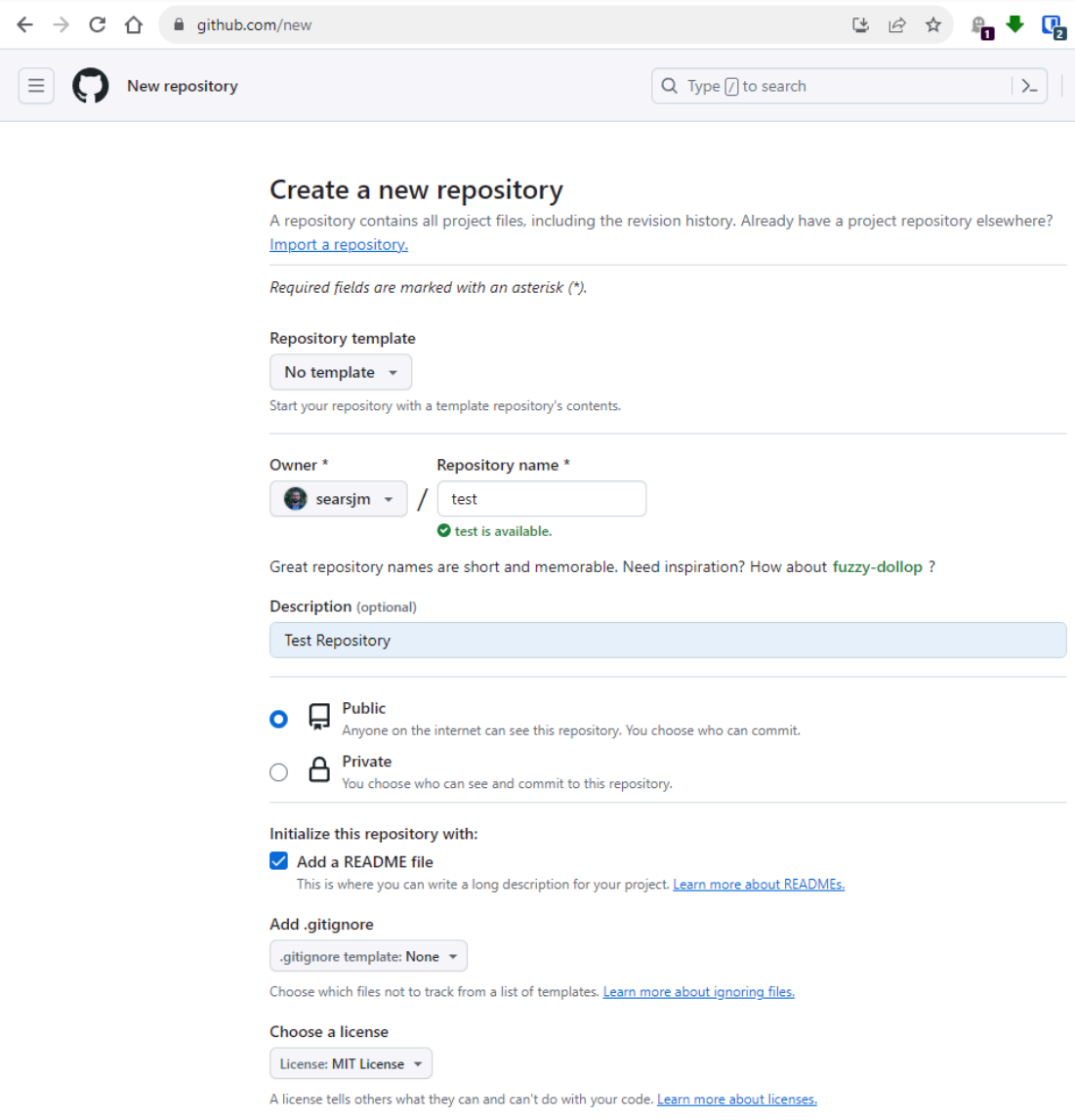
1. Create the repo on GitHub and initialize with a README.
2. Copy the HTTPS/SSH link (the green "Clone or Download" button).[1]
3. Open up RStudio.
4. Navigate to **File -> New Project -> Version Control -> Git**.
5. Paste your copied link into the "Repository URL:" box.
6. Choose the project path ("Create project as subdirectory of:") and click **Create Project**.

Now, I want you to practice by these steps by creating your own repo on GitHub — call it "test" — and cloning it via an RStudio Project.

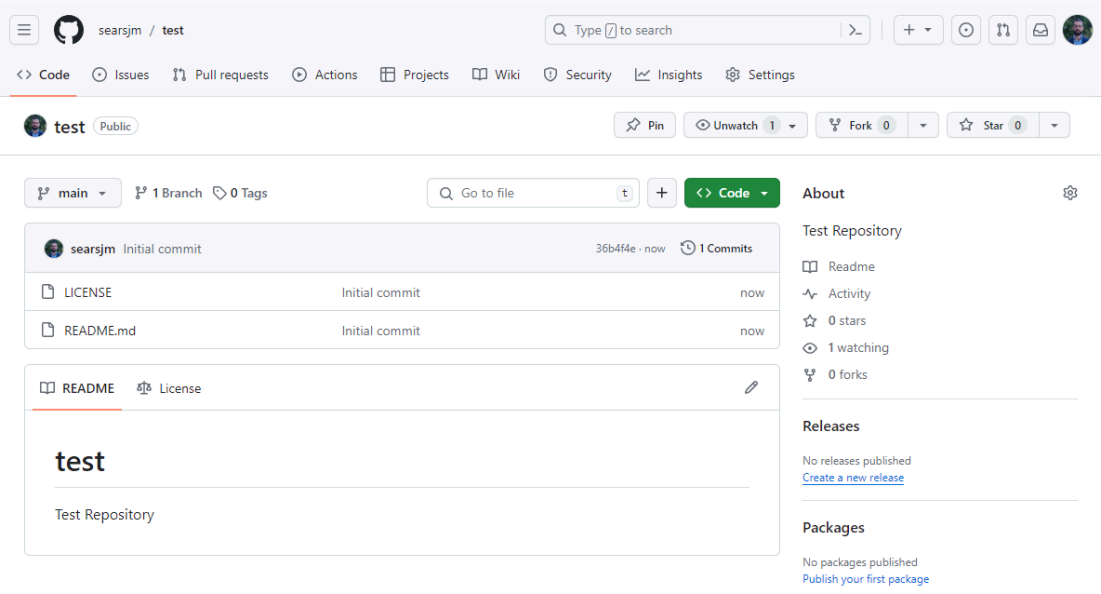
Create a Repository on GitHub (Repo)



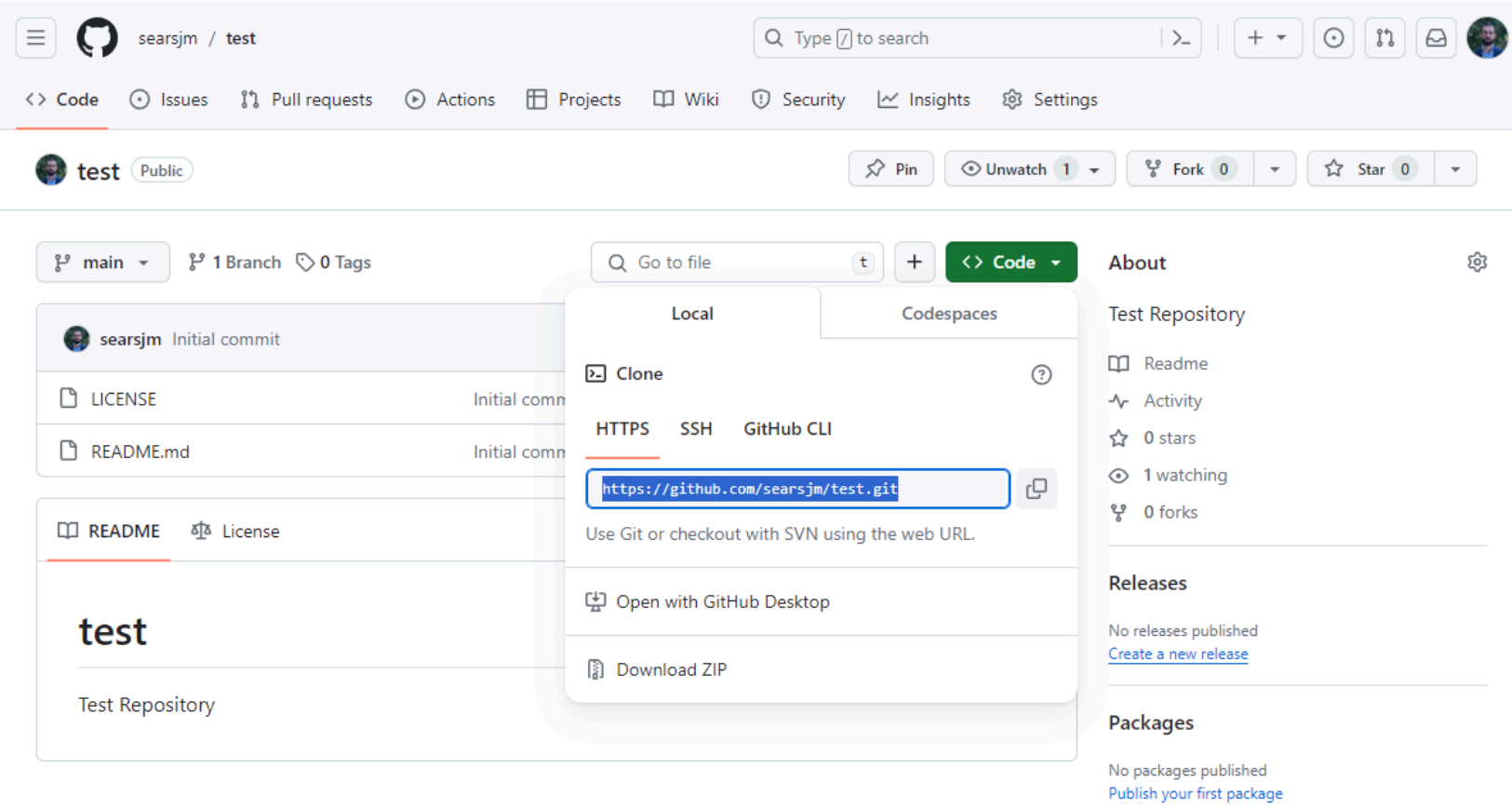
Create a Repository on GitHub (Repo)



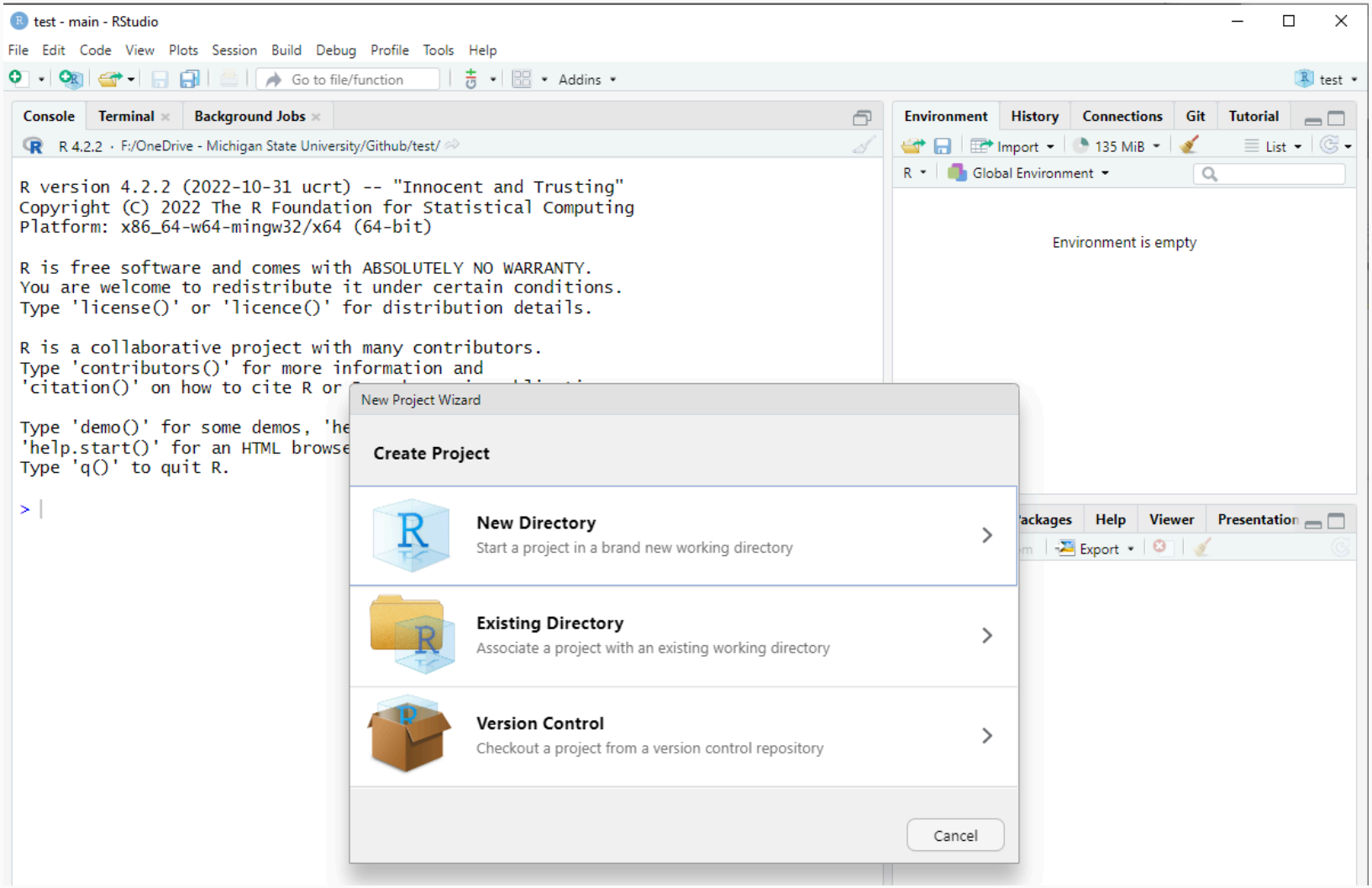
Create a Repository on GitHub (Repo)



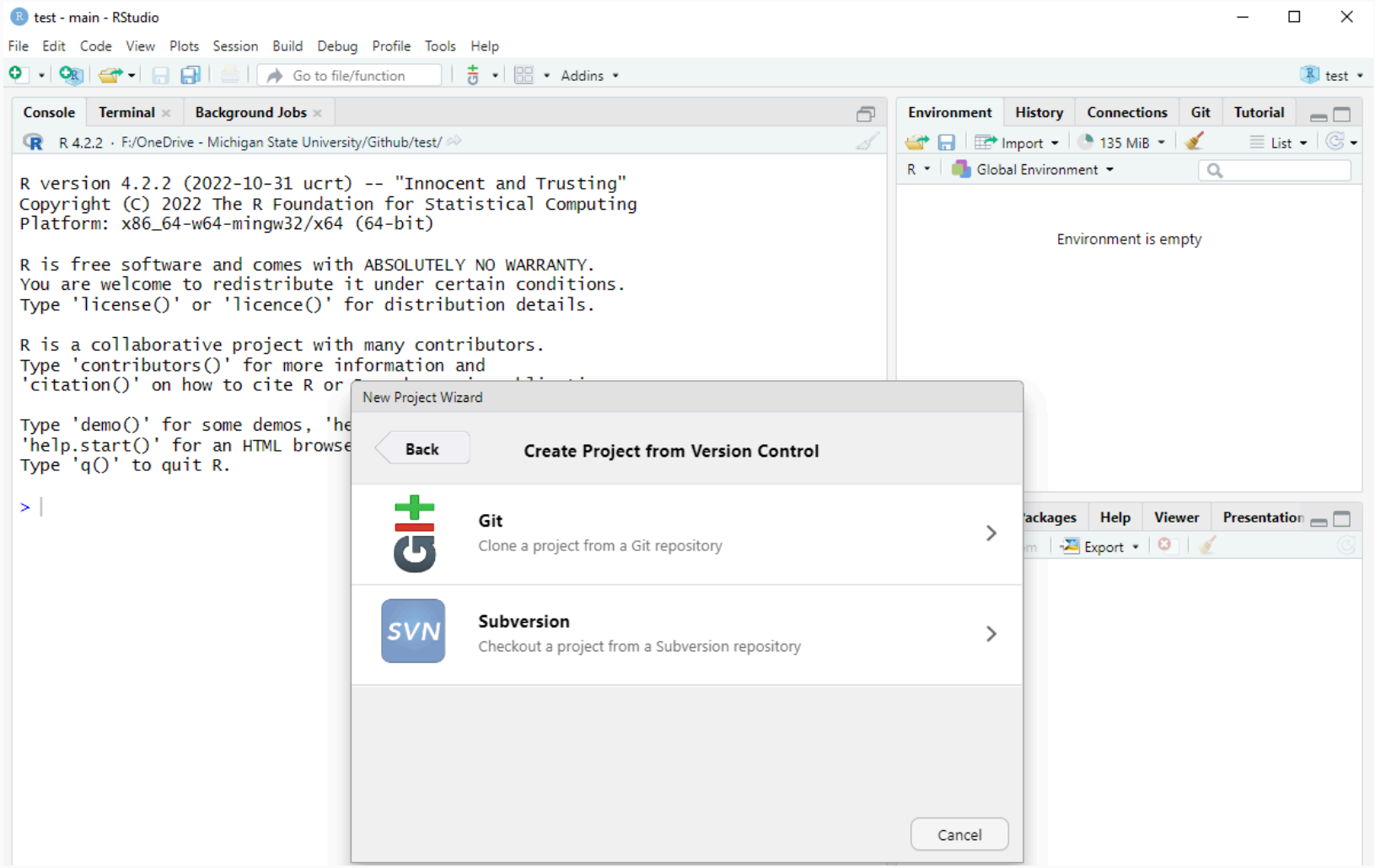
Create a Repository on GitHub (Repo)



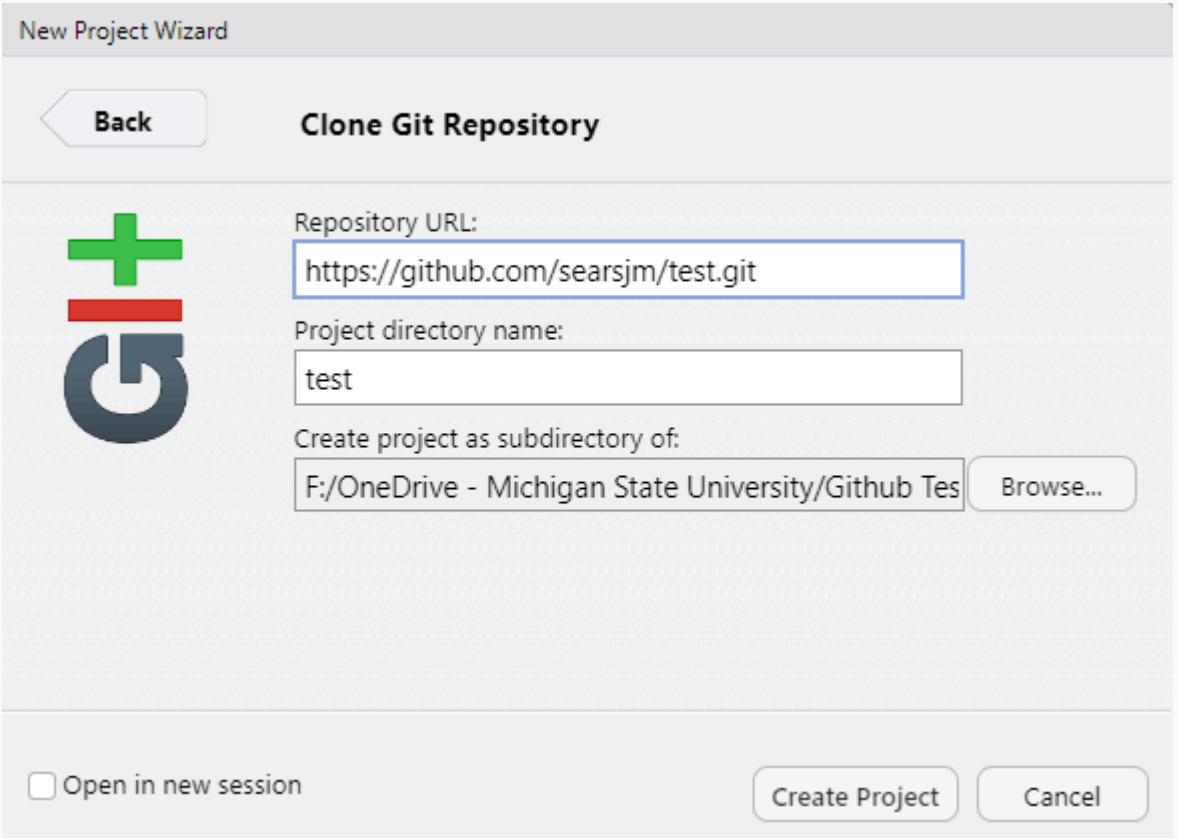
Copy Repo Link



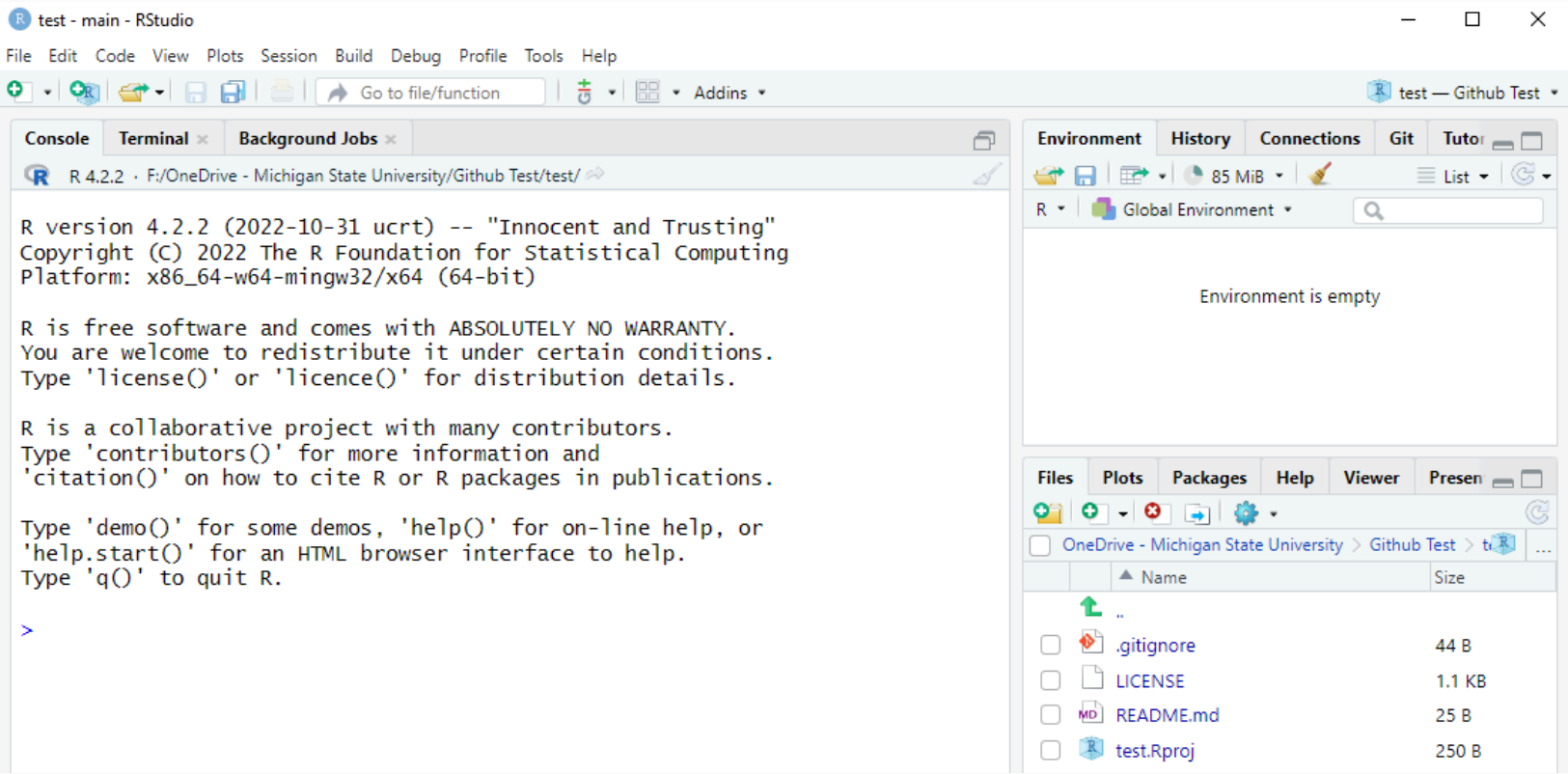
Add Repo into RStudio



Add Repo into RStudio



Add Repo into RStudio



Making Local Changes

Look at the **top-right panel** in your RStudio IDE. Do you see the **Git** tab?

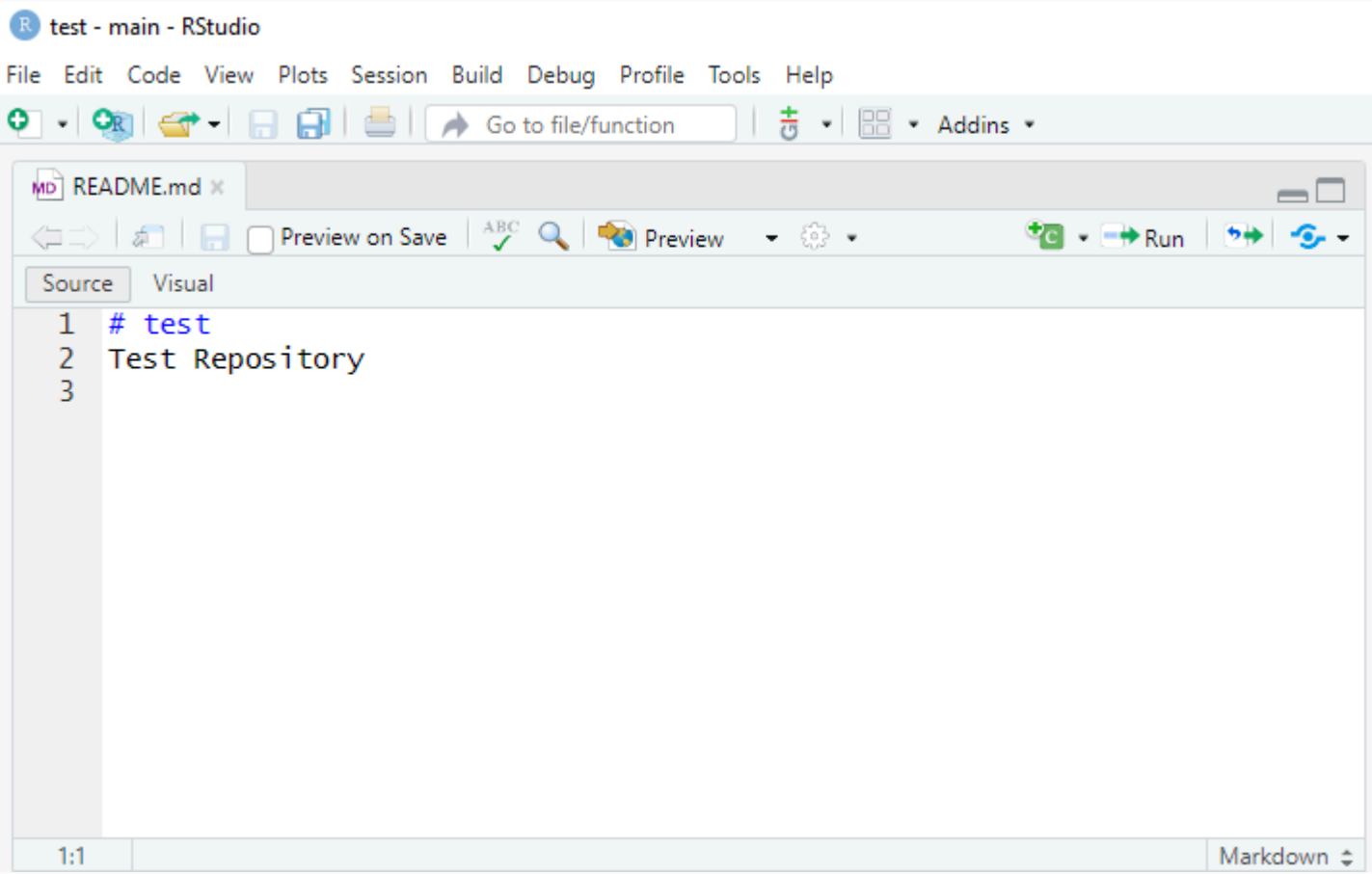
- Click on it.
- There should already be some files in there, which we'll ignore for the moment.[1]

Now open up the README file (see the "Files" tab in the bottom-right panel).

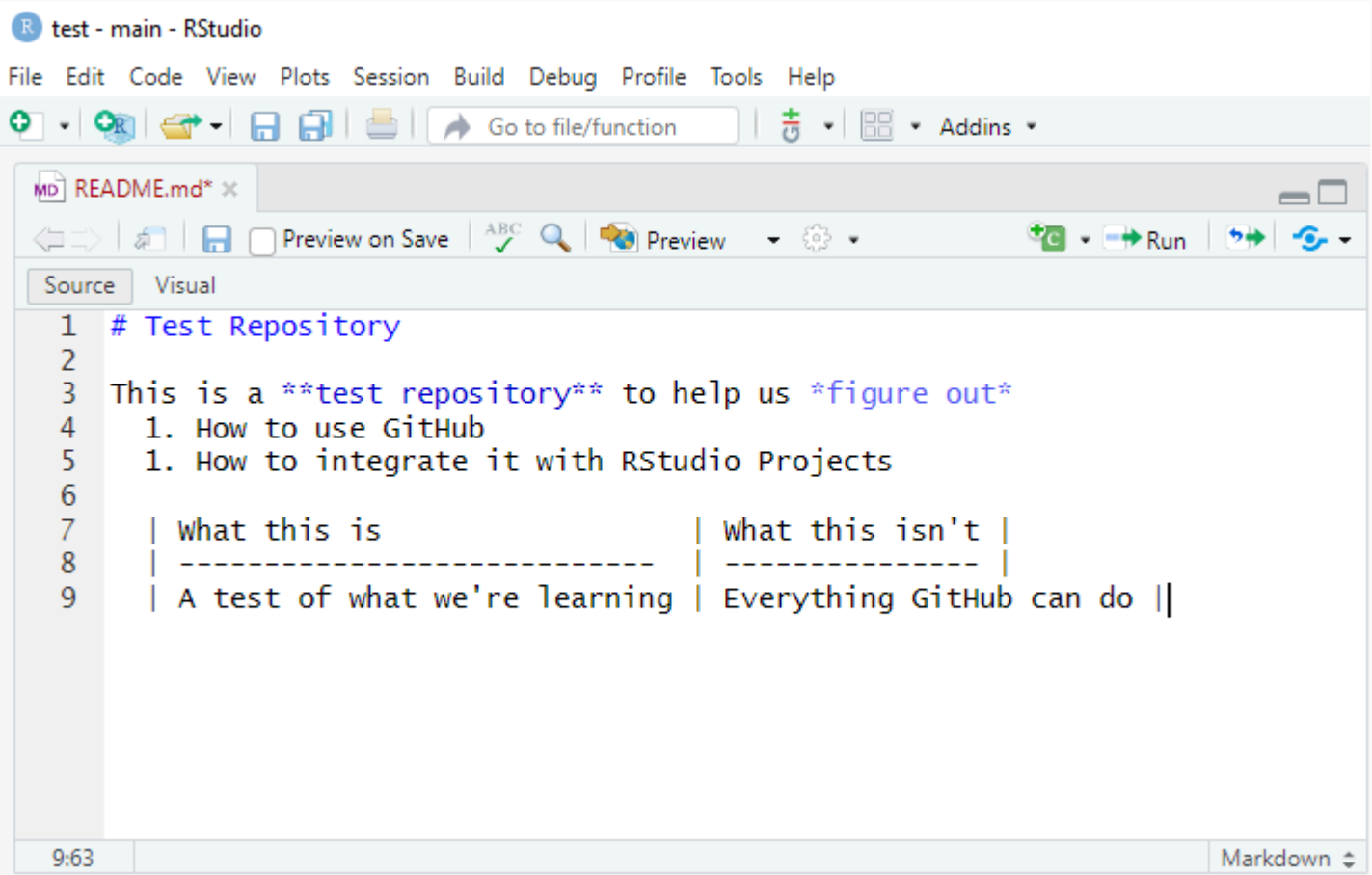
- Add some text and save the README.
- Do you see any changes in the **Git** panel? Good. (Raise your hand if not.)

[1] They're important, but not for the purposes of this section.

Making Local Changes



Making Local Changes



Main Git operations

Now that you've **cloned** your first repo and made some local changes, it's time to learn the **four main Git operations**.

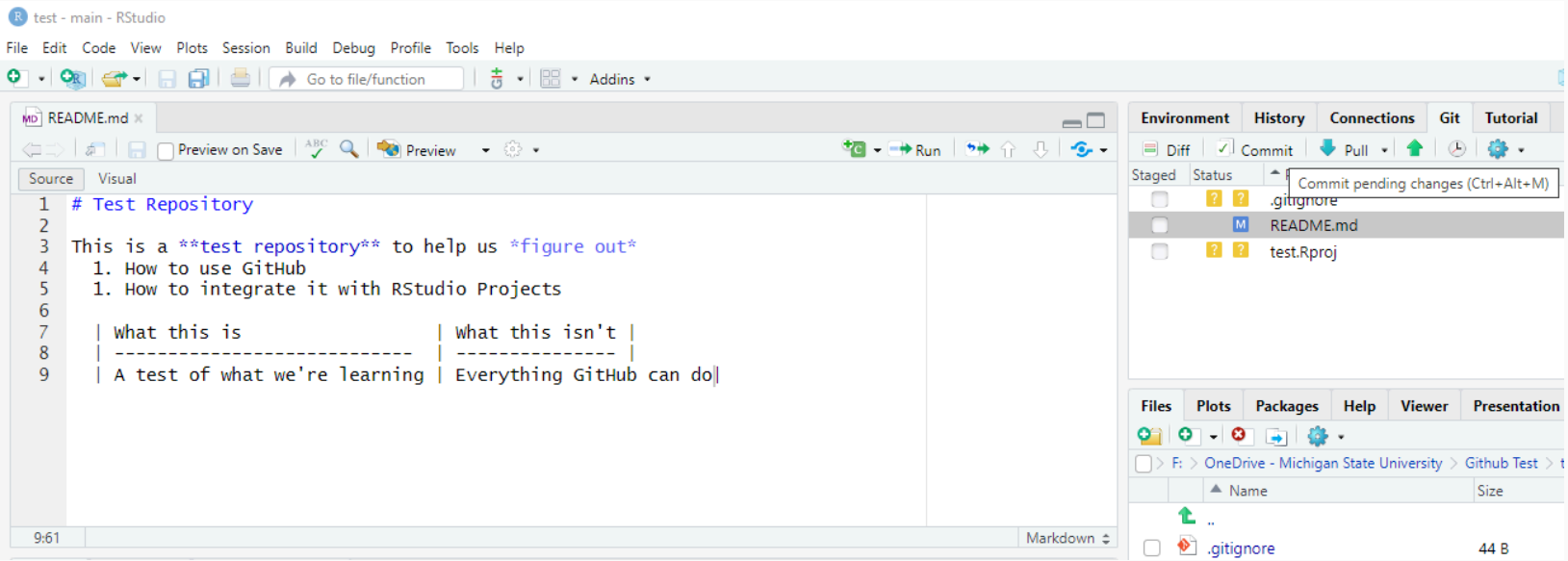
1. **Stage** (or "add")
   - Tell Git that you want to add changes to the repo history (file edits, additions, deletions, etc.)
2. **Commit**
   - Tell Git that, yes, you are sure these changes should be part of the repo history.
3. **Pull**
   - Get any new changes made on the GitHub repo (i.e. the upstream remote), either by your collaborators or you on another machine.
4. **Push**
   - Push any (committed) local changes to the GitHub repo

Main Git operations

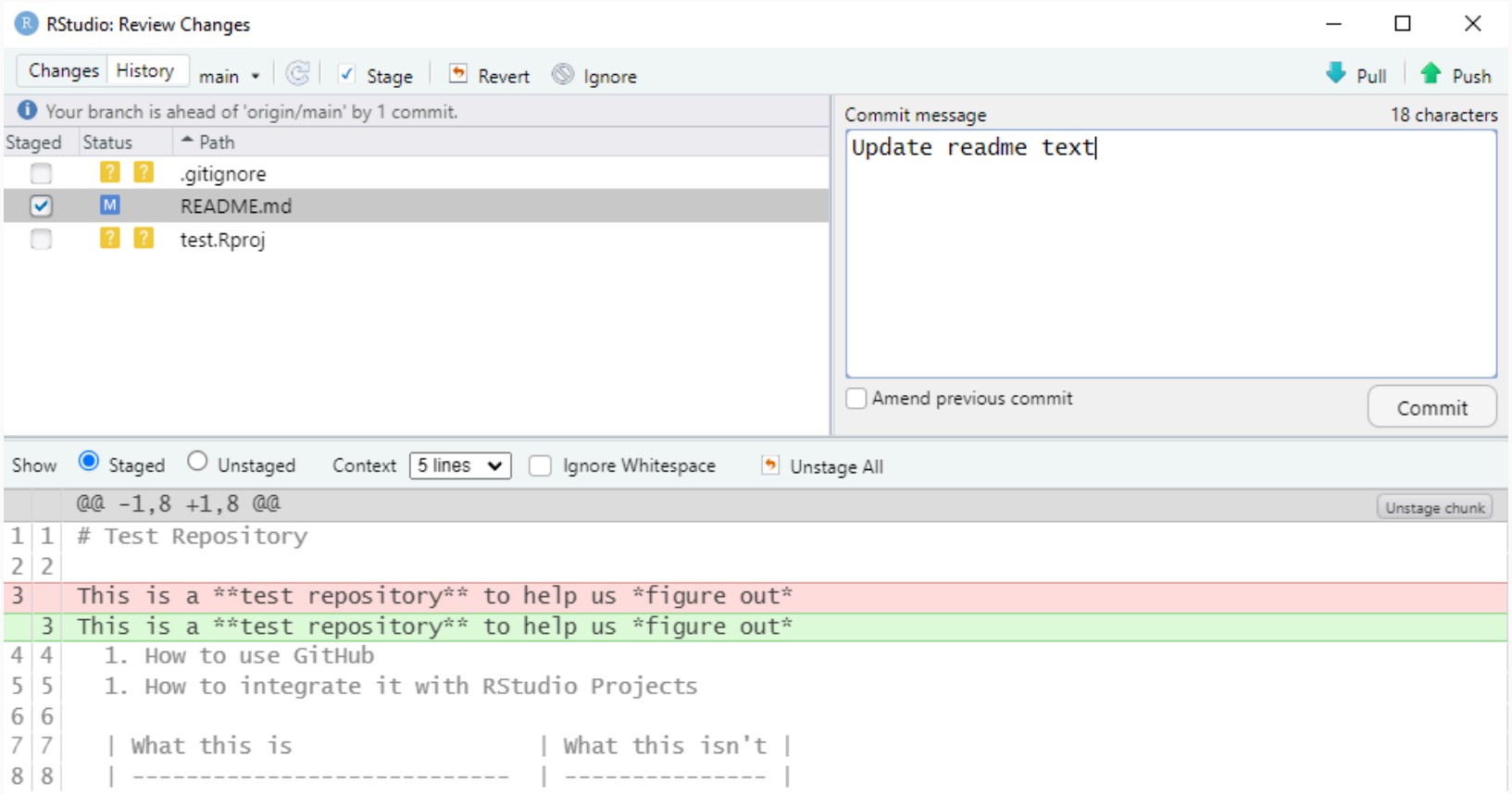
1. **Stage** (or "add")
   - Tell Git that you want to add changes to the repo history (file edits, additions, deletions, etc.)
2. **Commit**
   - Tell Git that, yes, you are sure these changes should be part of the repo history.
3. **Pull**
   - Get any new changes made on the GitHub repo (i.e. the upstream remote), either by your collaborators or you on another machine.
4. **Push**
   - Push any (committed) local changes to the GitHub repo

For the moment, it will be useful to group the first two operations and last two operations together. (They are often combined in practice too, although you'll soon get a sense of when and why they should be split up.)

Stage and Commit



Stage and Commit



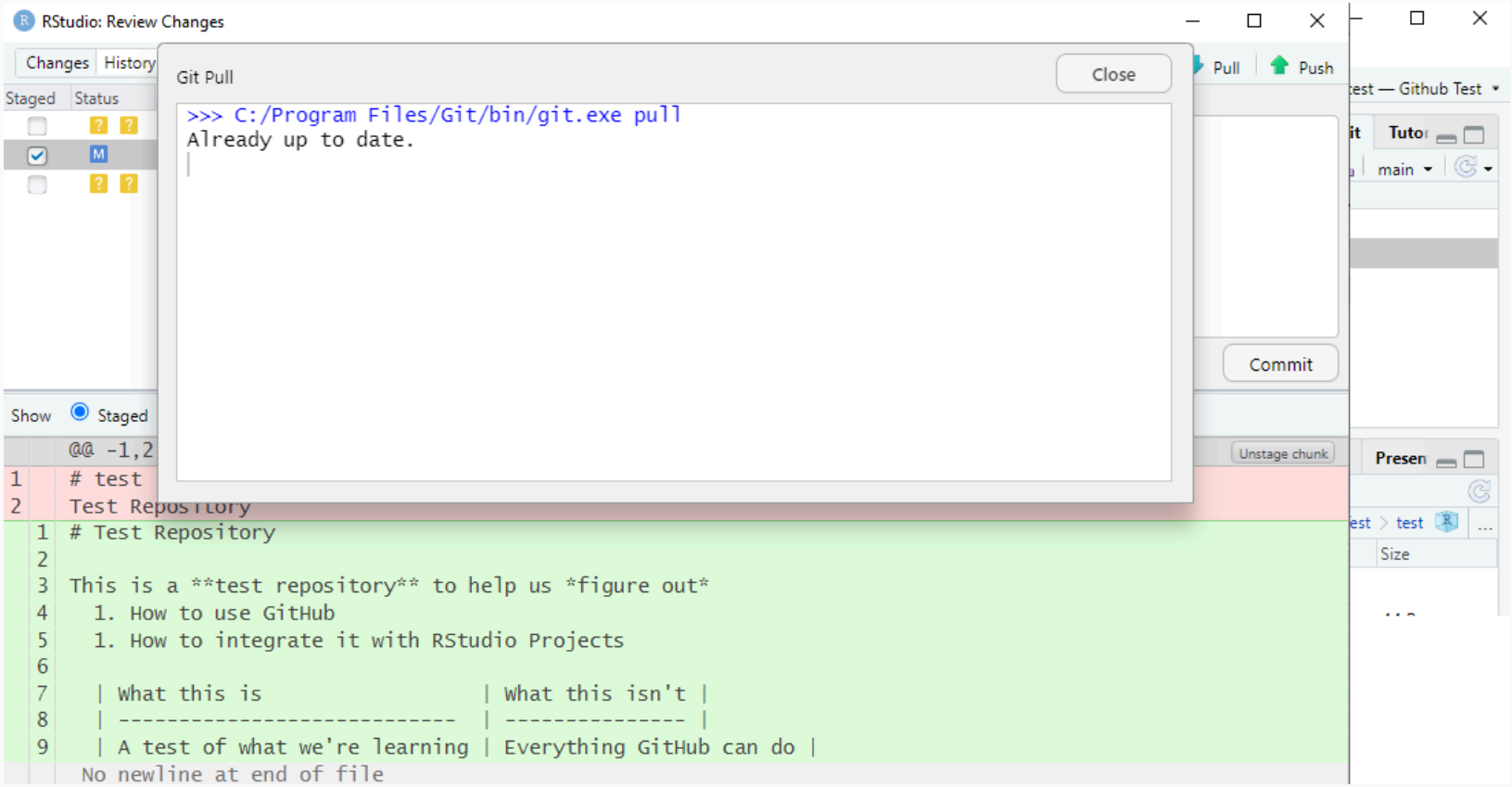
Stage and Commit

Git Commit

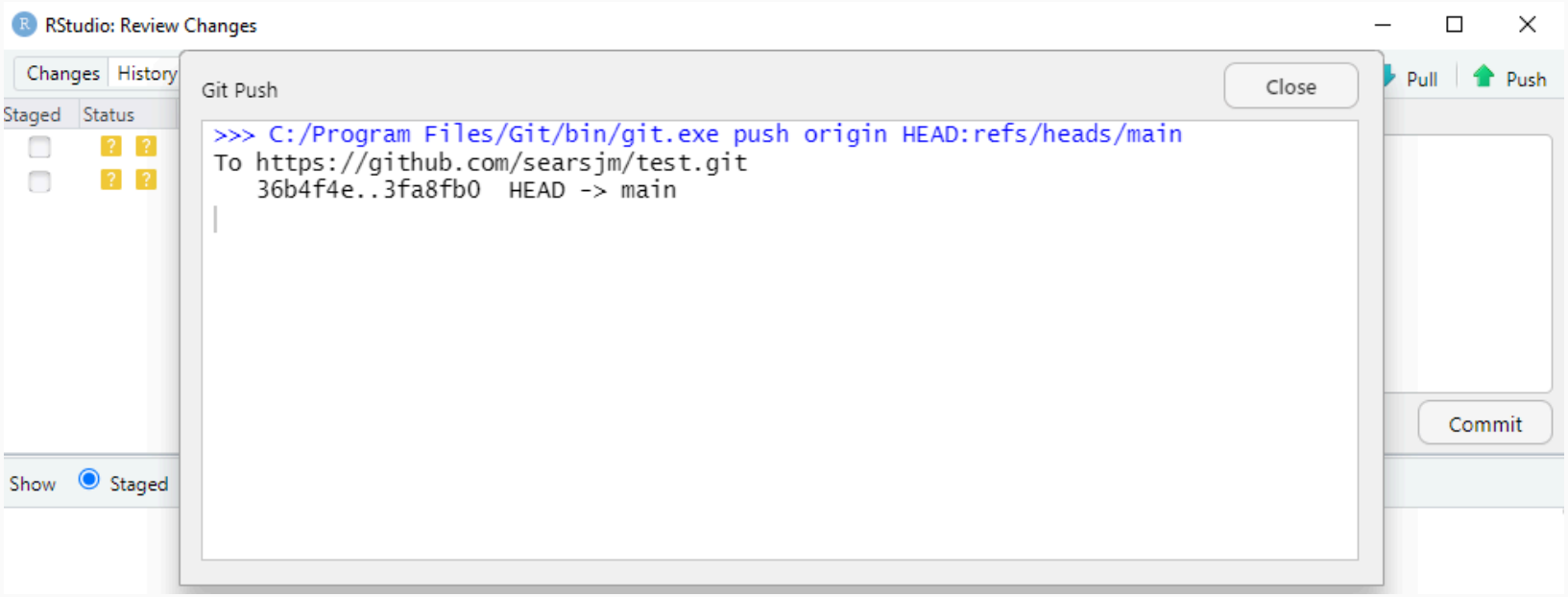
Close

```
>>> C:/Program Files/Git/bin/git.exe commit -F C:/Users/James/AppData/Local/Temp/
[main 50fe2dd] Update readme text
 1 file changed, 1 insertion(+), 1 deletion(-)
```

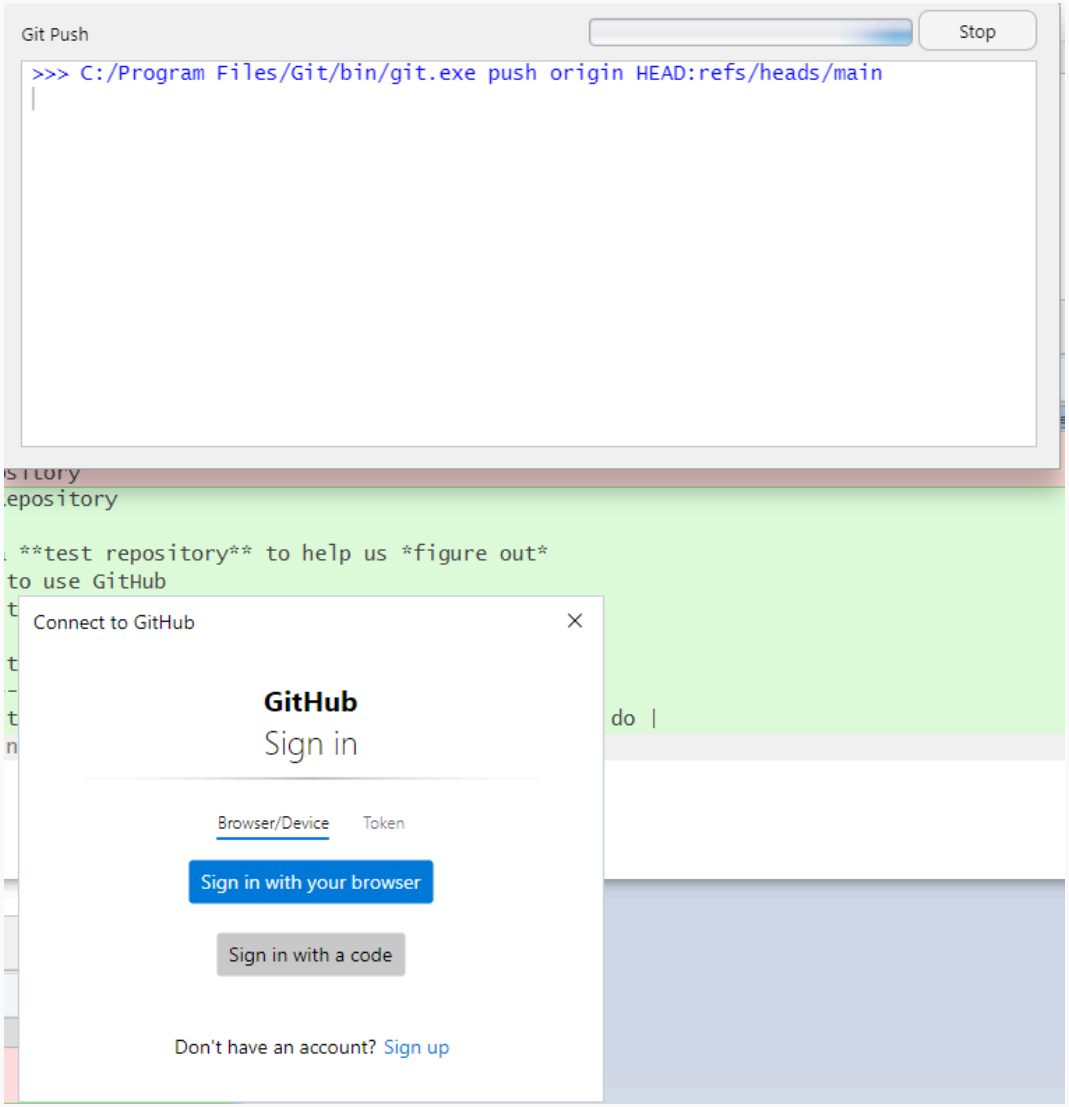
Pull


RStudio: Review Changes
Changes
History
Staged
Status
Git Pull
Close
>>> C:/Program Files/Git/bin/git.exe pull
Already up to date.
Pull
Push
Commit
Show
Staged
Unstage chunk
@@ -1,2
# test
Test Repository
Test Repository
This is a **test repository** to help us *figure out*
1. How to use GitHub
1. How to integrate it with RStudio Projects
What this is	What this isn't
A test of what we're learning	Everything GitHub can do
No newline at end of file
main
Size

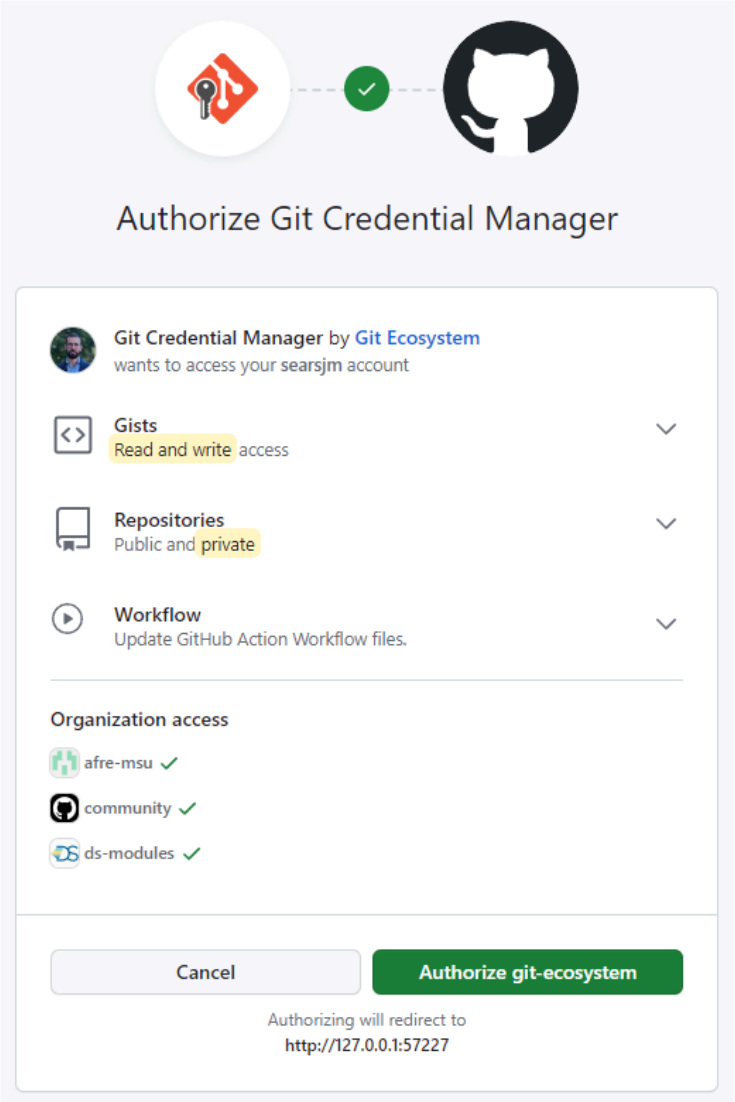
Push


RStudio: Review Changes
Changes
History
Staged
Status
Pull
Push
Commit
Show
Staged
Git Push
Close
>>> C:/Program Files/Git/bin/git.exe push origin HEAD:refs/heads/main
To https://github.com/searsjm/test.git
36b4f4e..3fa8fb0 HEAD -> main

Sign RStudio into Github



Sign RStudio into Github



Changes Now Visible on Github

searsjm / test

Type / to search

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

test Public

Pin | Unwatch 1 | Fork 0 | Star 0

main | 1 Branch | 0 Tags | Go to file t | + | Code

searsjm Update Readme 52525f2 · now 3 Commits

LICENSE	Initial commit	42 minutes ago
README.md	Update Readme	now

README | MIT license

Test Repository

This is a **test repository** to help us *figure out*

1. How to use GitHub
2. How to integrate it with RStudio Projects

What this is	What this isn't
A test of what we're learning	Everything GitHub can do

About

Test Repository

Readme
MIT license
Activity
0 stars
1 watching
0 forks

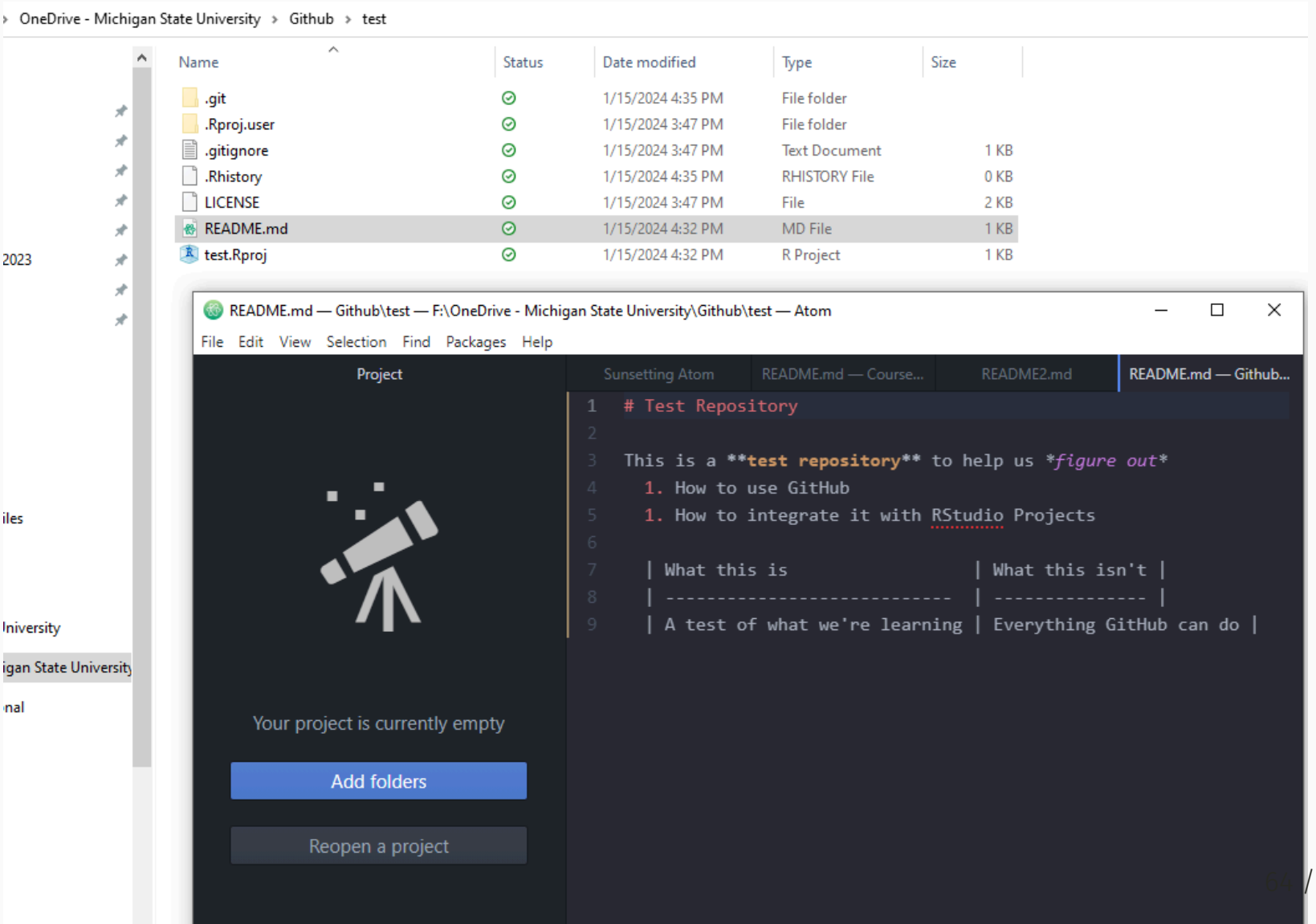
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Changes Now Visible Locally Too



Recap

Here's a step-by-step summary of what we just did.

- Made same changes to a file and saved them locally.
- **Staged** these local changes.
- **Committed** these local changes to our Git history with a helpful message.
- **Pulled** from the GitHub repo just in case anyone else made changes too (not expected here, but good practice).
- **Pushed** our changes to the GitHub repo.

Aside: Always pull from the upstream repo *before* you push any changes. Seriously, do this even on solo projects; making it a habit will save you headaches down the road.

Recap

Here's a step-by-step summary of what we just did.

- Made same changes to a file and saved them locally.
- **Staged** these local changes.
- **Committed** these local changes to our Git history with a helpful message.
- **Pulled** from the GitHub repo just in case anyone else made changes too (not expected here, but good practice).
- **Pushed** our changes to the GitHub repo.

PS — You were likely challenged for your GitHub credentials at some point. Learn how to cache these **here**.

Why this Workflow: GitHub

Creating the repo on GitHub first means that it will **always be "upstream"** of your (and any other) local copies.

- In effect, this allows GitHub to act as the **central node** in the distributed VC network.
- Especially valuable when you are collaborating on a project with others — more on this later — but also has advantages when you are working alone.
- If you would like to move an existing project to GitHub, my advice is still to create an empty repo there first, clone it locally, and then copy all your files across.

Why this Workflow: RStudio

RStudio Projects are great.

- They interact seamlessly with Git(Hub), as we've just seen.
- They also solve absolute vs. relative path problems, since the .Rproj file acts as an anchor point for all other files in the repo.[1]

[1] Calling files from their full `YourComputer/YourName/Documents/Special-Subfolder/etc` paths in your scripts is the enemy of reproducibility!

GitHub Desktop

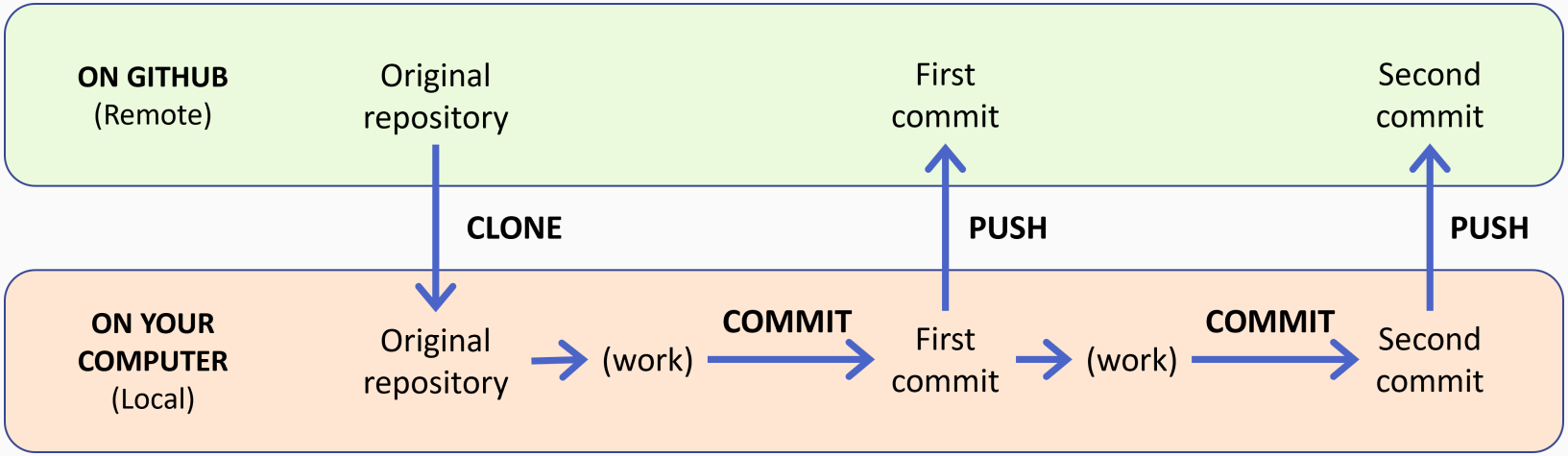
Version Control with GitHub Desktop

Although GitHub integration with RStudio has lots of functionality, there are times where we want to keep track of files and projects **outside of RStudio**.

This is where **GitHub Desktop** comes in.

Github Desktop Workflow

With GitHub Desktop, we can maintain a similar workflow

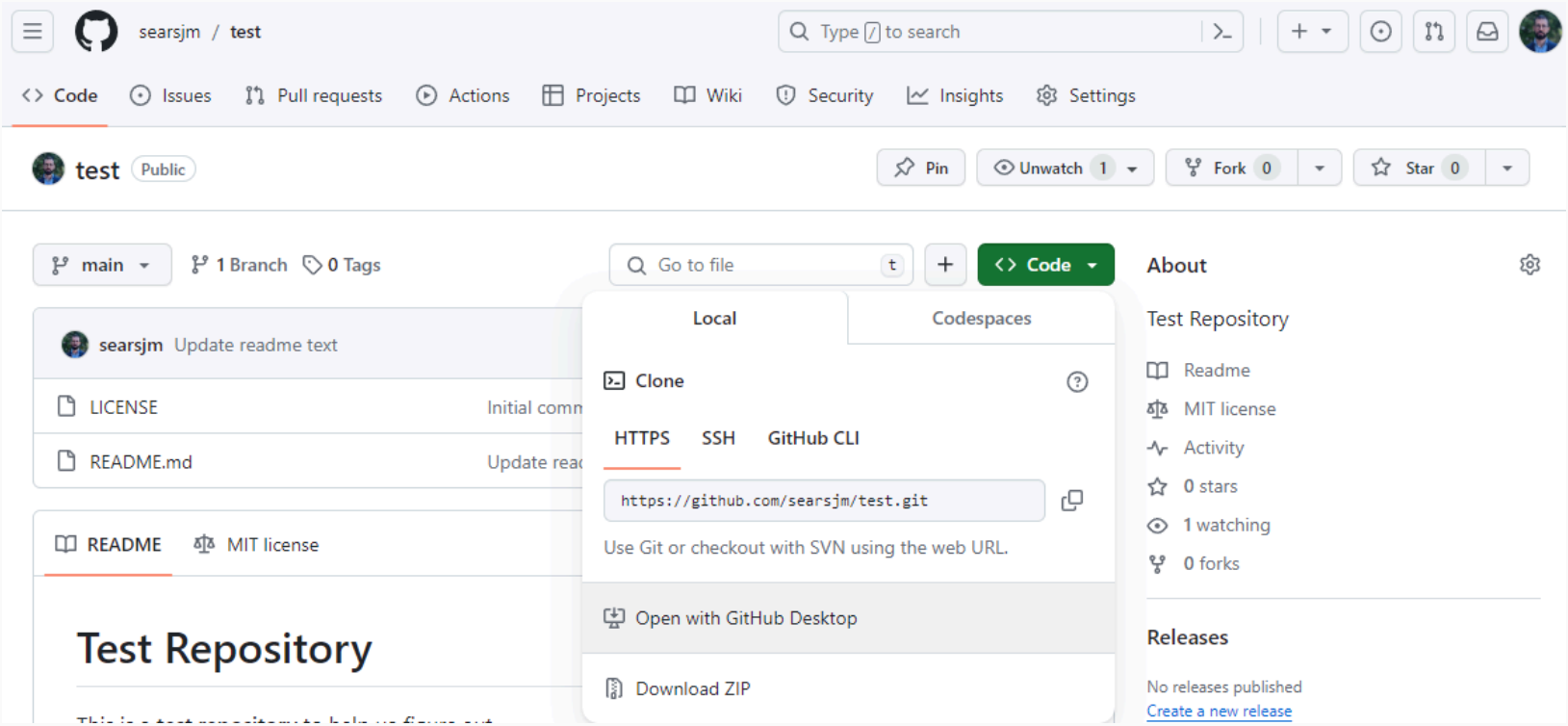


Github Desktop Workflow

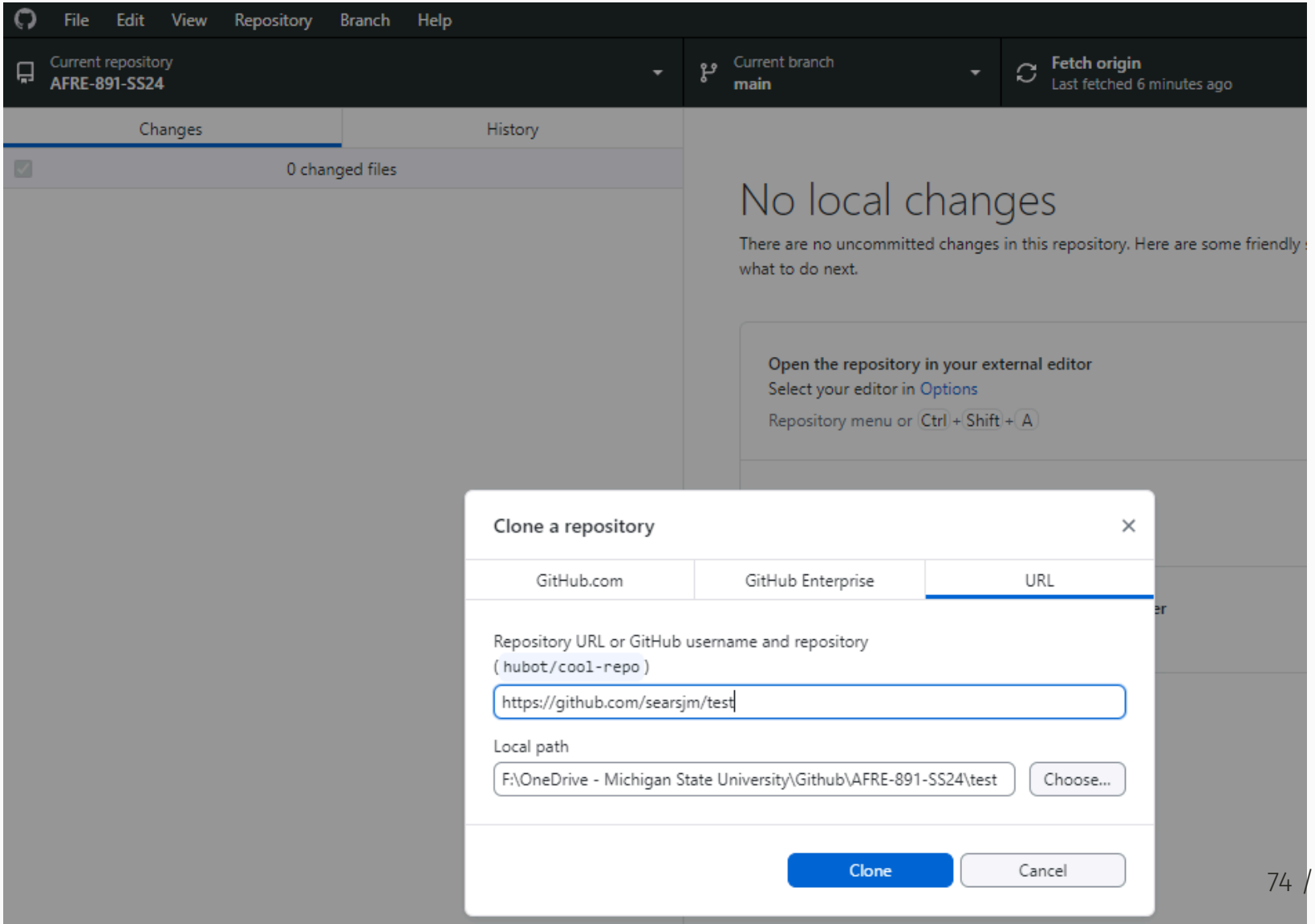
With GitHub Desktop, we can maintain a similar workflow

1. Create a new repository on GitHub.com
2. **Clone** the repository to your local machine
3. Do some work (i.e. edit the repository)
4. **Commit** changes to the repository
5. **Push** your commit to GitHub

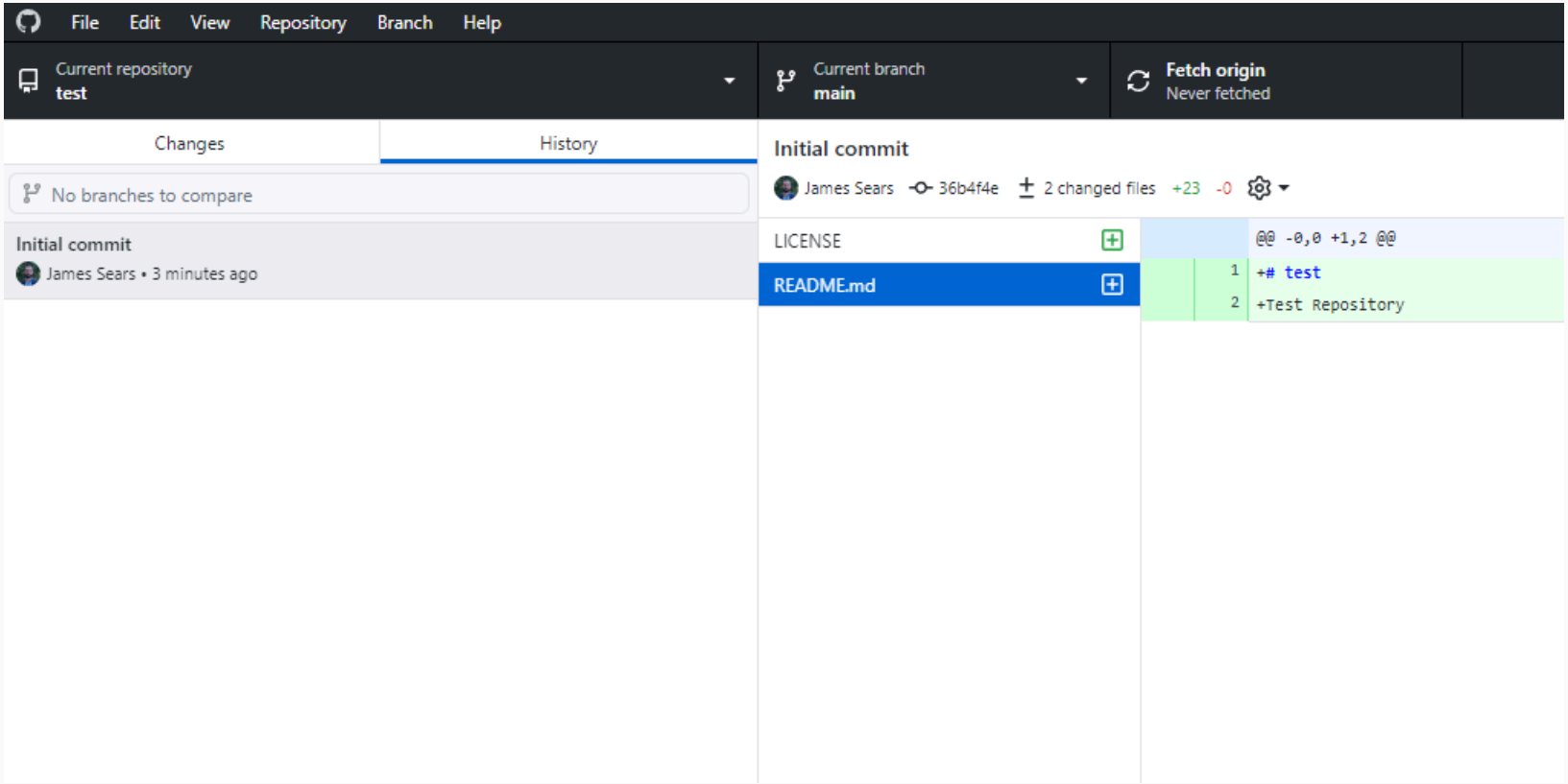
Clone Repository



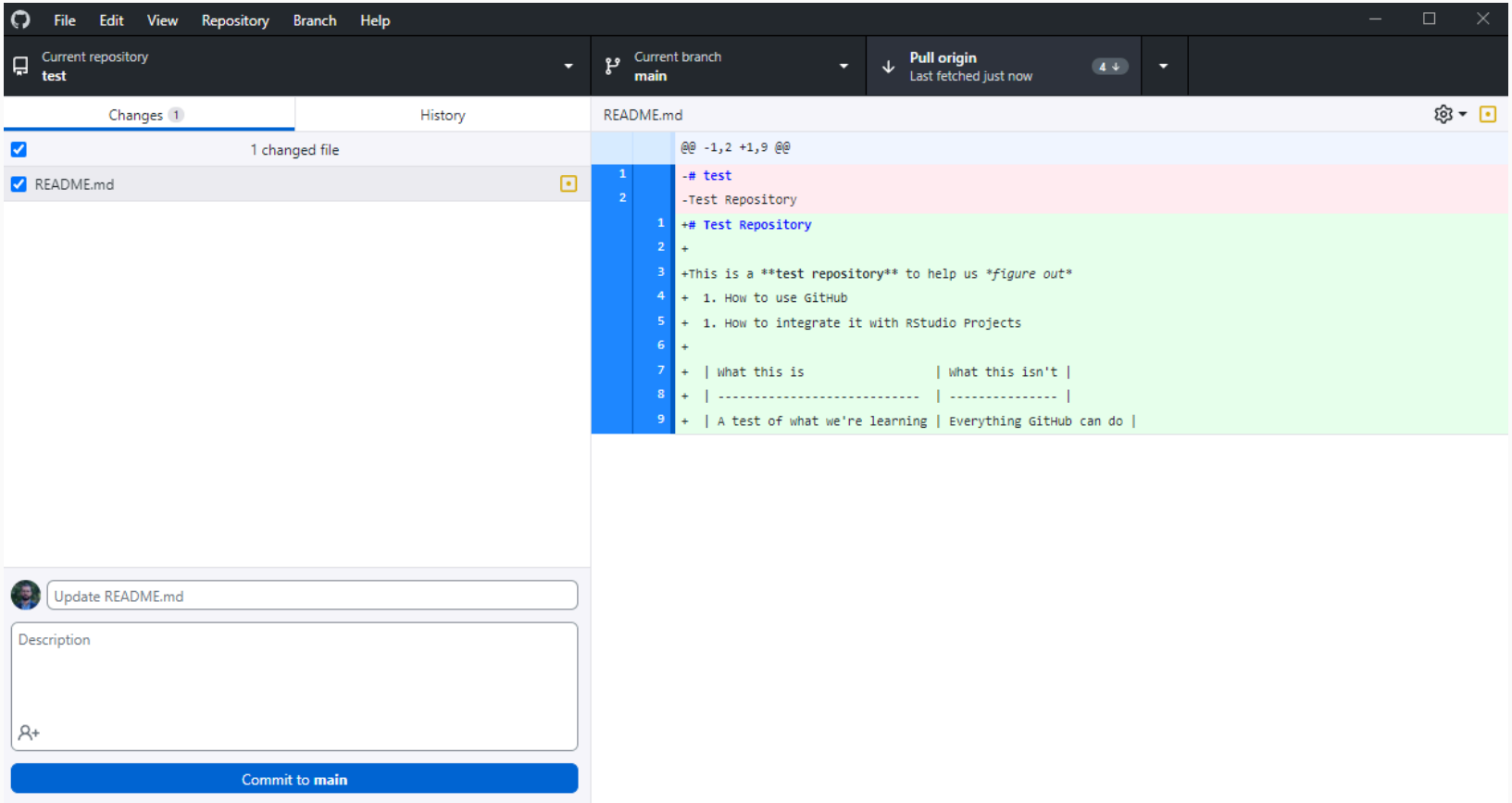
Clone Repository



Clone Repository



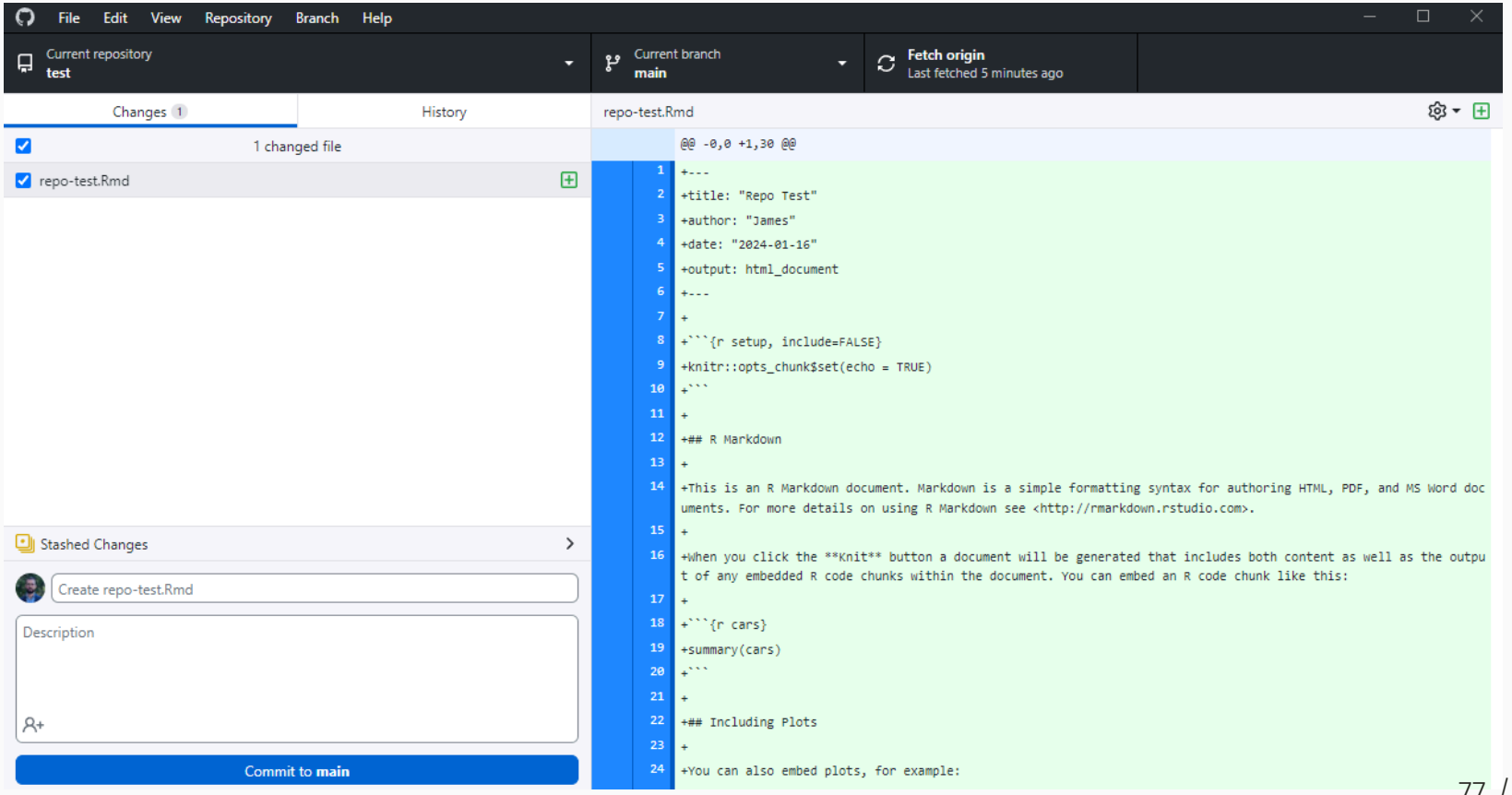
Fetch Origin + Pull to Stay Current



Make Local Changes to Repo

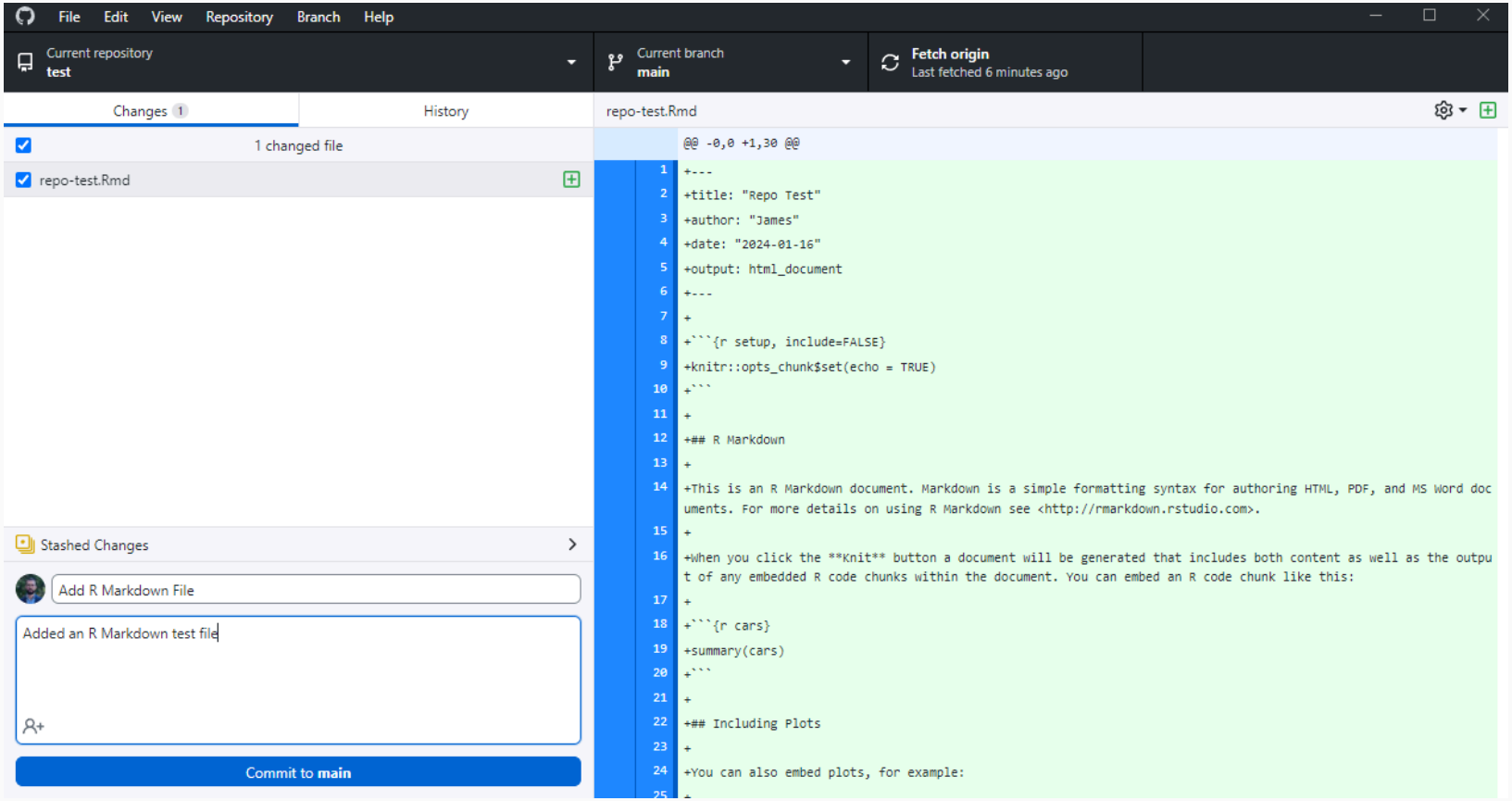
Save a new .Rmd file named `repo-test` into the "test" folder

- GitHub Desktop automatically stages the changes

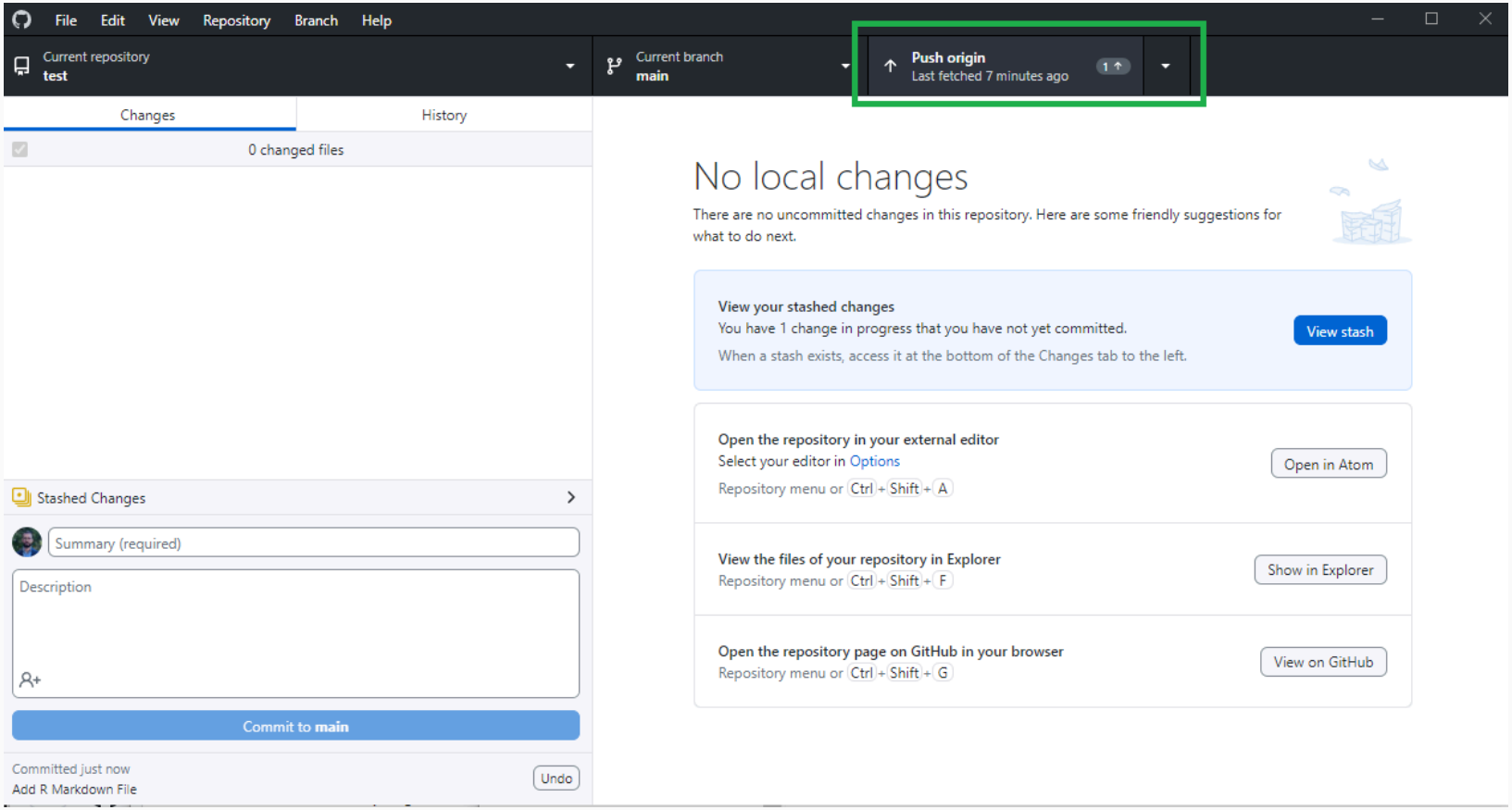


Commit Changes

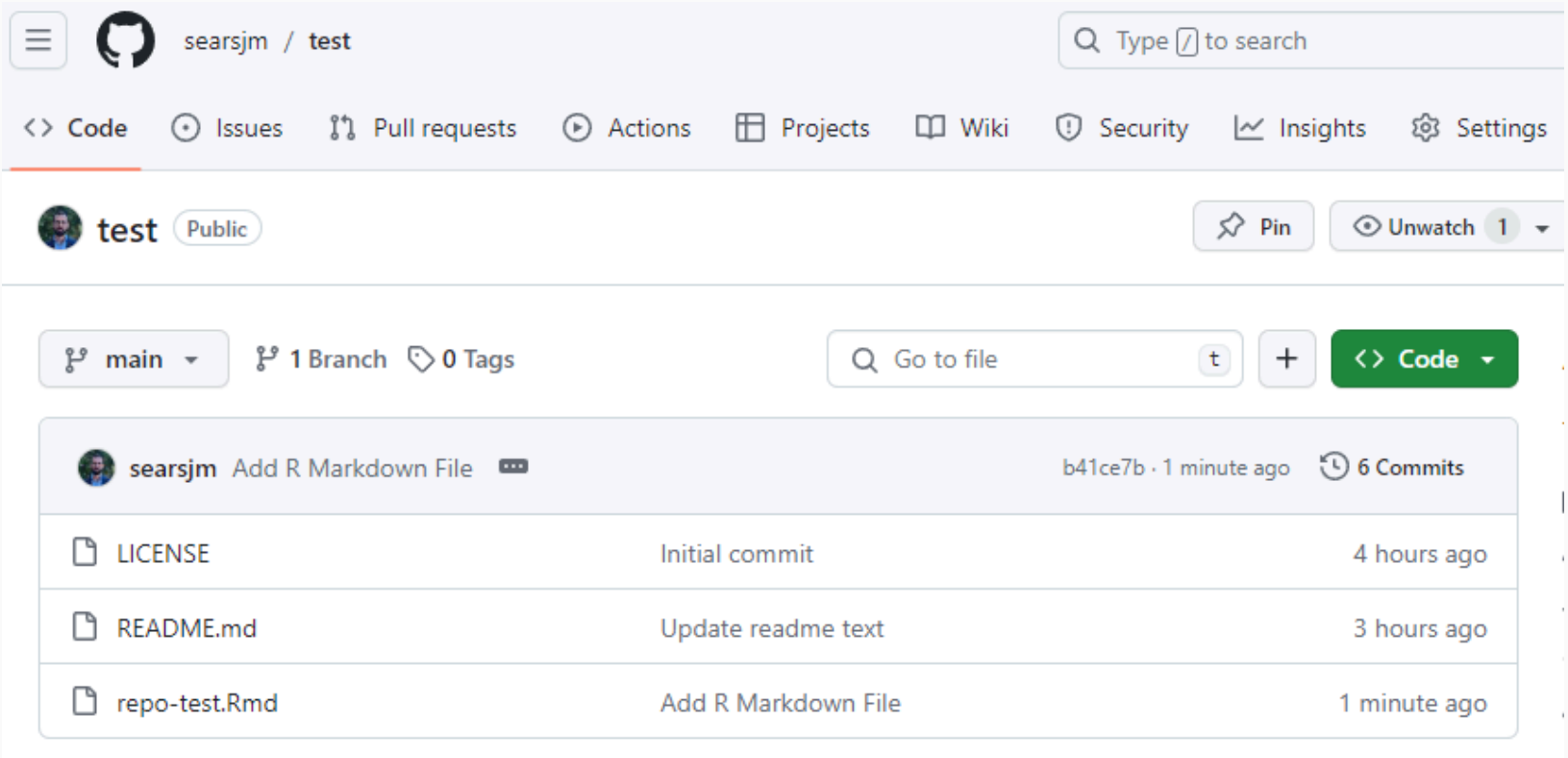
Add commit title and description, then commit to GitHub



Push Changes to GitHub Desktop



Repo now Matches Local Version



Other Tips and Productivity Tools

Productivity Miscellanea

What follows are miscellaneous things that I find **improve my productivity**

- **Synced Cloud Storage** (SpartanDrive or Dropbox/Box)
- **Overleaf** for LaTeX collaboration
- **Connected Papers** for literature networks

Synced Cloud Storage and OneDrive

Synced Cloud Storage is hugely beneficial if you work across **multiple computers** or **with many collaborators**.

- Make sure each computer has the most up-to-date version of all your files
- Renders flash drives almost entirely obselete!

Synced Cloud Storage and OneDrive

One easy way to do this: **SpartanDrive/OneDrive**

All faculty + staff get **5 TB of free storage** on **SpartanDrive (MSU's version of OneDrive)**

Pros

- Free
- Syncable desktop apps
- Part of the MSU Office365 ecosystem

Cons

- Part of the MSU Office365 ecosystem
- Limited storage (5TB, 250gb max filesize)
- Sometimes finicky

Synced Cloud Storage

Alternatives to SpartanDrive:

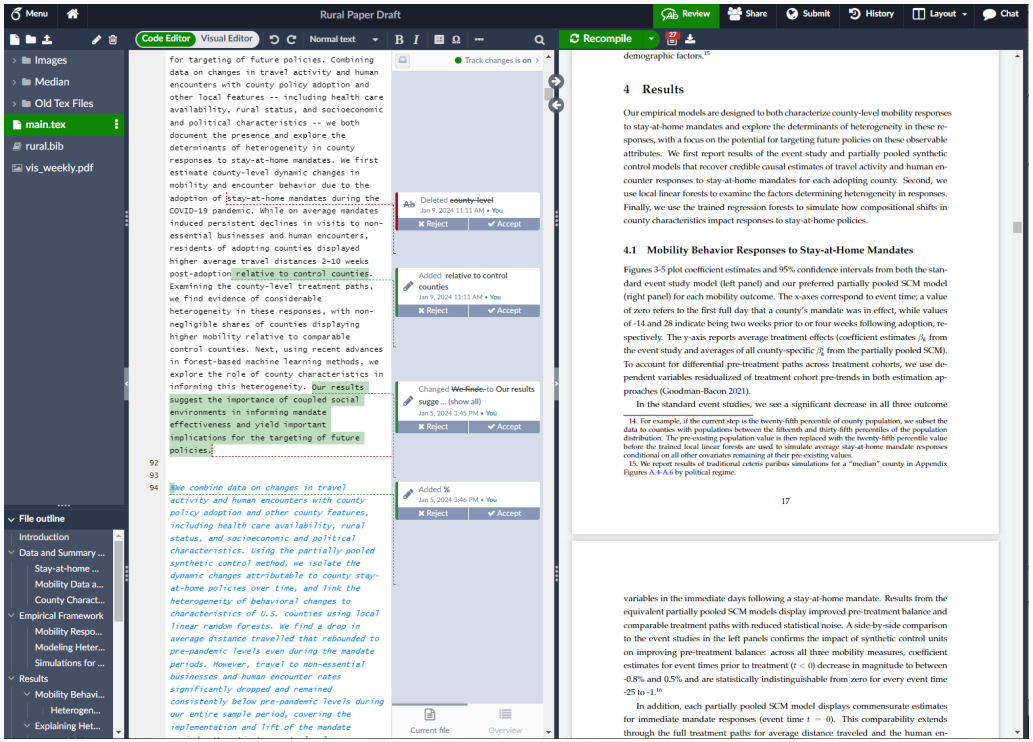
- **Dropbox**: 2GB free ($10/mo for 2TB)
- **Box:** 10GB free ($10/mo for 100GB)

Ultimate choice of platform may depend on coauthors + your current choice, but it's a good idea to **download the desktop app** to keep your files synced + backed up!

- Also gives basic version control

Overleaf

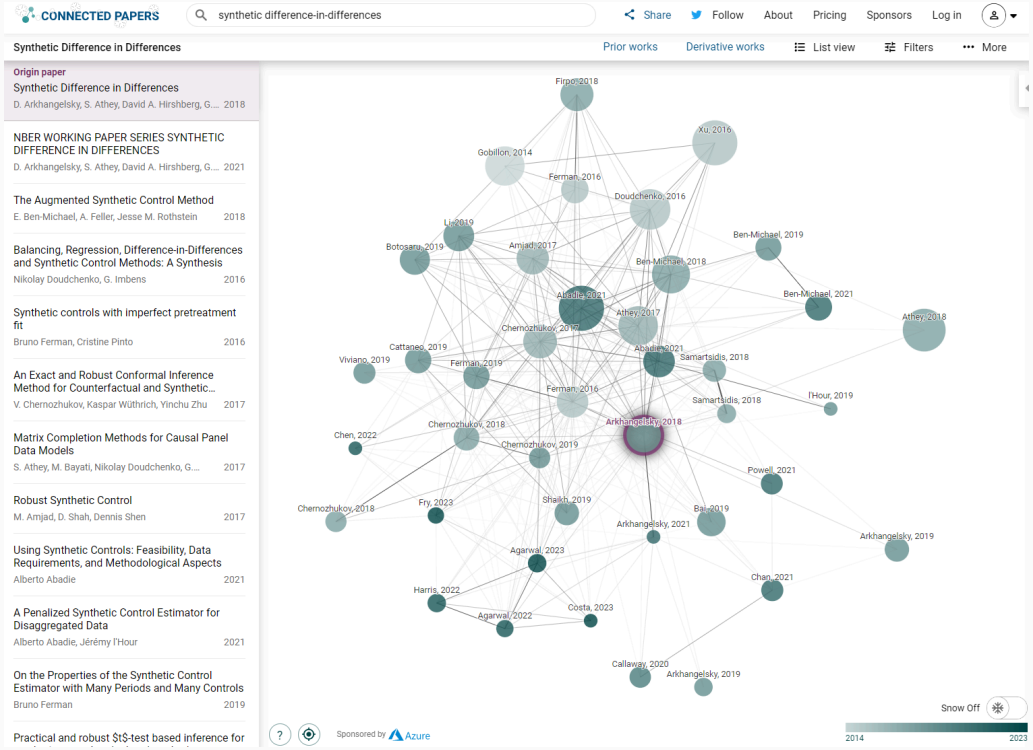
If you typeset using LaTeX, **Overleaf** streamlines access and collaboration



Free Version: remotely host all files, access for you + 1 collaborator

Paid ($7.40/mo): track changes + full document history, Git(hub) + Dropbox integration, up to 6 collaborators

Connected Papers



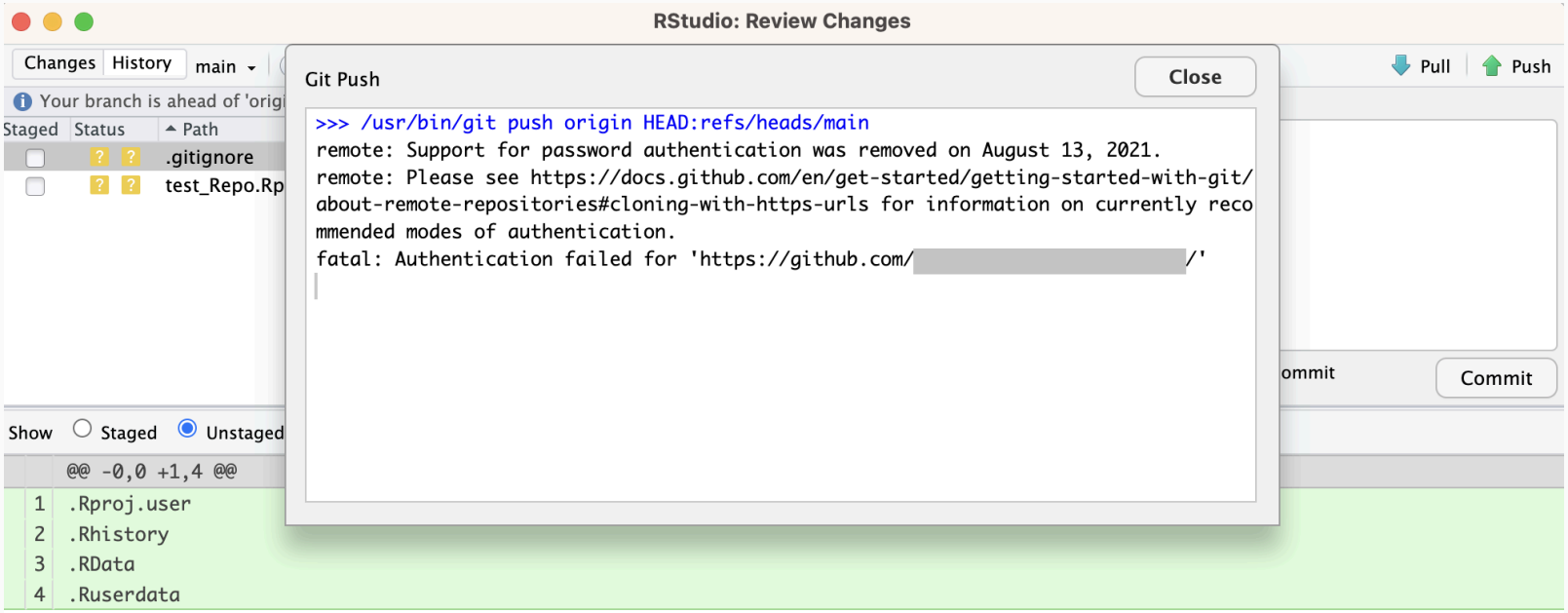
- Graphical representation of paper networks
  - Visualize literature as directed graph
  - 5 free per month (**unlimited for $6/mo**)

Your Productivity Tips + Tricks?

Troubleshooting Git Credential Issues in RStudio

Troubleshooting Git Credential Issues

Do you get a password authentication error when trying to push to GitHub from RStudio?



Troubleshooting Steps

To begin, install the `usethis` package:

```
pacman:p_load(usethis)

# or:
# install.packages("usethis")
# followed by
# library(usethis)
```

Troubleshooting Steps

Next, run the following:

```
library("usethis")
git_sitrep()
```

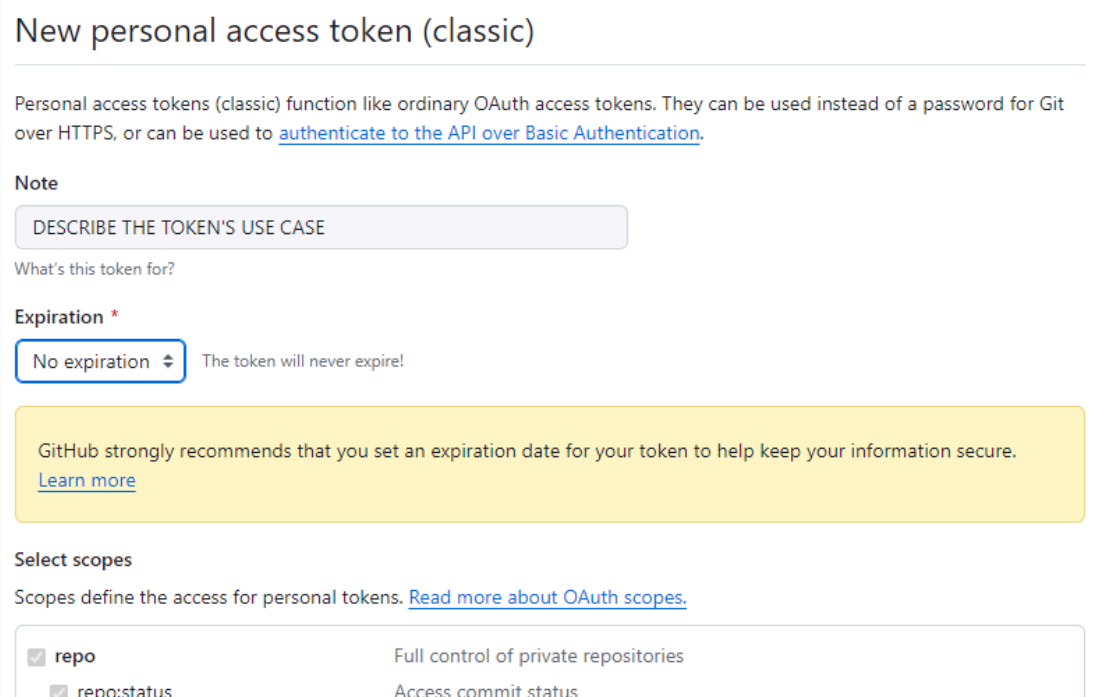
View the output: do you see any Xs or lines saying "lacks recommended scopes", "error", or "can't retrieve"?

Create a Personal Access Token (PAT)

To fix these errors, we'll create a **Personal Access Token (PAT)** on GitHub.

The below code will open a browser window - follow the steps to create a PAT

```
usethis::create_github_token()
```



Add a PAT to RStudio

Back in RStudio, run

```
gitcreds::gitcreds_set()
```

Follow the prompts to add/replace existing credentials, pasting in the PAT.

Check PAT

Once added, run `git_sitrep` one last time to verify.

If all looks good, try pushing again!

```
> git_sitrep()
Git config (global)
• Name: 'James Sears'
• Email: 'james.sears.m@gmail.com'
• Global (user-level) gitignore file: <unset>
• Vaccinated: FALSE
i See `?git_vaccinate` to learn more
• Default Git protocol: 'https'
• Default initial branch name: 'master'
GitHub
• Default GitHub host: 'https://github.com'
• Personal access token for 'https://github.com': '<discovered>'
• GitHub user: 'searsjm'
• Token scopes: 'gist, repo, user, workflow'
• Email(s): 'james.sears.m@gmail.com (primary)', 'searsja1@msu.edu'
Git repo for current project
i No active usethis project
>
```

Table of Contents