

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

James Sears\*

AFRE 891/991 FS 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

1. [Prologue](#)
2. [R Markdown](#)
3. [Version Control](#)
4. [GitHub Desktop](#)
5. [Other Tips and Productivity Tools](#)
6. [Not Covered: Git\(Hub\) + RStudio](#)
7. [Not Covered: Troubleshooting Git Credential Issues in RStudio](#)

# 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](https://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

Installed [R](#).

Installed [RStudio](#).

Add the `rmarkdown` package

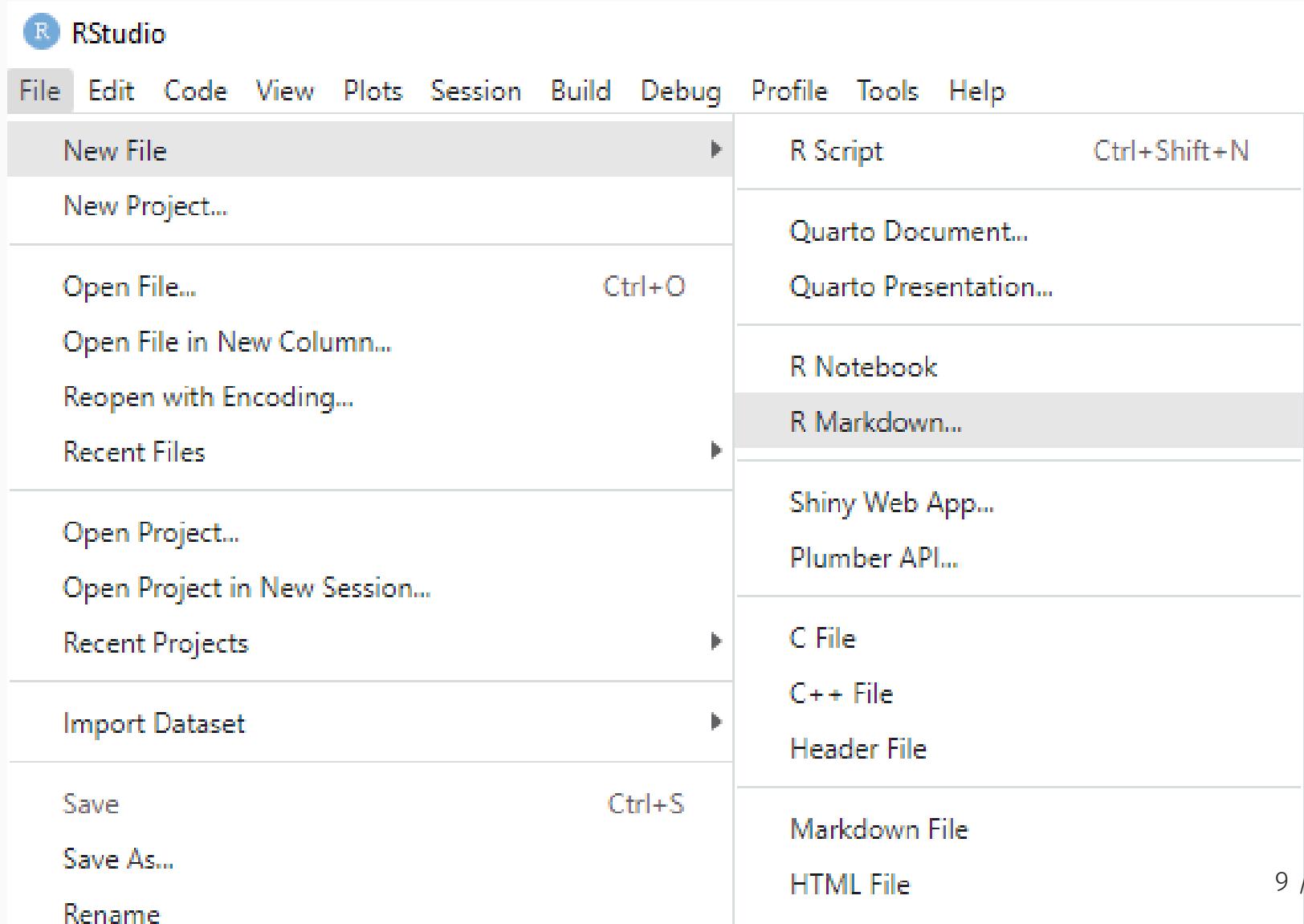
```
install.packages("rmarkdown")
```

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

New R Markdown

Document

Presentation

Shiny

From Template

**Title:** Test R Markdown

**Author:** James

**Date:** 2024-01-10

Use current date when rendering document

**Default Output Format:**

HTML  
Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF  
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word  
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

# R Markdown: Creating a New .Rmd File

The screenshot shows the RStudio IDE interface. The title bar says "RStudio". The menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The toolbar has various icons for file operations like Open, Save, and Print. A search bar says "Go to file/function". A dropdown menu says "Addins". The tab bar shows several files: "02-Productivity.Rmd", "Untitled1", "R-Markdown-example.Rmd", "02-Git.Rmd", and "01-R-Intro.Rmd". Below the tabs are buttons for "Knit on Save", "ABC", "Knit", "Run", and "Outline". The main area is a code editor with the following content:

```
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  
21
```

# 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 heading with 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<sup>2</sup> 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: Text Format

Draw **tables** using | and -

Col A	Col B	Col C
This	is	a
Table		wow

# Markdown: Text Format

You can adjust the **alignment** of table text by adding `:`'s in the second row:

- `:----` for left-aligned
- `:-:-:` for center-aligned
- `---:--:` for right-aligned

	Column A	Column B	Column C
	<code>:-:-:</code>	<code>:-:-:</code>	<code>---:--:</code>
	Col A	is	left-aligned
	Col B	is	center-aligned
	Col C	is	right-aligned

# Markdown: Text Format

You can adjust the **alignment** of table text by adding `:`'s in the second row:

- `:----` for left-aligned
- `:---:` for center-aligned
- `----:` for right-aligned

<b>Column A</b>	<b>Column B</b>	<b>Column C</b>
Col A	is	left-aligned
Col B	is	center-aligned
Col C	is	right-aligned

# 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`:



# Markdown: LaTeX

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

Add an **inline equation** with `$TeX$`

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

Add multiple rows of LaTeX with

`$$`

LaTeX lines here

`$$`

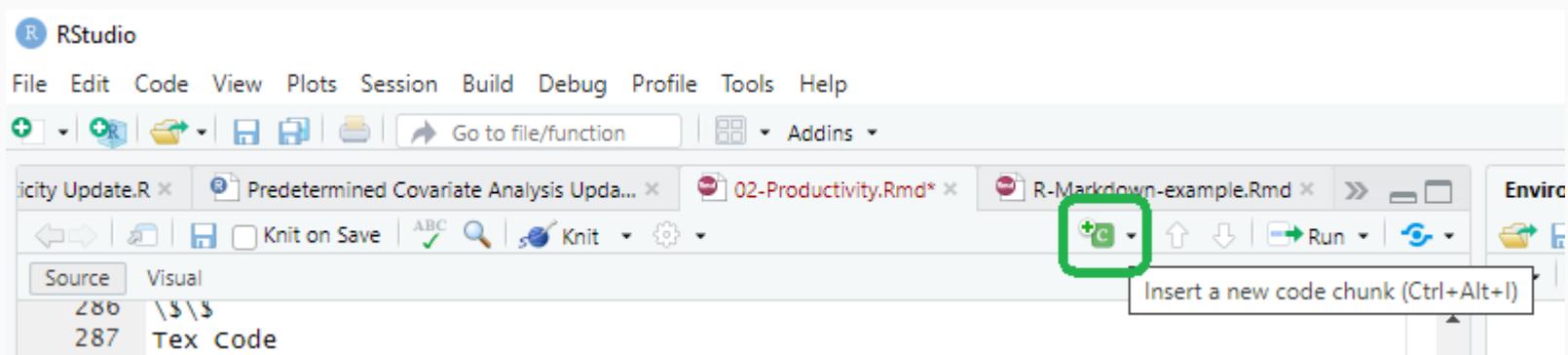
Use the **standard LaTeX commands** for symbols/characters

# Rmd: R Code

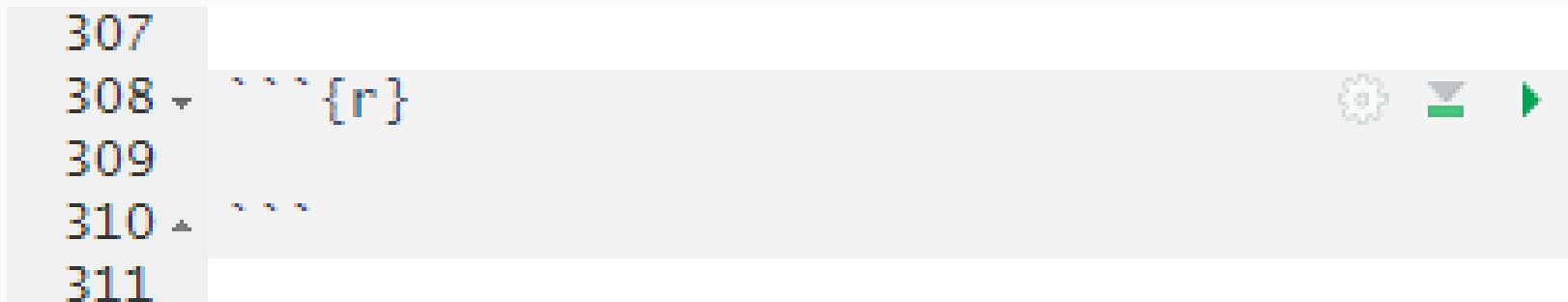
R code is primarily executed with **code chunks**

Add a chunk with

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



# Rmd: Code Chunks



A screenshot of the RStudio code editor interface. The code area shows several lines of R code:

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

The line numbers are in blue. The code block starts with a call to the `r` chunk. To the right of the code area are three icons: a gear (settings), a downward arrow (refresh), and a green arrow (run).

**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

# 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 - ````
```

# Code Chunk Options

You can also **change the color** of code chunks and/or output in the rendered document through code chunk options<sup>1</sup>

- `class.source` to change the **code chunk**
- `output.source` to change the **output**

**default**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

**bg-primary**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

<sup>1</sup> We'll chat more about this when we get to web scraping, but what's actually happening here is that we're harnessing some built-in CSS classes to change the backgrounds. This also means that, using CSS, you can define custom classes and format things however you'd like.

# Code Chunk Options

You can also **change the color** of code chunks and/or output in the rendered document through code chunk options<sup>1</sup>

- `class.source` to change the **code chunk**
- `output.source` to change the **output**

**bg-success**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

**bg-info**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

<sup>1</sup> We'll chat more about this when we get to web scraping, but what's actually happening here is that we're harnessing some built-in CSS classes to change the backgrounds. This also means that, using CSS, you can define custom classes and format things however you'd like.

# Code Chunk Options

You can also **change the color** of code chunks and/or output in the rendered document through code chunk options<sup>1</sup>

- `class.source` to change the **code chunk**
- `output.source` to change the **output**

**bg-warning**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

**bg-danger**

```
1 + 2 * 3 ^ 4
```

```
## [1] 163
```

<sup>1</sup> We'll chat more about this when we get to web scraping, but what's actually happening here is that we're harnessing some built-in CSS classes to change the backgrounds. This also means that, using CSS, you can define custom classes and format things however you'd like.

# Rmd: Inline Code

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

```
- r -
```

```
four = 2+2
```

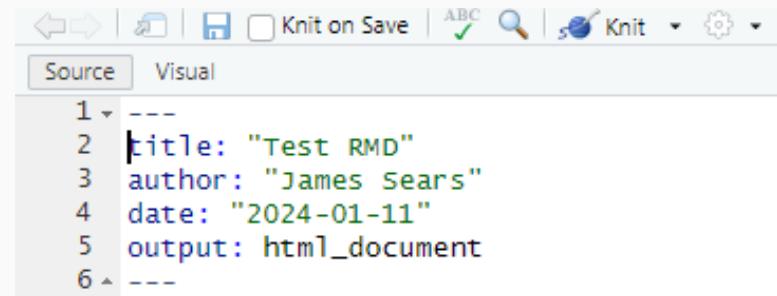
This can output in line with text:  $2 + 2 = 4$

# 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<sup>1</sup>: HTML  
(`html_document`), PDF  
(`pdf_document`), LaTeX  
(`latex_document`), or Word  
(`word_document`)<sup>2</sup>



A screenshot of the RStudio interface showing the Source tab. The code editor displays the following R Markdown header:

```
1 ---  
2 title: "Test RMD"  
3 author: "James Sears"  
4 date: "2024-01-11"  
5 output: html_document  
6 ---
```

1: See [CH 3 of "R Markdown: The Definitive Guide"](#) for more on how to customize [output formats](#)

2: For better formatted Word output with greater customisability, use the `officedown` package's [rdox\\_document](#) format.

## 2. R Setup

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

```
1
2
3
4
5
6
7
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

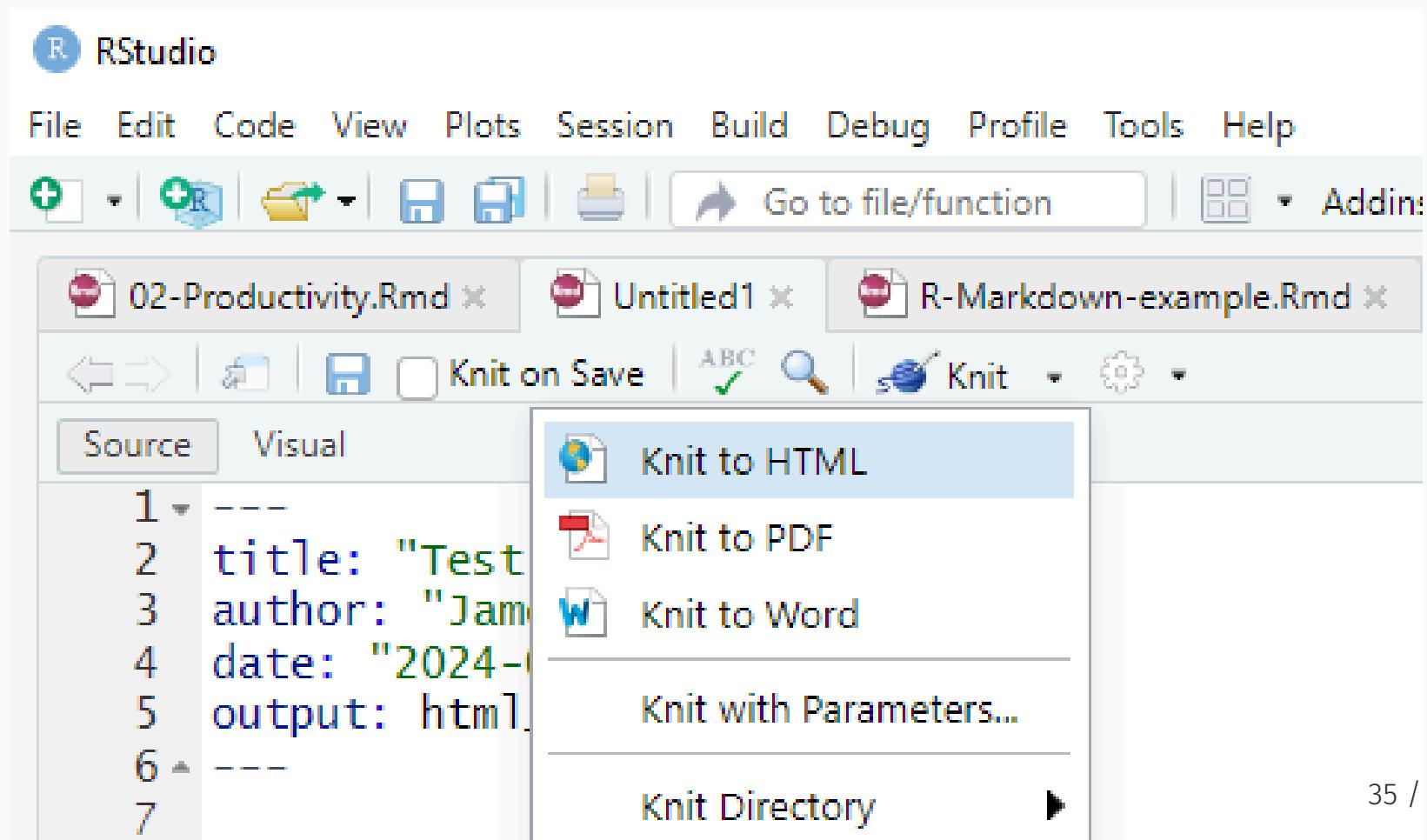
# 3. Contents

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

The screenshot shows a web browser window displaying an R Markdown document. The title bar reads "F:/OneDrive - Michigan State University/Teaching/MSU 2023-2024/AFRE 891 SS23/Lecture-Slides/02-Productivity/output/R-Markdown-example.html". The document content is as follows:

## 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

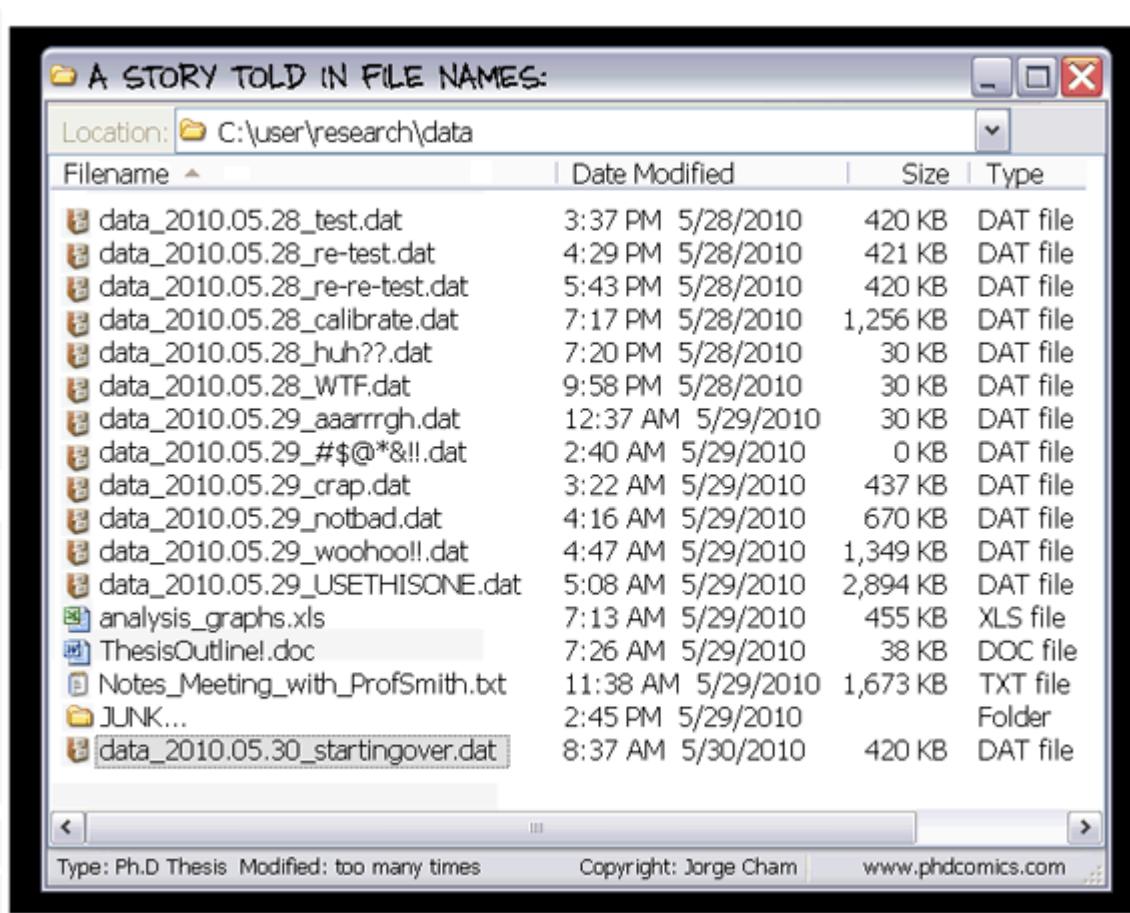
1. Create a new R Markdown file named "R-Markdown-Ex.Rmd"
2. In the setup chunk, load the **dslabs** and **tidyverse** packages
  - Use the `data()` function to read in the `divorce_margarine` dataset
3. Add a header labeled "Correlation vs. Causation" and a text explanation below for why we often want 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 favorite 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.<sup>2</sup>
- 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** for free.

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

# Using GitHub

There are a couple of different main ways that we could use GitHub:

## 1. Through [github.com](https://github.com) Only

- **Pros:** doesn't require any software/local repo copies
- **Cons:** much more time-intensive and not automated (the whole point of this thing!)

## 2. Through the command line ([GitHub CLI](https://cli.github.com))

- **Pros:** fully programmatic, requires no additional software
- **Cons:** fully programmatic!

# Using GitHub

There are a couple of different main ways that we could use GitHub:

## 3. Integrated with RStudio

- **Pros:** fully integrates RStudio projects
- **Cons:** limited to R projects, doesn't play as nicely with GitHub Classroom

## 4. Through the GitHub Desktop App

- **Pros:** Intuitive GUI, sync any kinds of files/projects
- **Cons:** requires an extra piece of software, [GitHub Desktop](#)

We're going to focus primarily on 4, but the back of the deck will contain slides working through 3 if you want to experiment with that too.

# 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**.

- For example, when you want version control of projects that **don't only use R** (or don't use it at all)

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

# Version Control with GitHub Desktop

This next section is about learning the basic Git(Hub) commands and the recipe for successful version control with GitHub Desktop.

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

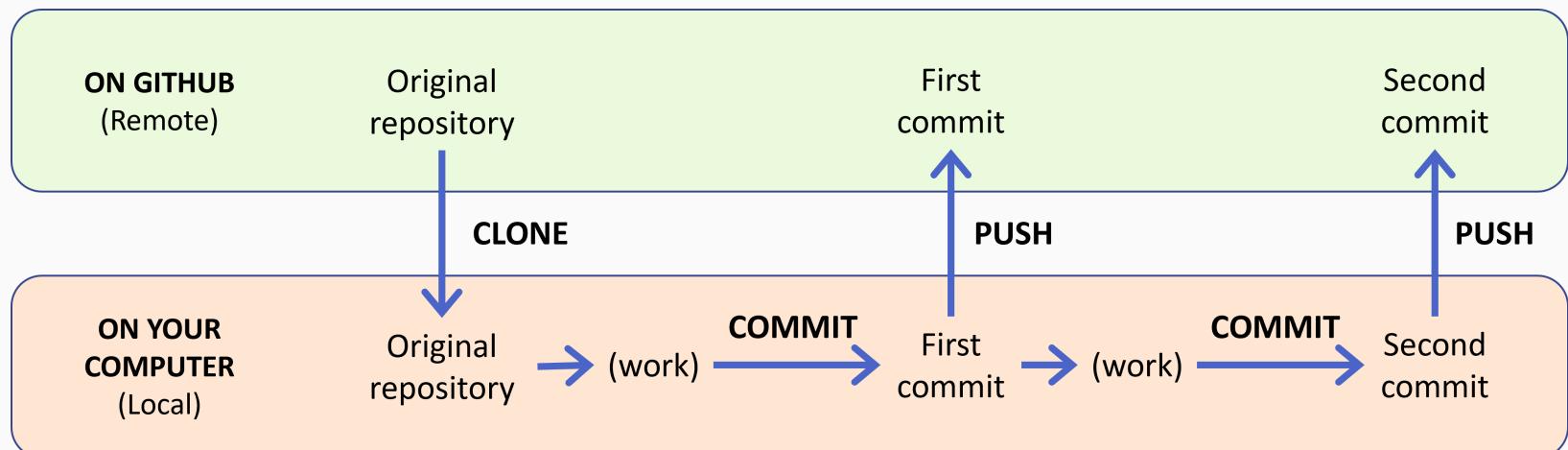
- The tools that we're using all form part of a coherent **data science ecosystem**.
- Greatly reduces the cognitive overhead ("aggregation") associated with traditional workflows, where you have to juggle multiple programs and languages at the same time.

# Github Desktop Workflow

With GitHub, if we were working

1. On our own
2. From a single computer

we could just follow the below workflow:



Since we're collaborating with others/potentially across machines, we'll also add in a few more actions.

# Main Git operations

The first Git operation in the workflow is **Cloning**



# (Git) Cloning

No, not *that* kind of cloning.

**Cloning:** making a local copy of a **GitHub Repository** (repo for short).

In order to clone a repo, we first need a repo to clone. Let's start by cloning our **Course repo!**

We can do this

1. From the repo page on github.com (direct link or SSH)
2. From GitHub Desktop directly

# Course Repo Cloning (github.com)

Let's start with the first approach. With GitHub Desktop installed/open, navigate to our [class web page repo](#).

The screenshot shows a GitHub repository page for 'afre-msu / AFRE-891-991-FS25'. The repository is public and contains several commits by user 'searsjm' from last week. The commits include updates to README.md, adding course logistics and lecture slides, and adding a LICENSE file. The repository has 0 forks, 0 stars, and 0 watchers. The 'Code' tab is selected. The 'About' section provides a brief description of the repository as a course repository for AFRE 891/991, Advanced Data Analytics, Fall Semester 2025.

afre-msu / AFRE-891-991-FS25

Code Issues Pull requests Actions Projects Security Insights Settings

AFRE-891-991-FS25 Public Edit Pins Watch 0 Fork 0 Star 0

main Go to file + Code About

searsjm Update README.md 97029c5 · last week

Course Logistics Add Course Logistics, Lecture 0 last week

Lecture Slides Add Lecture 1 Slides last week

LICENSE Add Course Logistics, Lecture 0 last week

README.md Update README.md last week

README License

About

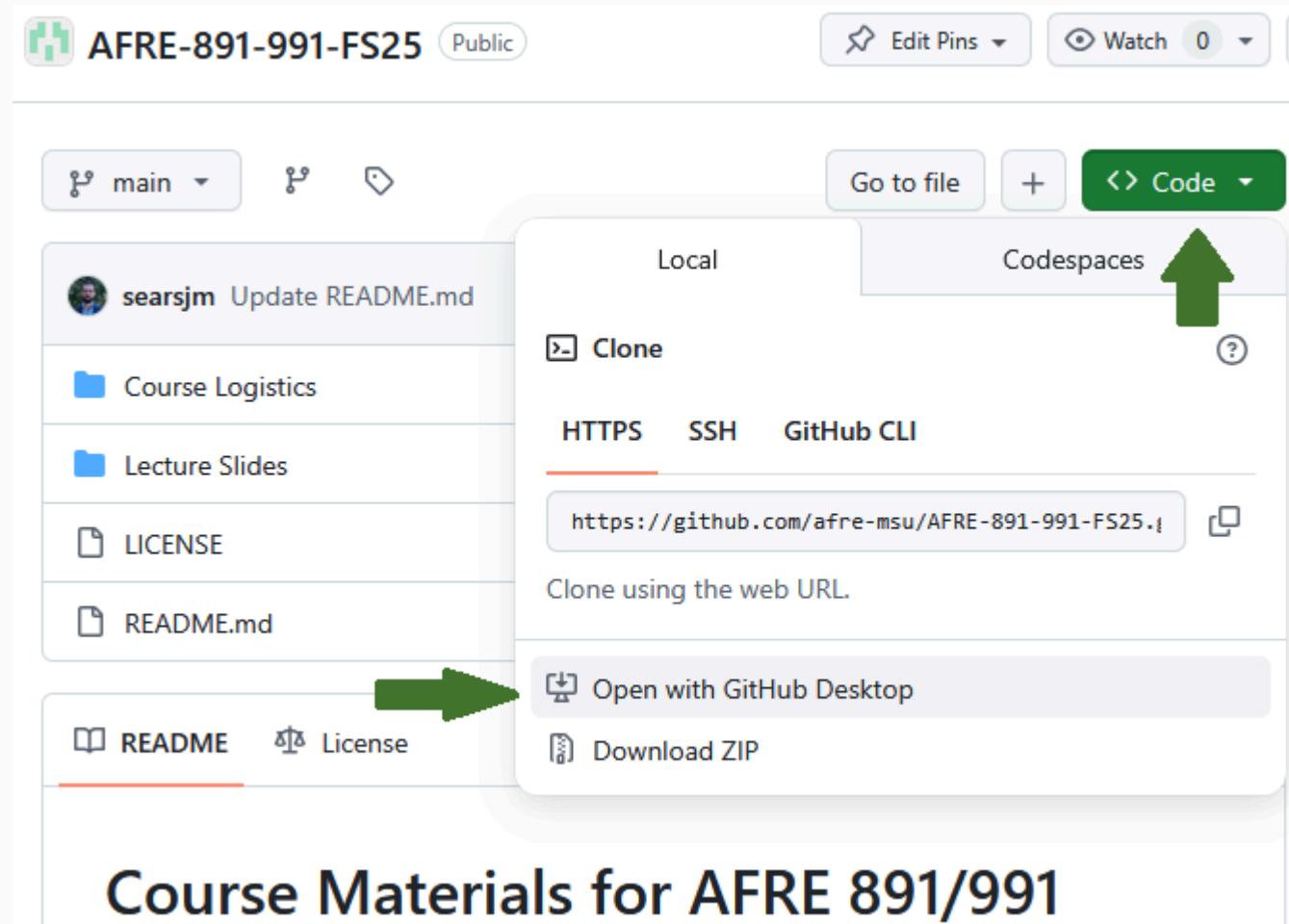
Course Repository for AFRE 891/991, Advanced Data Analytics, Fall Semester 2025

Readme View license Activity Custom properties 0 stars 0 watching 0 forks

Report repository

# Course Repo Cloning

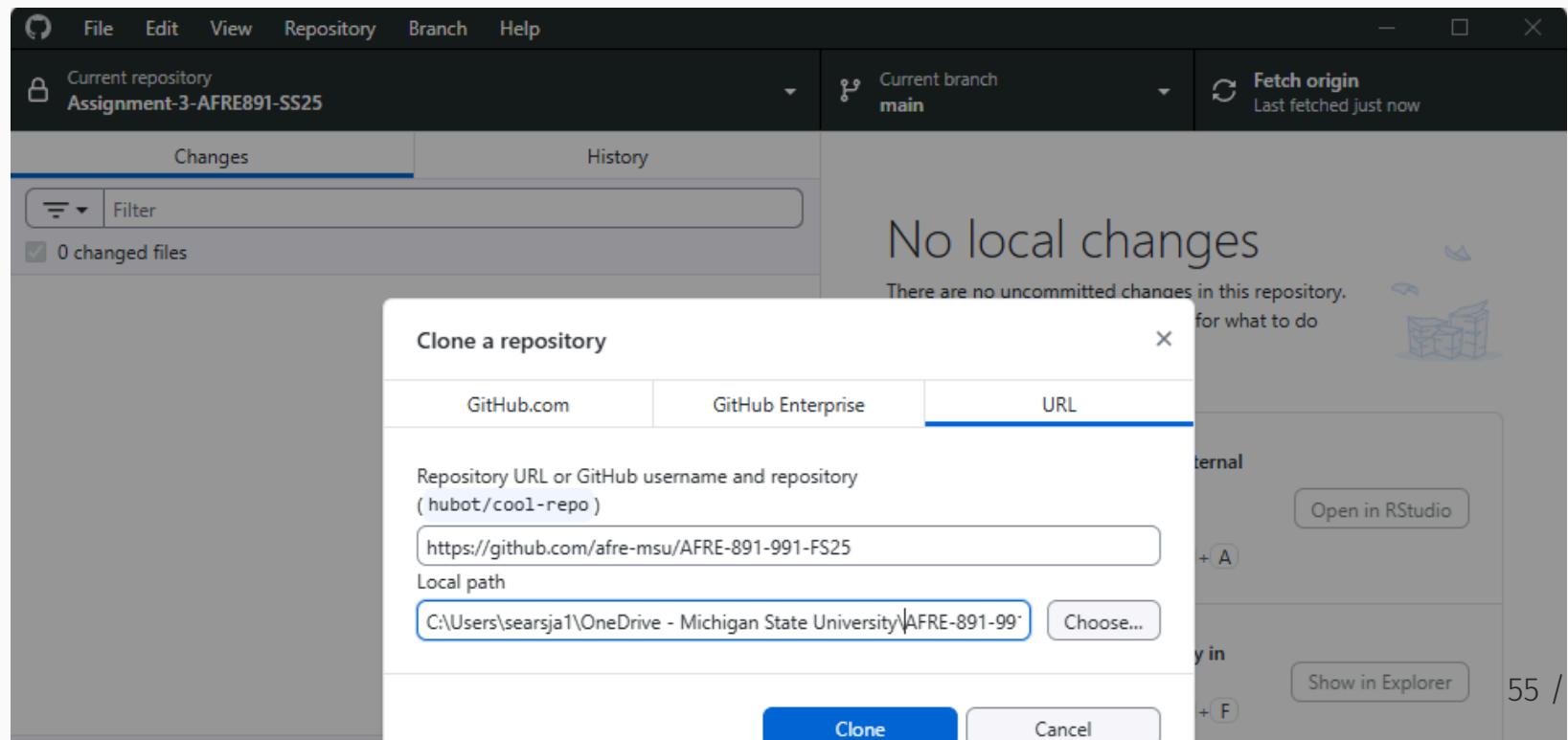
Click the green **Code** button and then "Open with GitHub Desktop"



# Course Repo Cloning

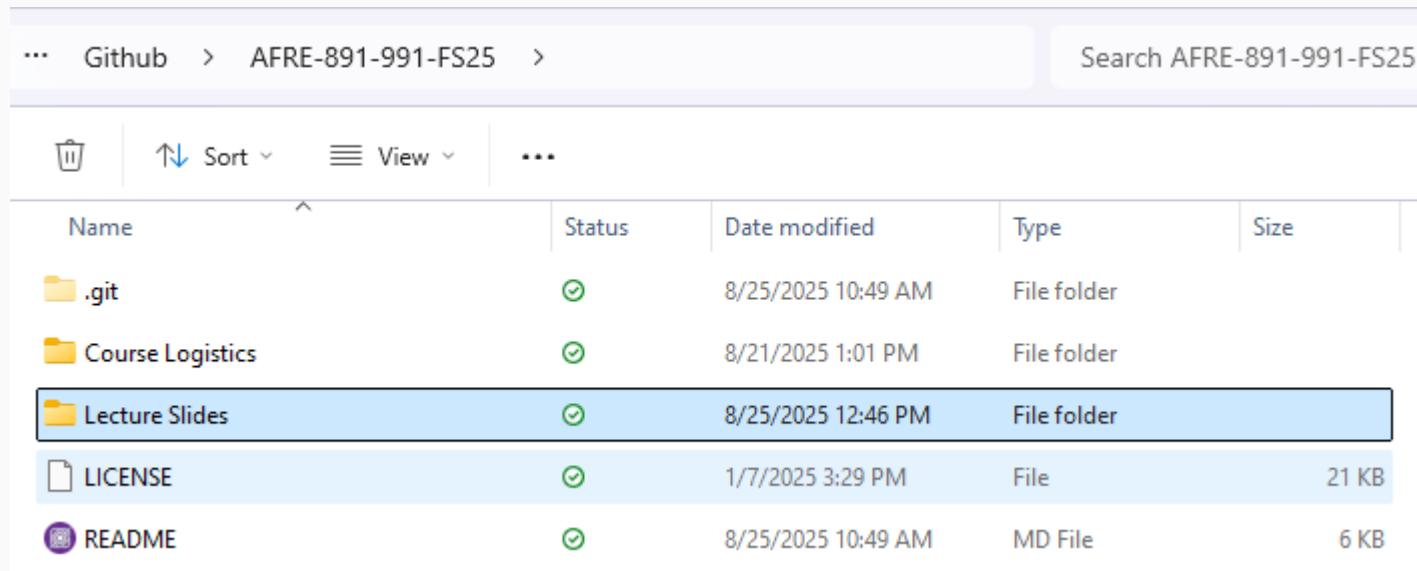
Allow your web browser to open the link. This will redirect you to GitHub Desktop, automatically adding in the repo URL.

Next, choose where you want the local copy of the repo file's saved and hit **Clone**



# Course Repo Cloning

Wait a little bit, and navigate to the local path you gave it. Voila!



The screenshot shows a file explorer interface with the following details:

- Path: ... > Github > AFRE-891-991-FS25 >
- Search bar: Search AFRE-891-991-FS25
- Toolbar: Includes a trash icon, Sort dropdown, View dropdown, and a three-dot menu.
- Table Headers: Name, Status, Date modified, Type, Size
- Table Data:
  - .git (File folder)
  - Course Logistics (File folder)
  - Lecture Slides (File folder) - This row is highlighted with a blue background.
  - LICENSE (File)
  - README (MD File)

Name	Status	Date modified	Type	Size
.git	✓	8/25/2025 10:49 AM	File folder	
Course Logistics	✓	8/21/2025 1:01 PM	File folder	
Lecture Slides	✓	8/25/2025 12:46 PM	File folder	
LICENSE	✓	1/7/2025 3:29 PM	File	21 KB
README	✓	8/25/2025 10:49 AM	MD File	6 KB

# Creating a Repo

While there are *tons* of awesome repositories on GitHub that we might want to clone and use as our starting point, other times we want to **create our own repo!**

Let's see how to do this from [github.com](https://github.com).

Start by creating your own repo on GitHub — call it "test" — and clone it, this time using the HTTPS link.<sup>3</sup>

<sup>3</sup> It's easiest to start with HTTPS, but **SSH** is advised for more advanced users.

# Create a Repository on GitHub (Repo)

searsjm

Type / to search

Overview Repositories 9 Projects Packages Stars

Popular repositories

- searsjm.github.io** Personal Website Public CSS
- databhub** Forked from yinitra-zz/ JupyterHubs for use by Python
- R-Notebook-Test** Public Jupyter Notebook
- test\_repo** Public archive Test repository for AFRE 891 lecture TeX
- covid\_ex** Public Example repo for filepaths
- Assignment-5-AFRE891-SS25** Public template Template repository for Assignment 5 AFRE 891 SS25

James Sears  
searsjm

I am an assistant professor in the Department of Agricultural, Food, and Resource Economics at Michigan State University.

Edit profile

2 followers • 0 following

Michigan State University

New issue

New repository

Import repository

New codespace

New gist

New organization

New project

122 contributions in the last year

Contribution settings ▾ 2025

Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
Mon												
Wed												
Fri												

2024 2023 2022 58 /

# Create a Repository on GitHub (Repo)

## Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

### 1 General

Owner \*



searsjm

Repository name \*

test

test is available.

Great repository names are short and memorable. How about [legendary-octo-broccoli?](#)

Description

Test Repo for AFRE 891/991 FS25

31 / 350 characters

### 2 Configuration

Choose visibility \*

Choose who can see and commit to this repository

Public

Start with a template

Templates pre-configure your repository with files.

No template

# Create a Repository on GitHub (Repo)

The screenshot shows a GitHub repository page for the user 'searsjm' with the repository name 'test'. The page includes a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, there are buttons for Pin, Watch (0), Fork (0), and Star (0). The main content area displays a list of files: 'LICENSE' and 'README.md' (both initial commits by 'searsjm' at 'now'), and 'README' and 'MIT license' (both by 'searsjm' at '1878bb8 · now'). On the right side, there is an 'About' section with details: 'Test Repo for AFRE 891/991 FS25', 'Readme', 'Activity', '0 stars', '0 watching', and '0 forks'. There are also sections for 'Releases' (no releases published) and 'Packages'.

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

test Public

Pin Watch 0 Fork 0 Star 0

main Go to file + Code About

searsjm Initial commit 1878bb8 · now 1 Commit

LICENSE Initial commit now

README.md Initial commit now

README MIT license

test

Test Repo for AFRE 891/991 FS25

Readme Activity 0 stars 0 watching 0 forks

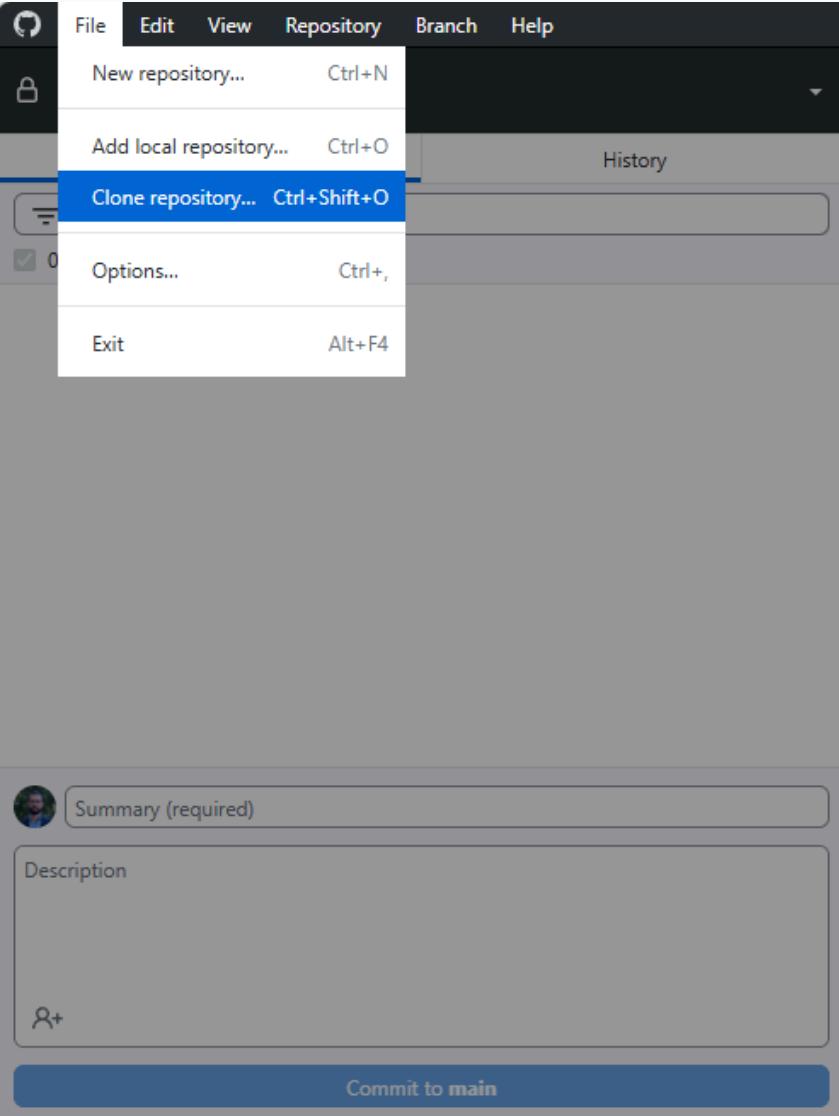
No releases published Create a new release

Packages

# Create a Repository on GitHub (Repo)

The screenshot shows a GitHub repository page for a repository named "test". The repository is owned by "searsjm" and is public. The main content area displays the repository's structure, including files like "LICENSE" and "README.md", and links to "README" and "MIT license". A modal window is open over the repository content, specifically the "Code" section. This modal provides cloning options: "Local" (selected) and "Codespaces", with "Clone" being the active tab. It shows three cloning methods: "HTTPS" (selected), "SSH", and "GitHub CLI". The "HTTPS" URL, <https://github.com/searsjm/test.git>, is highlighted with a blue border. Below the cloning options, there are links to "Open with GitHub Desktop" and "Download ZIP". To the right of the modal, the repository's "About" section is visible, containing details such as "Test Repo for AFRE 891/991 FS25", "Readme", "Activity", "0 stars", "0 watching", and "0 forks". Below the "About" section are sections for "Releases" (no releases published) and "Packages".

# Clone a Repository from GitHub (HTTPS)



The screenshot shows the GitHub Desktop application interface. The window title is "GitHub Desktop". The menu bar includes "File", "Edit", "View", "Repository", "Branch", and "Help". The "File" menu is open, showing options: "New repository...", "Add local repository...", "Clone repository...", "Options...", and "Exit". The "Clone repository..." option is highlighted with a blue selection bar. On the right side of the interface, there is a dark-themed sidebar with a user profile picture, a "Summary (required)" section, a "Description" section, and a search icon. The main area displays a message: "No local changes" with a sub-message: "There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next." Below this are three callout boxes: 1) "Open the repository in your external editor" with a "Select your editor in Options" link and an "Open in RStudio" button. 2) "View the files of your repository in Explorer" with a "Show in Explorer" button. 3) "Open the repository page on GitHub in your browser" with a "View on GitHub" button.

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor  
Select your editor in [Options](#) [Open in RStudio](#)

View the files of your repository in Explorer  
Repository menu or [Ctrl + Shift + F](#) [Show in Explorer](#)

Open the repository page on GitHub in your browser  
Repository menu or [Ctrl + Shift + G](#) [View on GitHub](#)

Commit to main

62 / 118

# Clone a Repository from GitHub (HTTPS)

The screenshot shows the GitHub desktop application interface. At the top, there's a menu bar with File, Edit, View, Repository, Branch, and Help. Below the menu is a toolbar with icons for repository status (green padlock), current branch (main), and fetch origin (refresh icon). The main window displays a repository named "Assignment-3-AFRE891-SS25". Under the repository name, it says "Current branch main" and "Fetch origin Last fetched 1 minute ago". Below this, there are tabs for "Changes" (selected) and "History". A sidebar on the left shows a list of "0 changed files". In the center, a modal dialog titled "Clone a repository" is open. It has tabs for "GitHub.com" (disabled), "GitHub Enterprise" (disabled), and "URL" (selected). The URL input field contains "https://github.com/searsjm/test.git". Below the URL is a "Local path" input field containing "C:\Users\searsja1\OneDrive - Michigan State University\Github\test". At the bottom of the dialog are two buttons: "Clone" (blue) and "Cancel". In the background, there's a message "No local changes" with a sub-message "There are no uncommitted changes in this repository." and a "What to do" section with icons for "Open in RStudio" and "Show in Explorer". At the bottom right of the main window, there's a "Commit to main" button.

No local changes

There are no uncommitted changes in this repository.

for what to do

Clone a repository

GitHub.com GitHub Enterprise URL

Repository URL or GitHub username and repository  
( hubot/cool-repo )

https://github.com/searsjm/test.git

Local path

C:\Users\searsja1\OneDrive - Michigan State University\Github\test

Clone Cancel

Summary (required)

Description

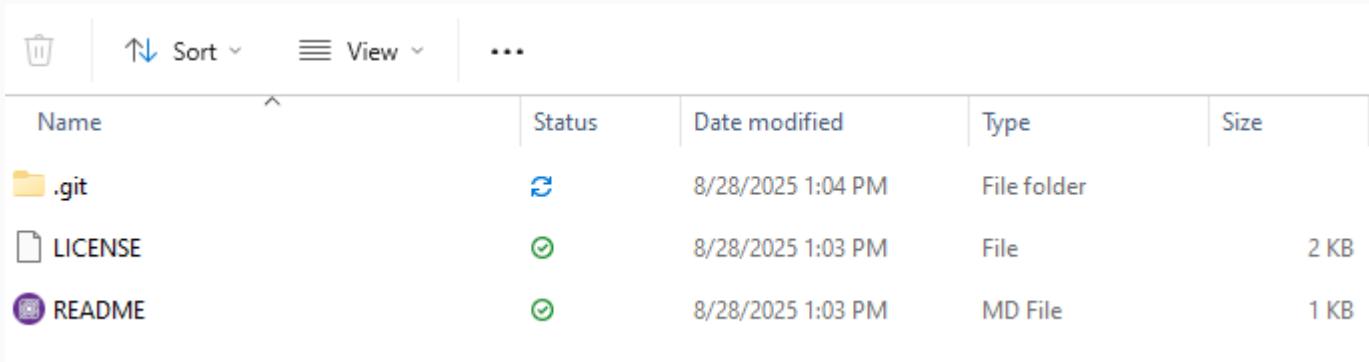
Open the repository page on GitHub in your browser

Repository menu or Ctrl + Shift + G

View on GitHub

Commit to main

# Clone a Repository from GitHub (HTTPS)



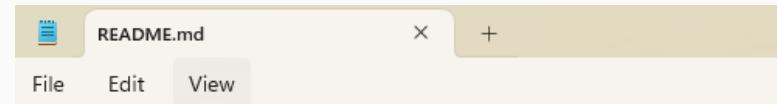
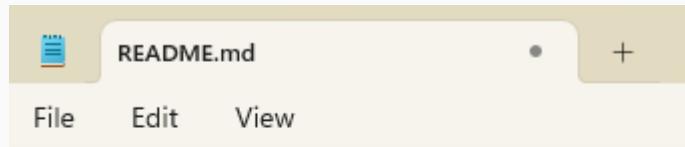
A screenshot of a file explorer interface showing the contents of a cloned GitHub repository. The repository contains three items: a '.git' folder, a 'LICENSE' file, and a 'README' file.

Name	Status	Date modified	Type	Size
.git	⟳	8/28/2025 1:04 PM	File folder	
LICENSE	🕒	8/28/2025 1:03 PM	File	2 KB
README	🕒	8/28/2025 1:03 PM	MD File	1 KB

# Making Local Changes

Let's start by making some changes to our local copies of the test repo files.

Open up the README file. Add some text and save the README.



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

# Making Local Changes

Open up Github Desktop. Do you see any changes?

The screenshot shows the Github Desktop application interface. At the top, there's a menu bar with File, Edit, View, Repository, Branch, and Help. Below the menu, it says "Current repository test" and "Current branch main". A "Fetch origin" status indicates it was last fetched 4 minutes ago.

The main area is divided into two tabs: "Changes" (which is selected) and "History". Under "Changes", there's a "Filter" input field and a checkbox that says "1 changed file". The file listed is "README.md".

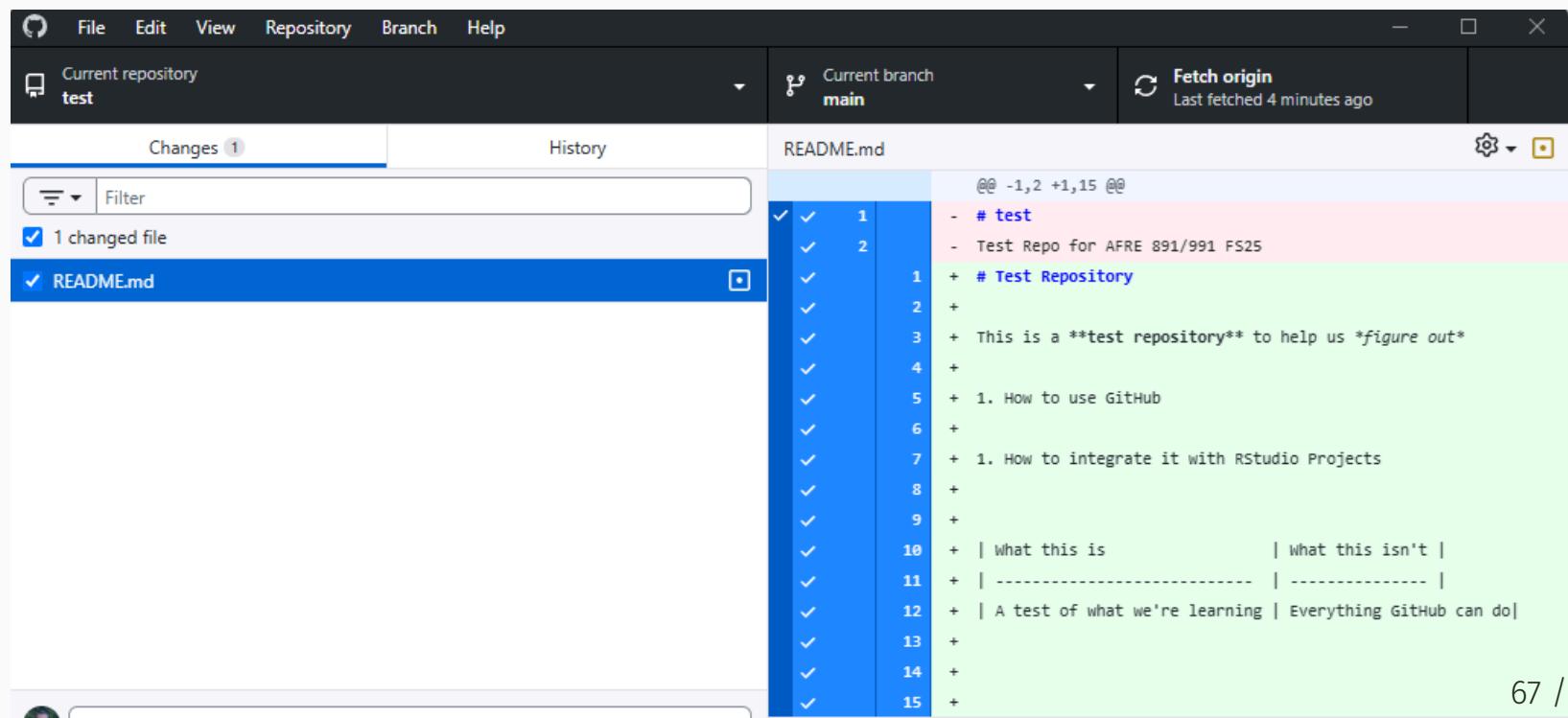
The central part of the screen displays a diff view for "README.md". The left column shows line numbers from 1 to 15, each preceded by a checkmark. The right column shows the actual text of the file. The text content includes a header, a section about testing, and a conclusion. The diff highlights some changes, such as the addition of "# Test Repository" and the removal of "Test Repo for AFRE 891/991 FS25".

At the bottom left, there's a "Update README.md" button with a user icon. Below it is a "Description" input field and a "Commit" button. The commit message placeholder is "Commit 1 file to main".

# Stage

Github Desktop detects local changes and automatically **stages** those changes

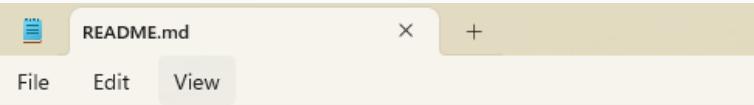
- Tells Git that you edited the `README.md` file and documents *how* it changed from the upstream repo version



# Commit and Push

For now, these changes are **local to our own repo copy**.

### Local System



The screenshot shows a local file editor window for a README.md file. The file contains the following text:

```
# Test Repository

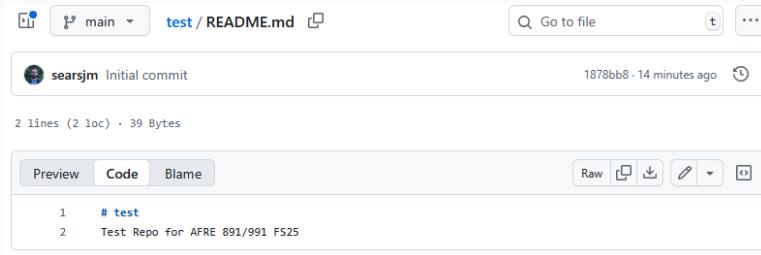
This is a **test repository** to help us *figure out*

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|
```

### GitHub



The screenshot shows a GitHub commit page for a README.md file in a repository named "test". The commit was made by "searsjm" and is titled "Initial commit". It was pushed 14 minutes ago with a commit hash of 1878bb8. The commit message is:

```
1 # test
2 Test Repo for AFRE 891/991 FS25
```

In order for them to **update the main repo itself**, we need to do two actions: **Commit** and **Push**.

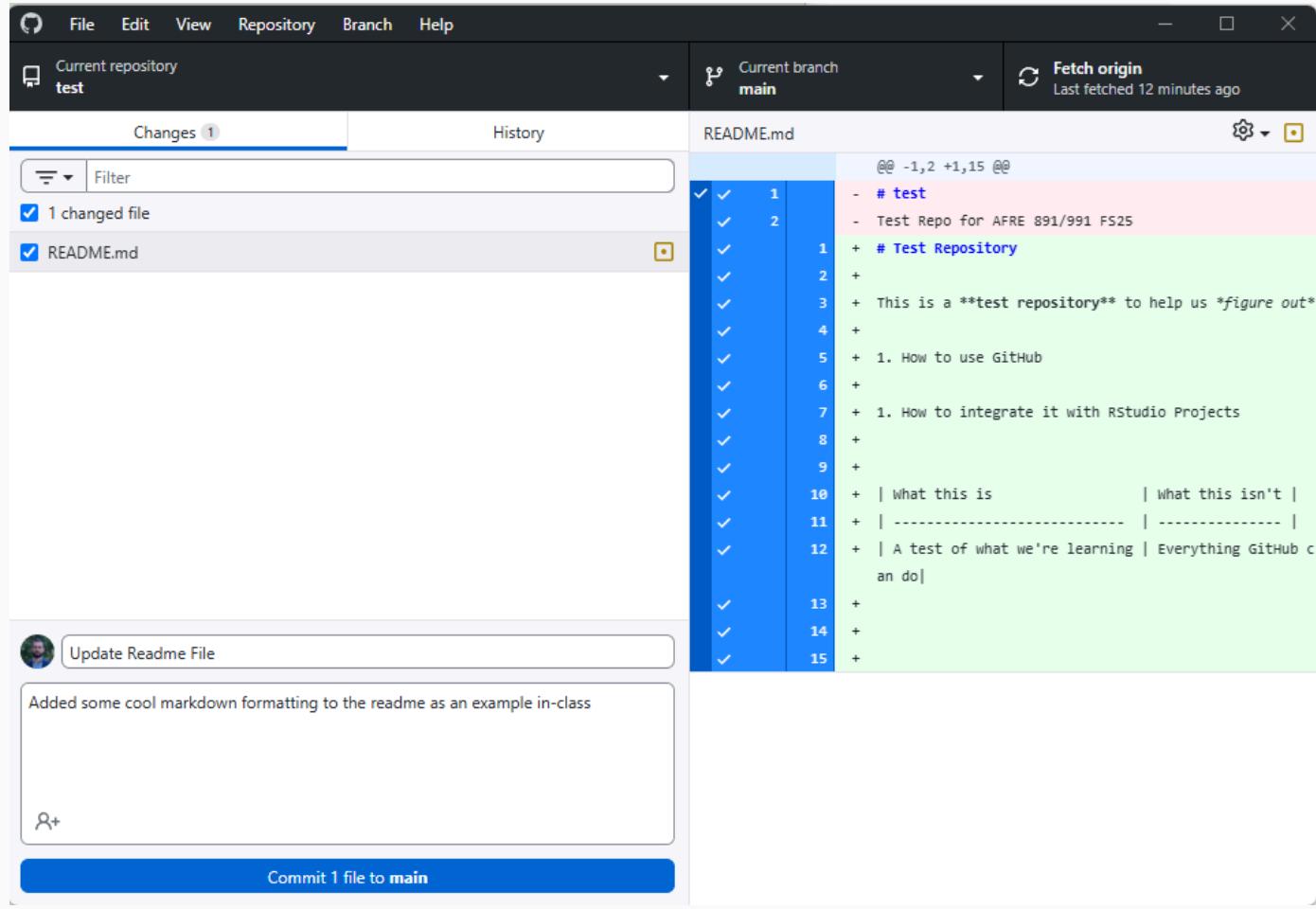
# Commit

Github Desktop automatically stages any changes it detects, so that's part of the trouble taken care of already!

Now we need to **Commit**: tell Git that, yes, you are sure these staged changes should be part of the repo history.

# Commit

To **commit** in GitHub Desktop, we need to first give our commit a name and a description (bottom-left):



# Push

Next we need to **Push**: send the local file changes/additions to update the repo on GitHub.

In the upper-right corner you should now see the option to **Push origin**:

The screenshot shows the GitHub desktop application interface. At the top, there's a menu bar with File, Edit, View, Repository, Branch, and Help. Below the menu, there are dropdowns for 'Current repository' set to 'test' and 'Current branch' set to 'main'. To the right of these dropdowns is a toolbar with a 'Push origin' button, which has a blue outline and the text 'Push origin' above it. Below the toolbar, there are two tabs: 'Changes' (which is selected) and 'History'. Under the 'Changes' tab, there's a 'Filter' section with a dropdown and a checkbox that says '0 changed files'. On the right side of the application, there's a large message area with the heading 'No local changes'. It says, 'There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.' Below this, there's a box containing the text 'Push commits to the origin remote' and 'You have 1 local commit waiting to be pushed to GitHub.', followed by a 'Push origin' button and a keyboard shortcut 'Ctrl + P'. At the bottom of the application window, there's a summary section with a user profile picture, the text 'Summary (required)', and a 'Description' field. There are also buttons for 'Open in RStudio' and 'Open in external editor'.

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote  
You have 1 local commit waiting to be pushed to GitHub.

Push origin

Always available in the toolbar when there are local commits waiting to be pushed or `Ctrl + P`

Summary (required)

Description

Open the repository in your external editor  
Select your editor in [Options](#)

Open in RStudio

71 / 118

# Push

Click it and wait. That's it!

The screenshot shows the RStudio interface with the GitHub extension. At the top, the menu bar includes File, Edit, View, Repository, Branch, and Help. The current repository is set to 'test'. The current branch is 'main', and it has been fetched just now. The main pane displays a message: 'No local changes' with a note that there are no uncommitted changes in the repository. It provides suggestions for what to do next, such as opening the repository in an external editor (with options for RStudio or another editor), viewing files in the Explorer, or opening the GitHub repository page. A summary section at the bottom indicates that a summary is required.

Current repository  
test

Current branch  
main

Fetch origin  
Last fetched just now

Changes History

Filter

0 changed files

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor  
Select your editor in [Options](#)  
Repository menu or [Ctrl + Shift + A](#)

Show in Explorer  
Repository menu or [Ctrl + Shift + F](#)

Open the repository page on GitHub in your browser  
Repository menu or [Ctrl + Shift + G](#)

Summary (required)

Description

72 / 118

# Changes Now Visible on GitHub

The screenshot shows a GitHub repository page for a repository named 'test'. The top navigation bar includes options for 'Pin', 'Watch' (0), 'Fork' (0), and 'Star' (0). Below the header, there are buttons for switching branches ('main'), creating new files ('+'), and viewing code ('Code'). The main content area displays three commits:

- searsjm** Update Readme File · bc4647c · 23 minutes ago
- LICENSE · Initial commit · 43 minutes ago
- README.md · Update Readme File · 23 minutes ago

The 'README' file is currently selected. The repository details on the right side include:

- Test Repo for AFRE 891/991 FS25
- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 0 forks

## 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

Releases

No releases published  
[Create a new release](#)

Packages

No packages published  
[Publish your first package](#)

# Recap

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

- Cloned a repo from GitHub
- Made same changes to a repo file and saved them locally.
- **Staged** these local changes (okay, Github Desktop did that for us).
- **Committed** these local changes to our Git history with a helpful message.
- **Pushed** our changes to the GitHub repo.

This completes the single collaborator/computer workflow.

But what if we're in a multiple author and/or multiple system situation?

**A:** we need to bring the last Git Action: **Pull**.

# Pull

Just like we **Pushed** our downstream changes to the GitHub repo, we can also **Pull** upstream changes *from* the repo

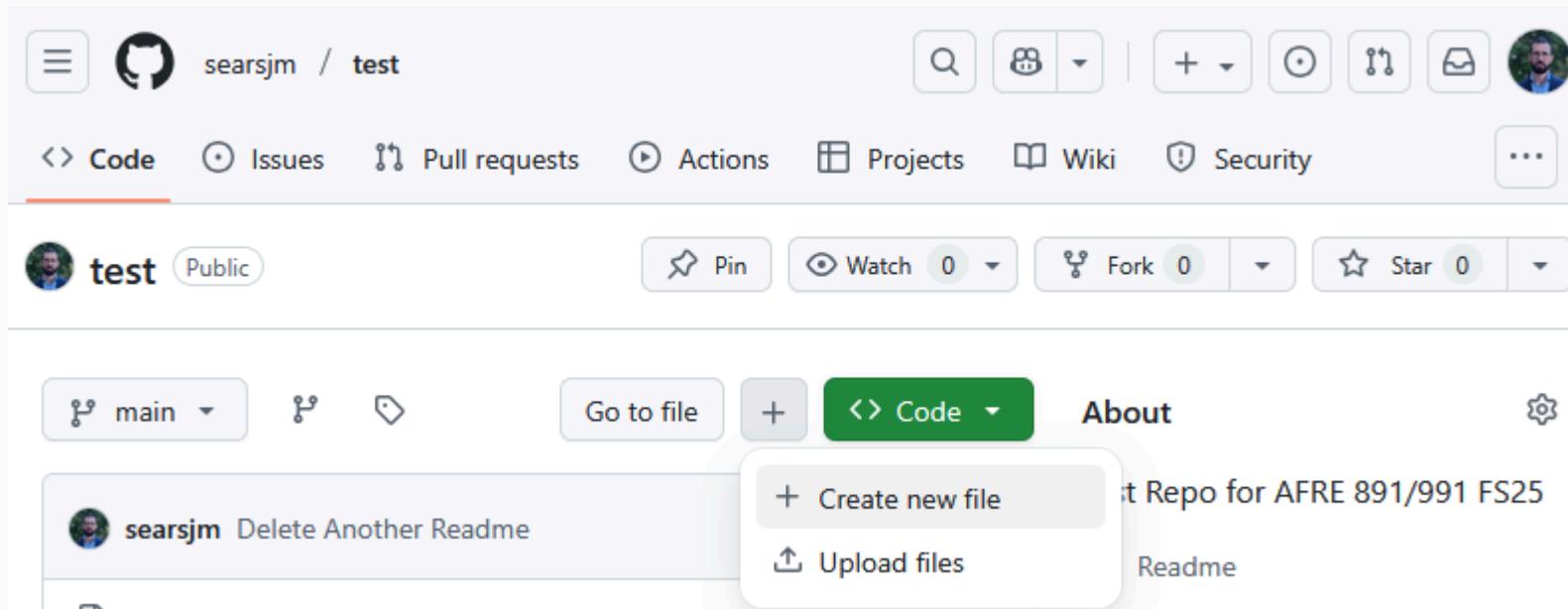
- Overwrites our current version of files with the newer versions from the GitHub repo

When working with others, it's **critical** to **Pull** updates from the GitHub repo just in case anyone else made changes too (not expected here, but good practice).

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.

# Pull

To set up a possible pull, let's make some **upstream changes**. On Github, click on the plus symbol to **add a new file**.



# Pull

The screenshot shows a GitHub pull request interface. At the top, there's a navigation bar with links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and a three-dot menu. Below the navigation bar, the URL is `test / Another Readme.md` and the branch is `in main`. There are two buttons: `Cancel changes` and a green `Commit changes...` button. Below these buttons are `Edit` and `Preview` tabs, with `Edit` selected. To the right of the preview area are buttons for `Spaces`, `2`, and `No wrap`. The main content area shows a diff:

```
1 # Another Readme
2
3 Oh dang, now there's two of them!
```

# Pull

The screenshot shows a GitHub commit dialog for a pull request. At the top, it displays the repository path `test / Another Readme.md` and the branch `in main`. There are buttons for `Cancel changes` and `Commit changes...`. On the left, there's a preview pane showing the file content:

```
1 # Another Readme
2
3 Oh dang, now t
```

The main area is titled `Commit changes` and contains the following fields:

- Commit message:** A text input field containing `Create Another Readme.md`.
- Extended description:** A large text area with placeholder text `Add an optional extended description...`.
- Commit Email:** An input field containing `james.sears.m@gmail.com`.
- Branch Selection:** Two radio button options:
  - Commit directly to the `main` branch
  - Create a **new branch** for this commit and start a pull request [Learn more about pull requests](#)

# Pull

main ▾ test / Another Readme.md ⚙ Go to file t ⋮

 searsjm Create Another Readme.md 72cade8 · now ⏱

3 lines (2 loc) · 52 Bytes

Preview Code Blame Raw ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

## Another Readme

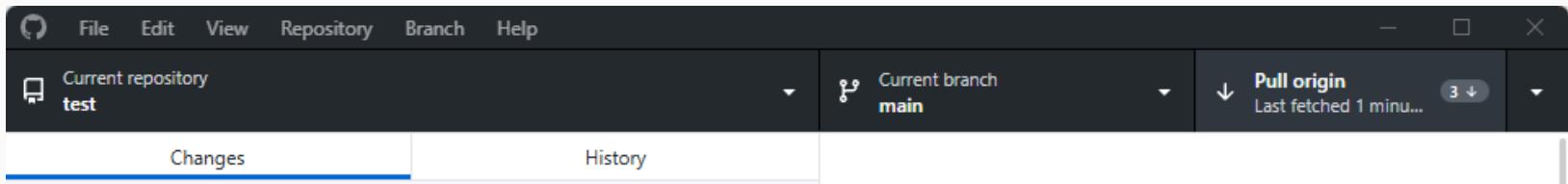
---

Oh dang, now there's two of them!

# Pull

Flip back over to GitHub Desktop. One of two things may have happened.

1. Your GitHub Desktop already detected the upstream changes

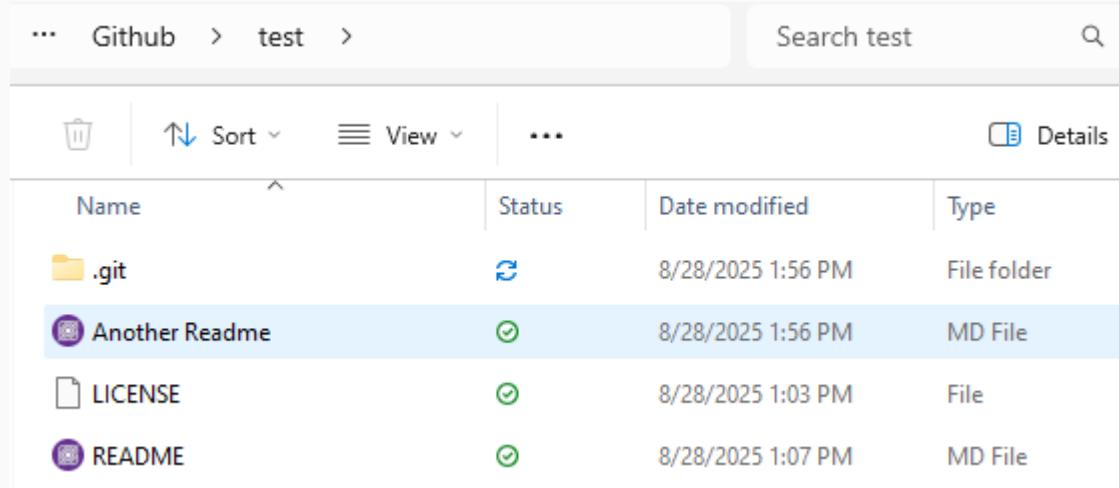


If you just see **fetch origin**, click it to refresh and retrieve the changes.

# Pull

Click the **pull origin** button to add the new file to your local clone

- Same thing would have happened if someone had modified existing files



A screenshot of a GitHub repository interface. The top navigation bar shows 'Github > test >'. A search bar is present on the right. Below the header is a toolbar with icons for delete, sort, view, and details. A table lists the contents of the repository:

Name	Status	Date modified	Type
.git	⟳	8/28/2025 1:56 PM	File folder
Another Readme	✓	8/28/2025 1:56 PM	MD File
LICENSE	✓	8/28/2025 1:03 PM	File
README	✓	8/28/2025 1:07 PM	MD File

# 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, 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.

# 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 obsolete!

# 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:

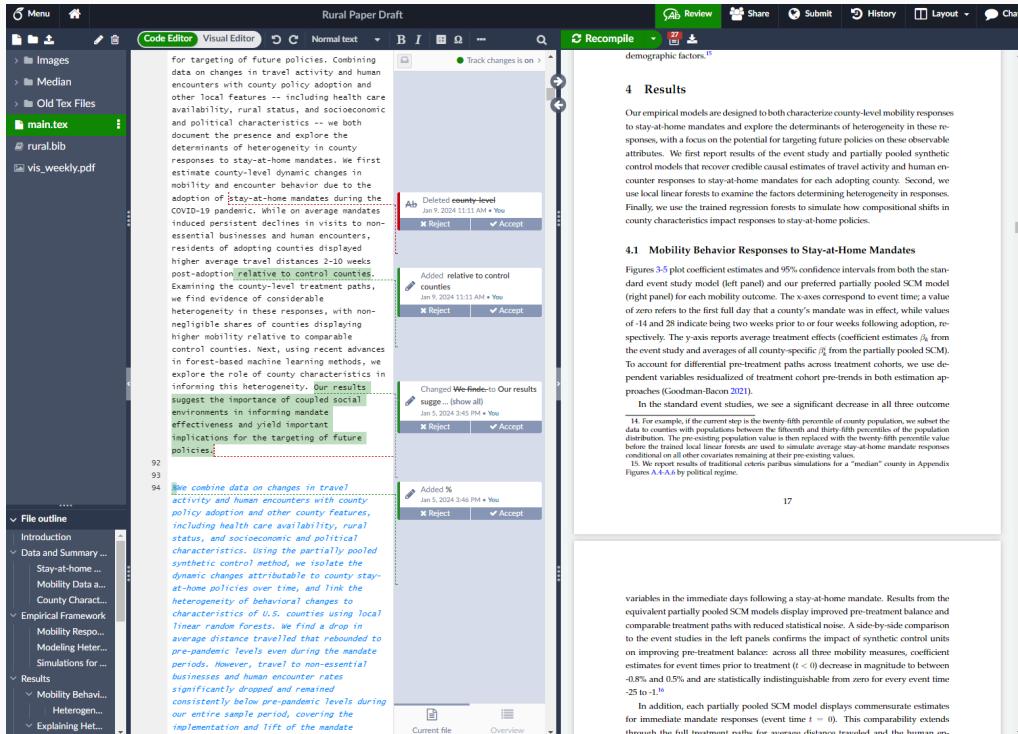
- **Dropbox**: 2GB free (\$10/mo for 2TB)
- **Box**: 10GB free (\$10/mo for 100GB)

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

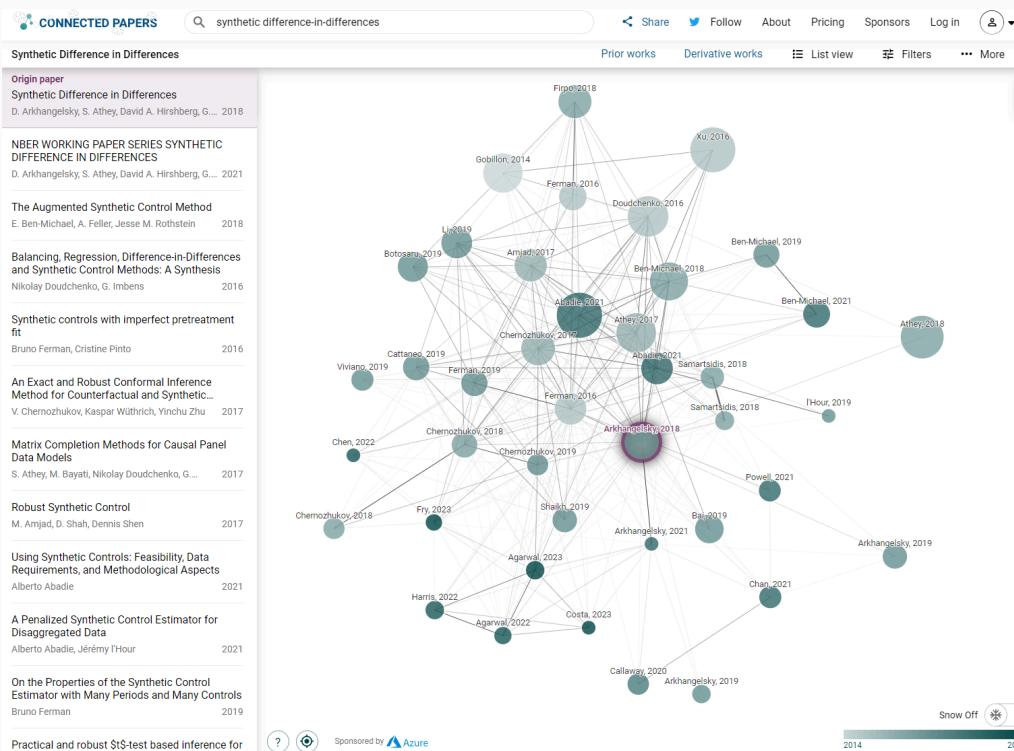
If you typeset using LaTeX, **Overleaf** streamlines access and collaboration



**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)**

# Productivity Miscellanea

**Your Productivity Tips + Tricks?**

# Git(Hub) and RStudio

# Git(Hub) Integration in RStudio

One of the (many) 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).
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**.

Let's practice this using our existing test repo. Start by copying the repo link.

# Create new R Project

To start, we need to create an **R Project**

- A project (i.e. folder) specific to a given task

Open a new RStudio window and click "**File > New Project**"

# Version Control

R test - main - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Console Terminal x Background Jobs x

R 4.2.2 · F:/OneDrive - Michigan State University/Github/test/

```
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or our package.
```

Type 'demo()' for some demos, 'help()' for help, 'start()' for an HTML browser. Type 'q()' to quit R.

> |

New Project Wizard

Create Project

-  **New Directory**  
Start a project in a brand new working directory
-  **Existing Directory**  
Associate a project with an existing working directory
-  **Version Control**  
Checkout a project from a version control repository

Cancel

Environment History Connections Git Tutorial

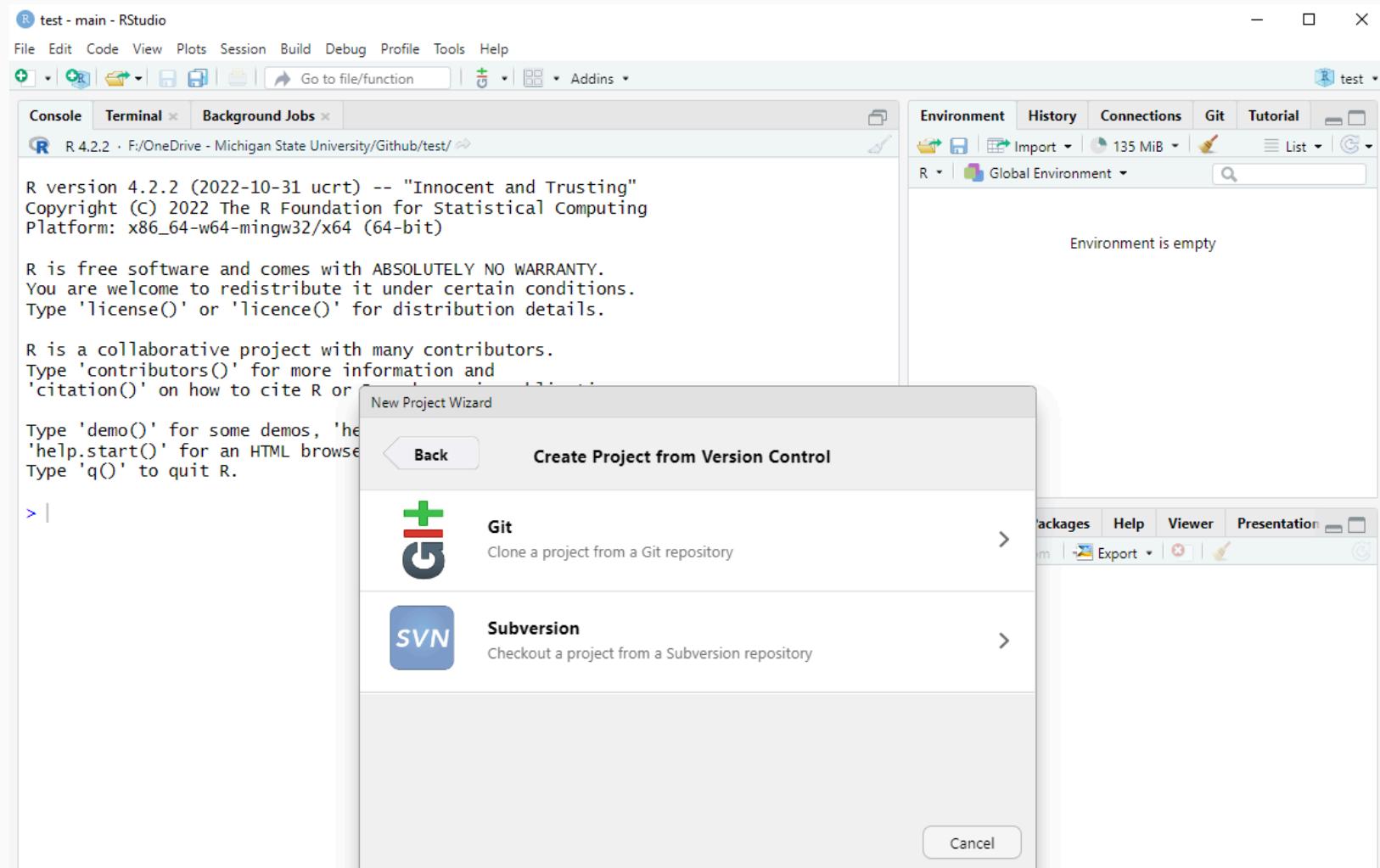
Import 135 MiB List Global Environment

Environment is empty

Packages Help Viewer Presentation

Export

# Choose Git



# Add Repo info

New Project Wizard

Back

## Clone Git Repository



Repository URL:

Project directory name:

Create project as subdirectory of:

Open in new session

# Add Repo into RStudio

The screenshot shows the RStudio interface with a GitHub repository named 'test' loaded. The top navigation bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with various icons. The main area has tabs for Console, Terminal, and Background Jobs. The Console tab displays the R startup message and basic usage instructions. The Global Environment panel shows that the environment is currently empty. The Files panel lists the contents of the repository, which include a .gitignore file (44 B), a LICENSE file (1.1 KB), a README.md file (25 B), and a test.Rproj project file (250 B).

R test - main - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Console Terminal × Background Jobs ×

R 4.2.2 · F:/OneDrive - Michigan State University/Github Test/test/

```
R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.

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 Tutor

test — Github Test

Environment is empty

Files Plots Packages Help Viewer Present

OneDrive - Michigan State University > Github Test > test

	Name	Size
	..	
	.gitignore	44 B
	LICENSE	1.1 KB
	README.md	25 B
	test.Rproj	250 B

# 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.<sup>4</sup>

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.)

<sup>4</sup> They're important, but not for the purposes of this section.

# Stage and Commit

The screenshot shows the RStudio interface with a project titled "test - main - RStudio". The left pane displays the contents of the "README.md" file:

```
1 # Test Repository
2
3 This is a **test repository** to help us *figure out*
4 1. How to use GitHub
5 1. How to integrate it with RStudio Projects
6
7 | what this is | what this isn't |
8 |-----|-----|
9 | A test of what we're learning | Everything GitHub can do|
```

The right pane shows the Git status. The "Status" tab is selected, displaying the following staged changes:

- .gitignore (Untracked)
- README.md (Modified)
- test.Rproj (Untracked)

A tooltip above the list says "Commit pending changes (Ctrl+Alt+M)". Below the status bar, the "Files" tab is active, showing a file tree with ".gitignore" selected.

# Stage and Commit

RStudio: Review Changes

Changes History main Stage Revert Ignore

Your branch is ahead of 'origin/main' by 1 commit.

Staged	Status	Path
<input type="checkbox"/>	<span>?</span> <span>?</span>	.gitignore
<input checked="" type="checkbox"/>	<span>M</span>	README.md
<input type="checkbox"/>	<span>?</span> <span>?</span>	test.Rproj

Commit message 18 characters  
Update readme text

Amend previous commit

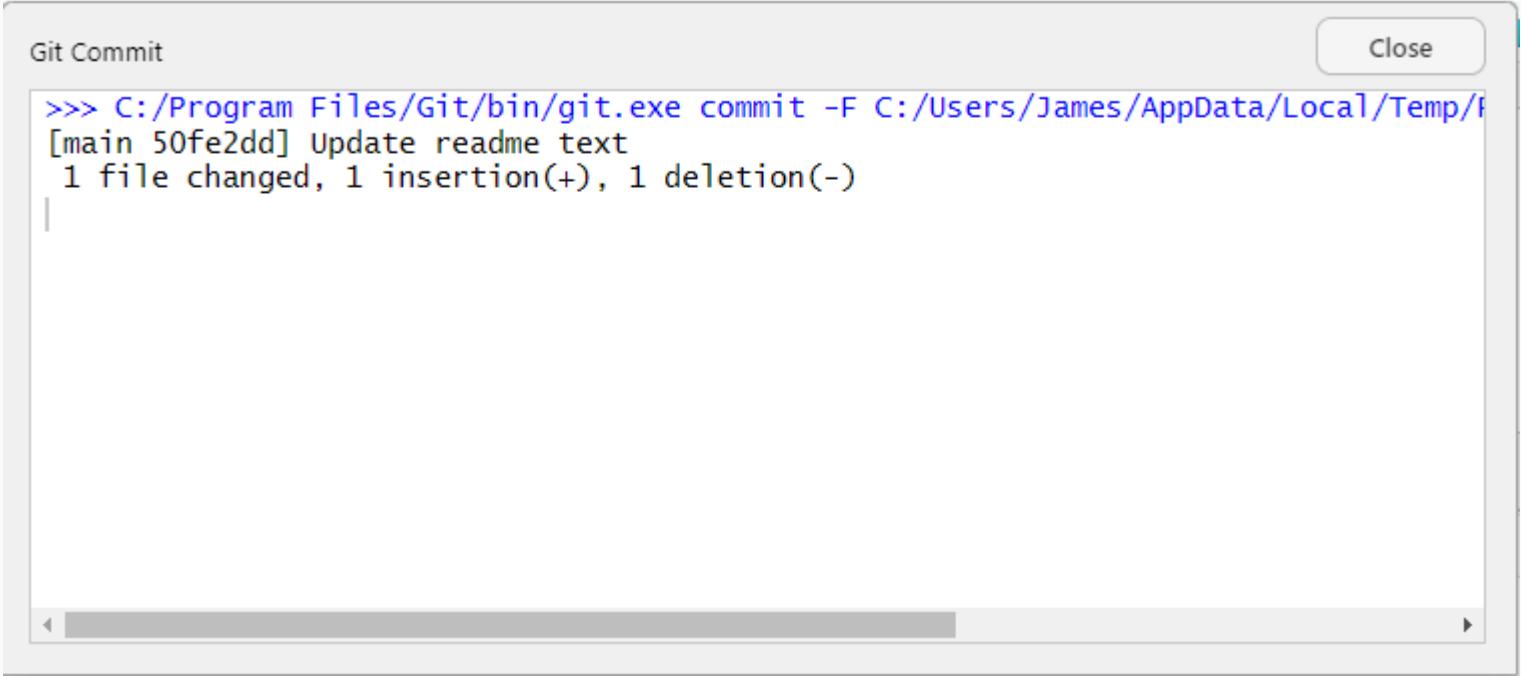
Show Staged Unstaged Context 5 lines  Ignore Whitespace  Unstage All

@@ -1,8 +1,8 @@

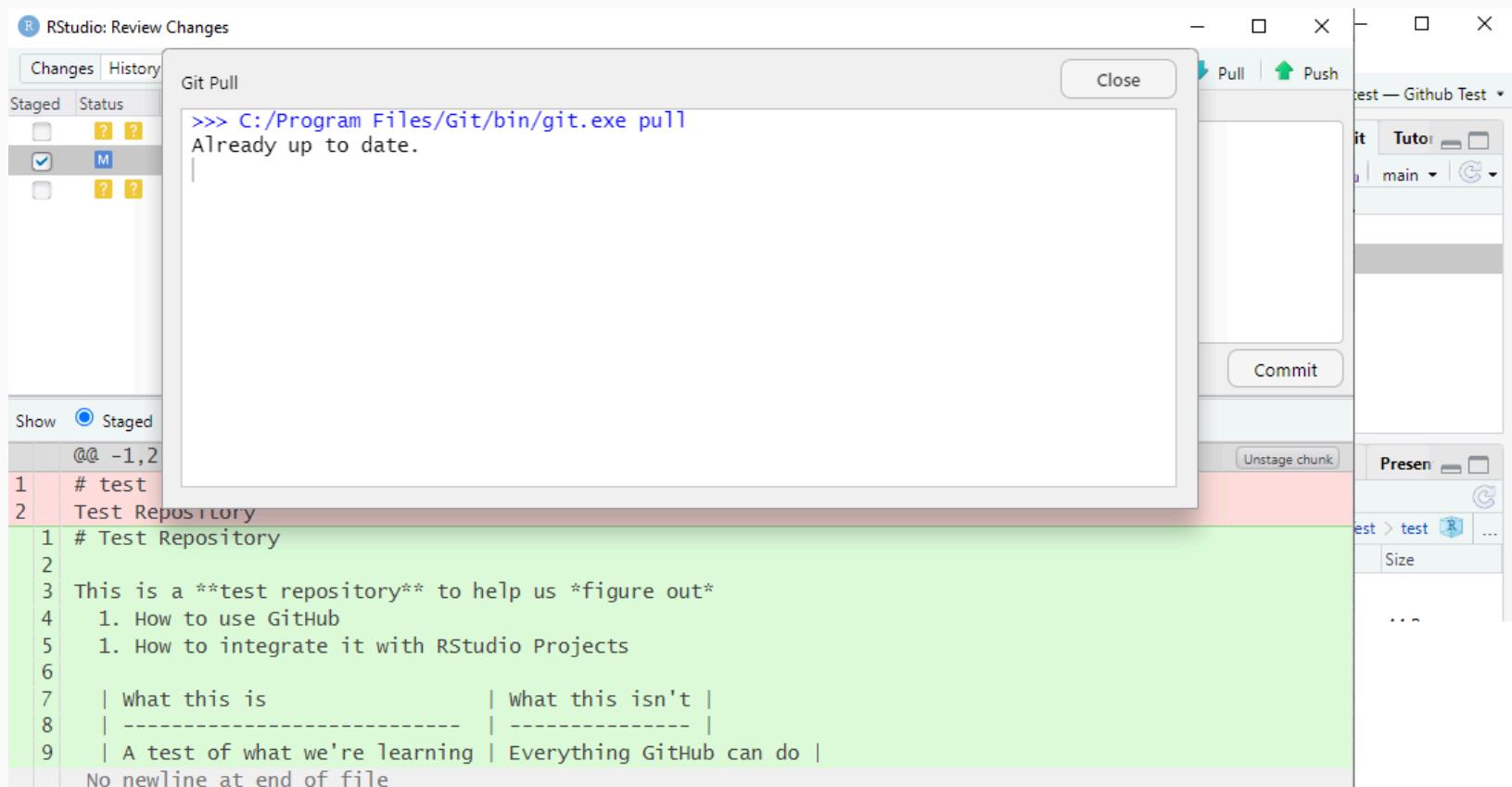
```
1 | 1 # Test Repository
2 | 2
3 | This is a **test repository** to help us *figure out*
3 | This is a **test repository** to help us *figure out*
4 | 1. How to use GitHub
5 | 1. How to integrate it with RStudio Projects
6 |
7 | what this is | what this isn't |
8 | ----- | ----- |
```

Unstage chunk

# Stage and Commit

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

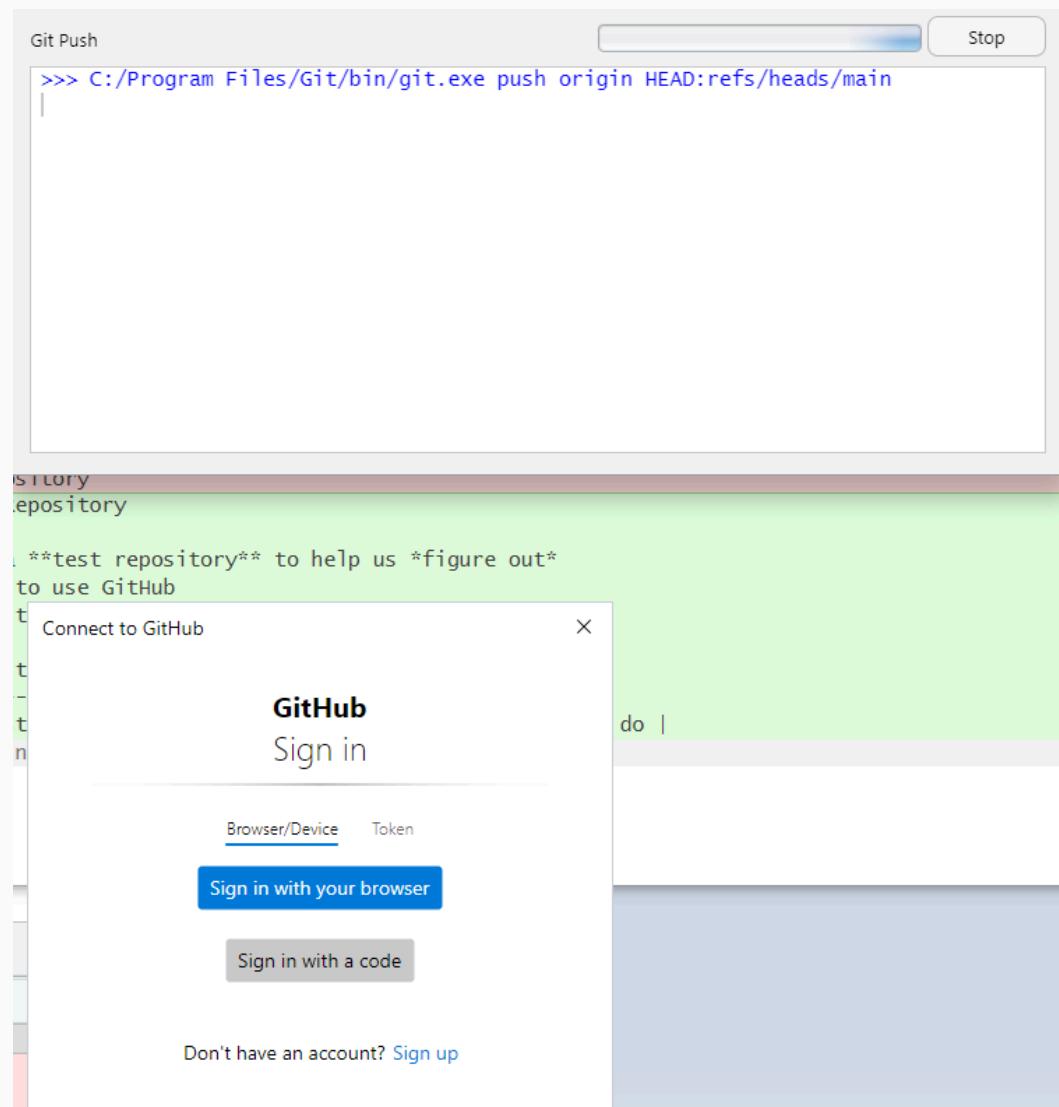
# Pull



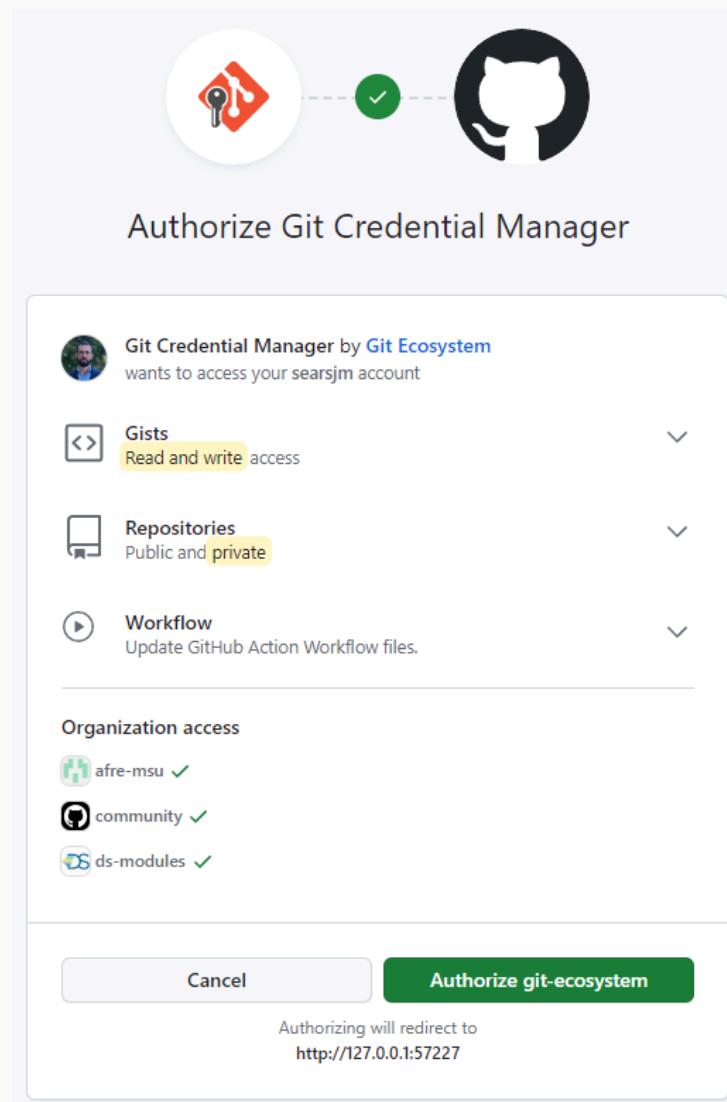
# Push



# Sign RStudio into Github



# Sign RStudio into Github



# Changes Now Visible on Github

The screenshot shows a GitHub repository page for the user 'searsjm' named 'test'. The repository is public and contains one branch ('main') and no tags. The commit history shows three commits: an initial commit for 'LICENSE' by 'Initial commit' 42 minutes ago, and two updates for 'README.md' by 'searsjm' (one at 'now' and one at '52525f2 · now'). The repository has 0 stars, 1 watcher, 0 forks, and 3 commits. The README file includes a table comparing what GitHub is and what it isn't.

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 + 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

OneDrive - Michigan State University > Github > test

Name	Status	Date modified	Type	Size
.git	✓	1/15/2024 4:35 PM	File folder	
.Rproj.user	✓	1/15/2024 3:47 PM	File folder	
.gitignore	✓	1/15/2024 3:47 PM	Text Document	1 KB
.Rhistory	✓	1/15/2024 4:35 PM	RHISTORY File	0 KB
LICENSE	✓	1/15/2024 3:47 PM	File	2 KB
README.md	✓	1/15/2024 4:32 PM	MD File	1 KB
test.Rproj	✓	1/15/2024 4:32 PM	R Project	1 KB

2023

README.md — Github\test — F:\OneDrive - Michigan State University\Github\test — Atom

File Edit View Selection Find Packages Help

Project Sunsetting Atom README.md — Course... README2.md README.md — GitHub...

```
1 # Test Repository
2
3 This is a **test repository** to help us *figure out*
4 1. How to use GitHub
5 1. How to integrate it with RStudio Projects
6
7 | What this is | What this isn't |
8 | ----- | ----- |
9 | A test of what we're learning | Everything GitHub can do |
```

iles

University

igan State University

nal

Your project is currently empty

Add folders

Reopen a project

# 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: 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.<sup>5</sup>

Note however that R Markdown files do the same thing re: paths *and* extend functionality to incorporating text, images, latex, and html.

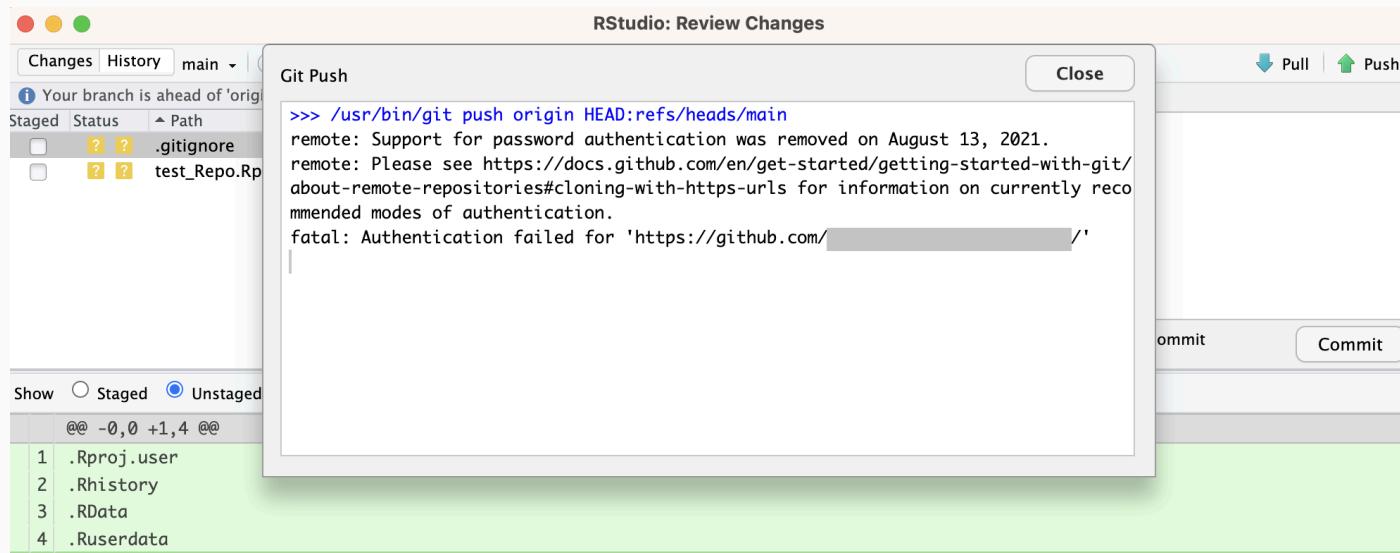
<sup>5</sup> Calling files from their full `YourComputer/YourName/Documents/Special-Subfolder/etc` paths in your scripts is the enemy of reproducibility!

# Troubleshooting Git Credential Issues in RStudio

# Troubleshooting Git Credential Issues

One reason I'm not focusing on the GitHub + RStudio integration is **credential errors**.

For instance, 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()
```

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

DESCRIBE THE TOKEN'S USE CASE

What's this token for?

### Expiration \*

No expiration

The token will never expire!

GitHub strongly recommends that you set an expiration date for your token to help keep your information secure.

[Learn more](#)

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

repo

Full control of private repositories

repo:status

Access commit status

# 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

1. [Prologue](#)
2. [R Markdown](#)
3. [Version Control](#)
4. [GitHub Desktop](#)
5. [Other Tips and Productivity Tools](#)
6. [Not Covered: Git\(Hub\) + RStudio](#)
7. [Not Covered: Troubleshooting Git Credential Issues in RStudio](#)