INSTITUTE FOR DEFENSE ANALYSES

Introduction to Git

Dr. John T. Haman, Project Leader

Dr. Curtis G. Miller

April 2022

Public release approved. Distribution is unlimited.

IDA Document NS D-33021

Log: H 2022-000108

INSTITUTE FOR DEFENSE ANALYSES
730 East Glebe Road
Alexandria, Virginia 22305

The Institute for Defense Analyses is a nonprofit corporation that operates three Federally Funded Research and Development Centers. Its mission is to answer the most challenging U.S. security and science policy questions with objective analysis, leveraging extraordinary scientific, technical, and analytic expertise.

About This Publication
This work was conducted by the Institute for Defense Analyses (IDA) under contract HQ0034-19-D-0001, Task BD-09-229990, "TestSci App," for the Office of the Director, Operational Test and Evaluation. The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

Acknowledgments
The IDA Technical Review Committee was chaired by Dr. V. Bram Lillard and consisted of Dr. Matthew R. Avery, Dr. Juan D. Remolina Gonzalez, and Dr. Thomas H. Johnson from the Operational Evaluation Division.

For more information:
Dr. John T. Haman, Project Leader
jhaman@ida.org • (703) 845-2132

V. Bram Lillard, Director, Operational Evaluation Division
vlillard@ida.org • (703) 845-2230

Copyright Notice
© 2022 Institute for Defense Analyses
730 East Glebe Road, Alexandria, Virginia 22305 • (703) 845-2000

This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013 [Feb. 2014].

INSTITUTE FOR DEFENSE ANALYSES

IDA Document NS D-33021

Introduction to Git

Dr. John T. Haman, Project Leader

Dr. Curtis G. Miller

Executive Summary

Version control software manages, archives, and (optionally) distributes different versions of files. Git is the most popular program for version control and serves as the backbone for websites such as Github, Bitbucket, and others.

In this mini-tutorial we will introduce basics of version control in general, and Git in particular. We focus on two aspects of version control: maintaining a history of a project, and maintaining multiple simultaneous versions. Effective history management keeps a project clean and means that previous versions can be stored should they prove useful while formalizing the history-management process. Maintaining simultaneous file versions facilitates collaboration and the creation of similar products.

Participants will be able to get started using Git right away. The tutorial covers the basics, including creating a repository, storing it externally on GitHub, tracking changes via commits, sharing those changes, getting others' changes, recovering old versions of a project, branching a project into multiple (simultaneous) versions, and merging versions together.

We show how to use Git from the command line and also how online tools provided by GitHub can help with the process and understanding a repository's features. We also describe some Git and reproducible research best practices and demonstrate them through a notional project.

We also explain what role Git plays in a reproducible research context. Git itself will not make research reproducible, but by systematically tracking the history of a project, Git shows the relationship between chosen analytical methods, data used, and the results and conclusions. All of these factors often change as a project evolves over time, and Git tracks that evolution.

Introduction to Git

Curtis Miller

April 27, 2022

Institute for Defense Analyses
730 East Glebe Road ● Alexandria, Virginia 22305

Introduction

We will see basic Git usage and concepts

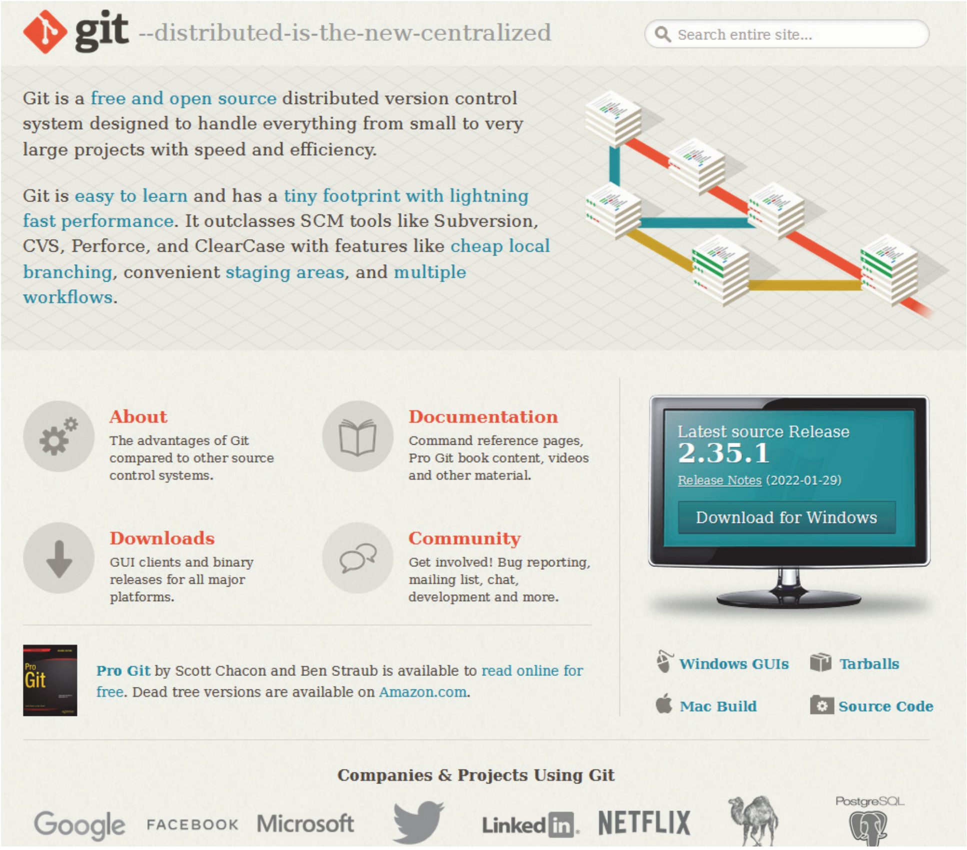
- Getting Started
- Repos
- Tracking Data
- Branches
- Conclusion

Git is a popular and powerful open-source, version control software system

- Version control software manages changes in files as files are edited over time and by different users
- With Git, users can track changes made to files and back up old versions of files
- When multiple variants of a product need to be managed or different people make changes, Git tracks the differences across branches

Getting Started

Git is free, open source, and the de facto version control system



git --distributed-is-the-new-centralized

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Latest source Release
2.35.1
Release Notes (2022-01-29)
Download for Windows

Windows GUIs
Tarballs
Mac Build
Source Code

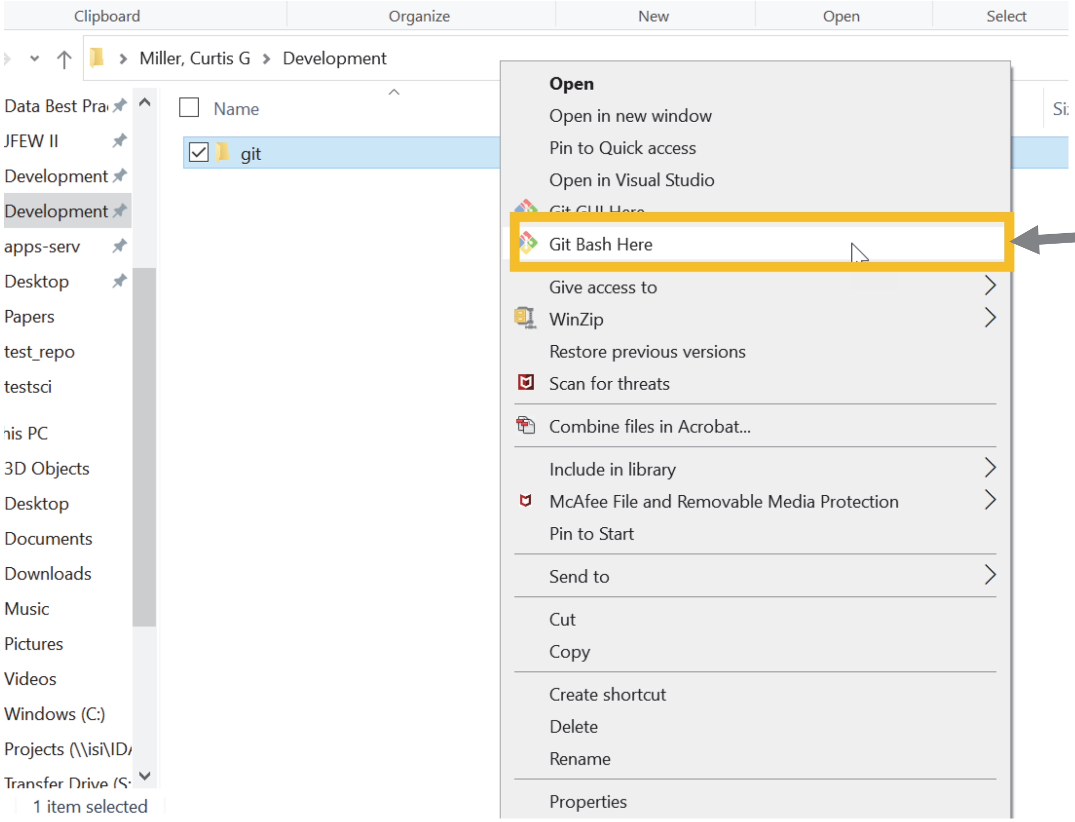
Companies & Projects Using Git

- Visit http://git-scm.com/ to download a copy of Git for your operating system (or use your operating system's package manager, if available)
- Follow the instructions of the installer to get set up
- We will be working in a Windows environment for this tutorial

I'm going to assume you have Git installed now.

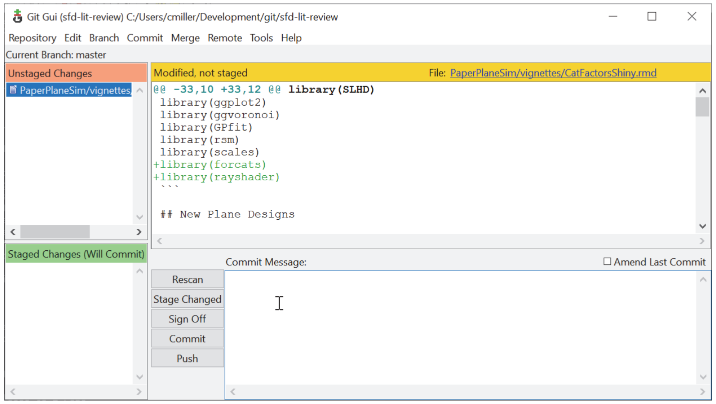
GPL: General Public License

You can open a Bash terminal to issue Git commands via a right-click on a directory on Windows

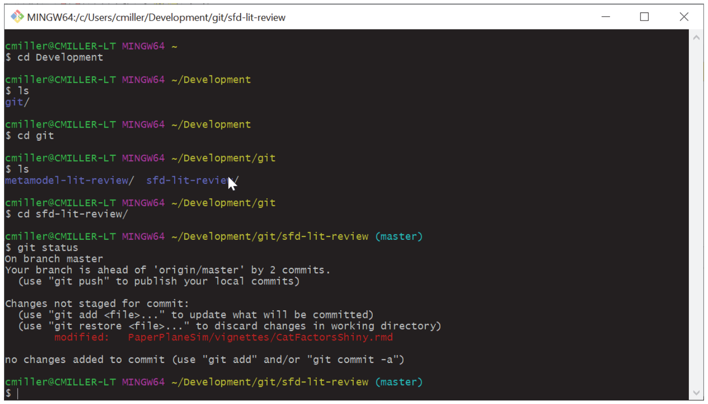

Clipboard
Organize
New
Open
Select
Miller, Curtis G
Development
Data Best Pra
JFEW II
Development
Development
apps-serv
Desktop
Papers
test_repo
testsci
his PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Projects (\\isi\ID
Transfer Drive (S:
1 item selected
Name
Si
git
Open
Open in new window
Pin to Quick access
Open in Visual Studio
Git GUI Here
Git Bash Here
Give access to
WinZip
Restore previous versions
Scan for threats
Combine files in Acrobat...
Include in library
McAfee File and Removable Media Protection
Pin to Start
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties

On Windows: click the folder, then right-click for this menu, then click Git Bash for a Bash terminal, or Git GUI for a graphical interface

Git on Windows provides a GUI and a Bash command line


Git Gui (sfd-lit-review) C:/Users/cmiller/Development/git/sfd-lit-review
Repository
Edit
Branch
Commit
Merge
Remote
Tools
Help
Current Branch: master
Unstaged Changes
PaperPlaneSim/vignettes/
Modified, not staged
File: PaperPlaneSim/vignettes/CatFactorsShiny.rmd
Staged Changes (Will Commit)
Commit Message:
☐ Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push

```
@@ -33,10 +33,12 @@ library(SLHD)
 library(ggplot2)
 library(ggvoronoi)
 library(GPfit)
 library(rsm)
 library(scales)
+library(forcats)
+library(rayshader)
 ```

 ## New Plane Designs
```


MINGW64:/c/Users/cmiller/Development/git/sfd-lit-review

```
cmiller@CMILLER-LT MINGW64 ~
$ cd Development

cmiller@CMILLER-LT MINGW64 ~/Development
$ ls
git/

cmiller@CMILLER-LT MINGW64 ~/Development
$ cd git

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ ls
metamodel-lit-review/  sfd-lit-review/

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ cd sfd-lit-review/

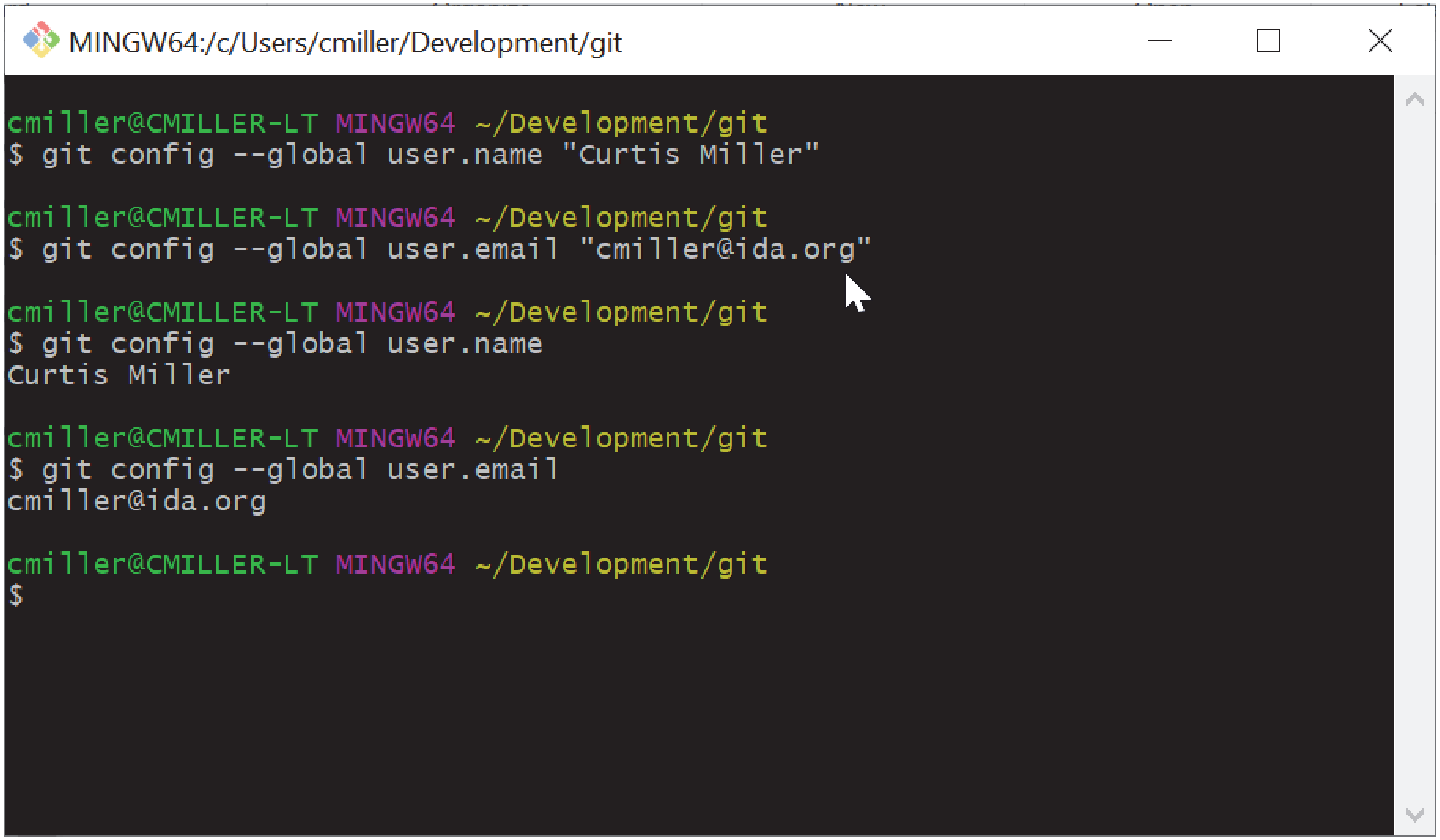
cmiller@CMILLER-LT MINGW64 ~/Development/git/sfd-lit-review (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   PaperPlaneSim/vignettes/CatFactorsShiny.rmd

no changes added to commit (use "git add" and/or "git commit -a")

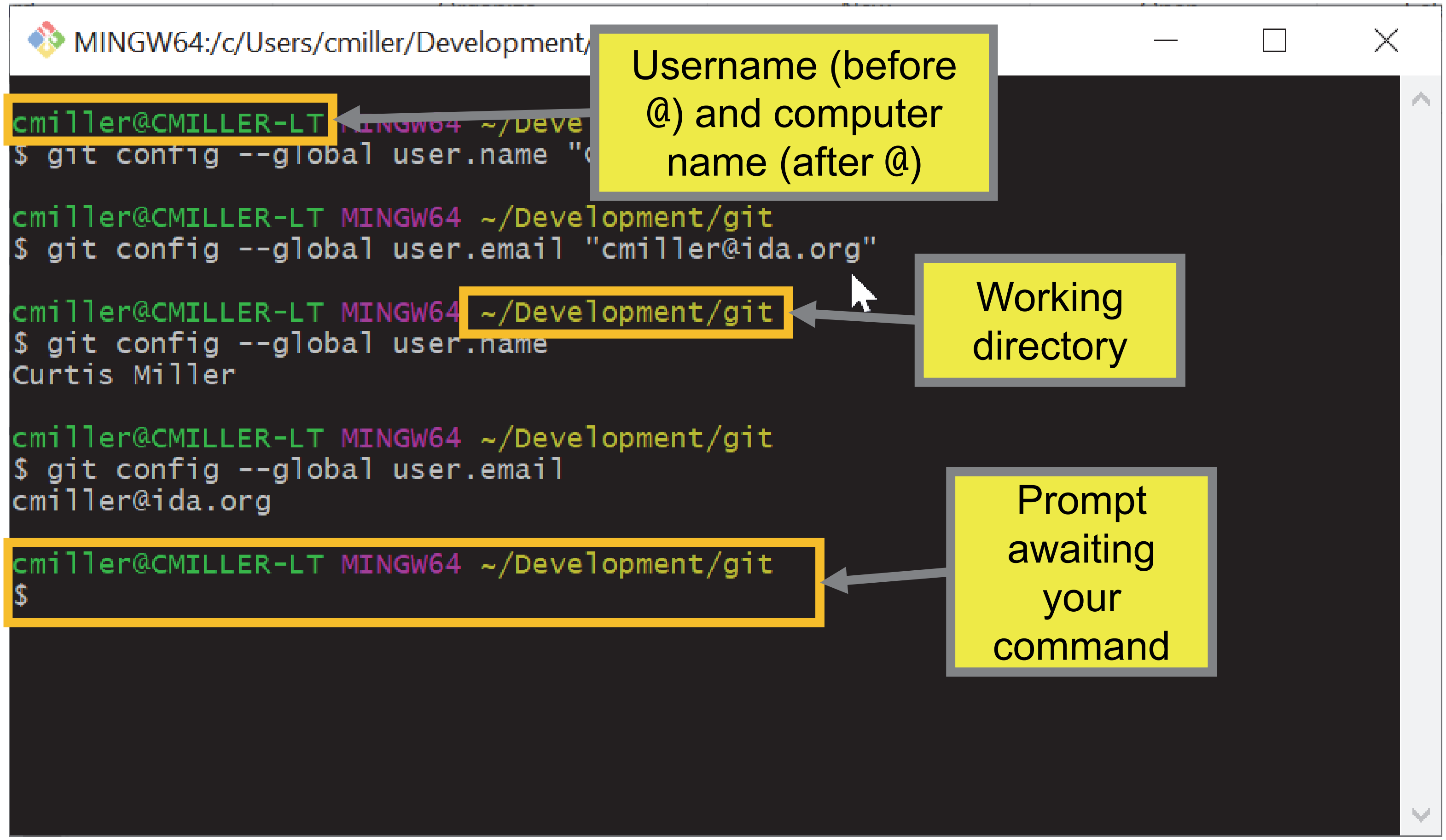
cmiller@CMILLER-LT MINGW64 ~/Development/git/sfd-lit-review (master)
$ 
```

- The Windows Git installer provides both a GUI and a Bash terminal emulator for using Git
- While the GUI may seem beginner friendly, it serves mostly as a front-end to the command line interface
- Full utilization of Git’s power is easiest when using the command line
- Git concepts are easier to explain when using the command line

GUI: Graphical User Interface
```

Set up initial user parameters using `git-config`



```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.name "Curtis Miller"

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email "cmiller@ida.org"

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.name
Curtis Miller

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email
cmiller@ida.org

cmiller@CMILLER-LT MINGW64 ~/Development/git
$
```

We can determine a lot from a Bash prompt



```
cmiller@CMILLER-LT MINGW64 ~/Deve
$ git config --global user.name "

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email "cmiller@ida.org"

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.name
Curtis Miller

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email
cmiller@ida.org

cmiller@CMILLER-LT MINGW64 ~/Development/git
$
```

Username (before @) and computer name (after @)

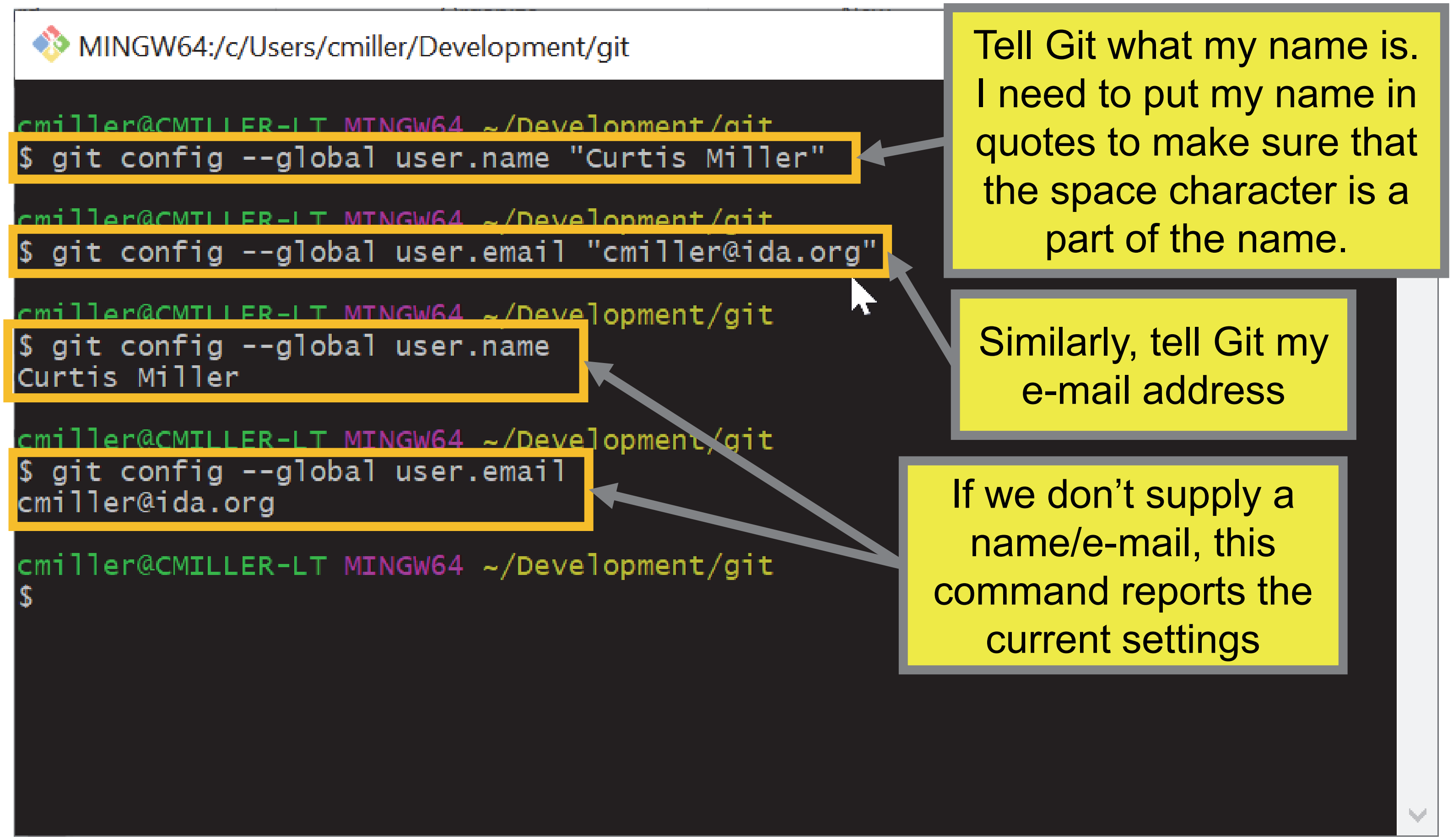
Working directory

Prompt awaiting your command

Bash is *not* Git; it is a shell (like PowerShell on Windows) and the default shell of Linux. We are talking about Git today, not Bash, though you may learn a little about Bash in the process.

We are using Git via Bash.

We can configure Git to know who we are; this information will be useful when others look at the Git history and see your work



MINGW64:/c/Users/cmiller/Development/git

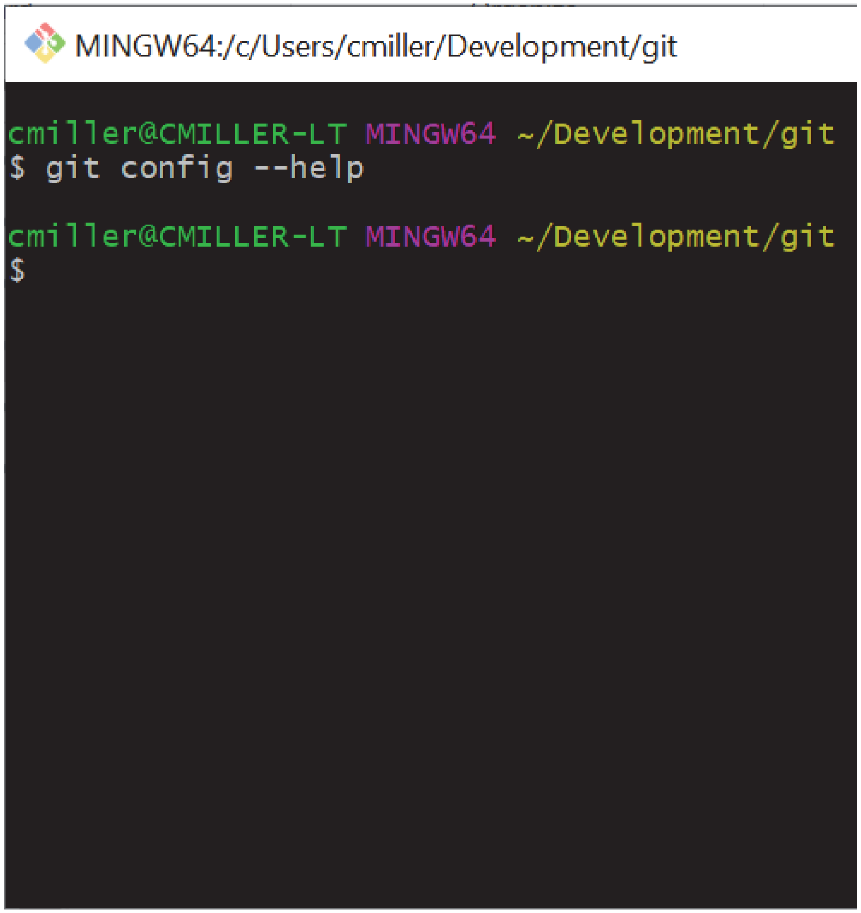
```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.name "Curtis Miller"

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email "cmiller@ida.org"

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.name
Curtis Miller

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --global user.email
cmiller@ida.org

cmiller@CMILLER-LT MINGW64 ~/Development/git
$
```

Tell Git what my name is. I need to put my name in quotes to make sure that the space character is a part of the name.

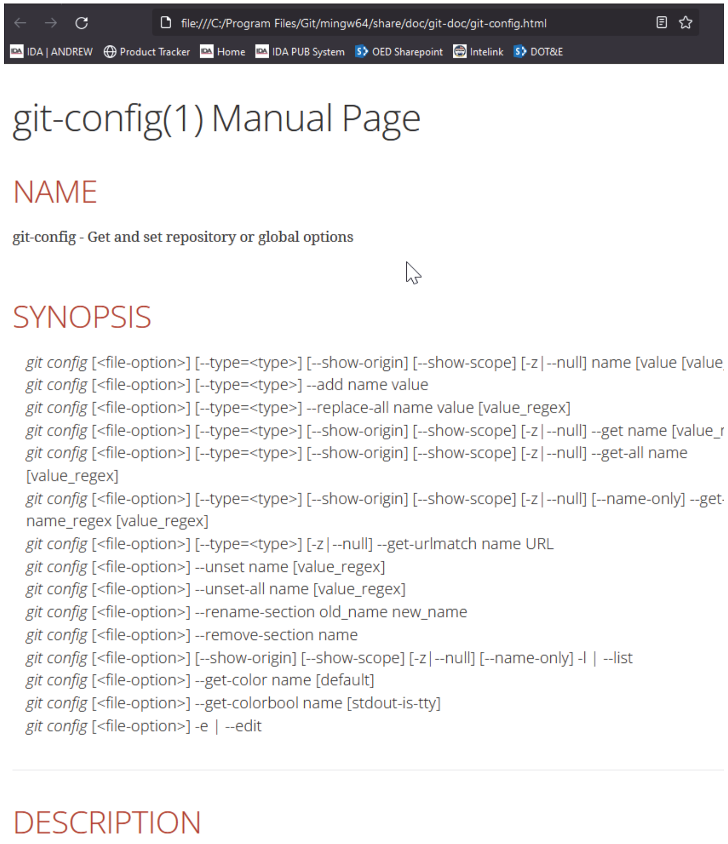
Similarly, tell Git my e-mail address

If we don’t supply a name/e-mail, this command reports the current settings

“How interesting! How can I learn more about how this command works?”


MINGW64:/c/Users/cmiller/Development/git

```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --help

cmiller@CMILLER-LT MINGW64 ~/Development/git
$
```


file:///C:/Program Files/Git/mingw64/share/doc/git-doc/git-config.html
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E

git-config(1) Manual Page

NAME

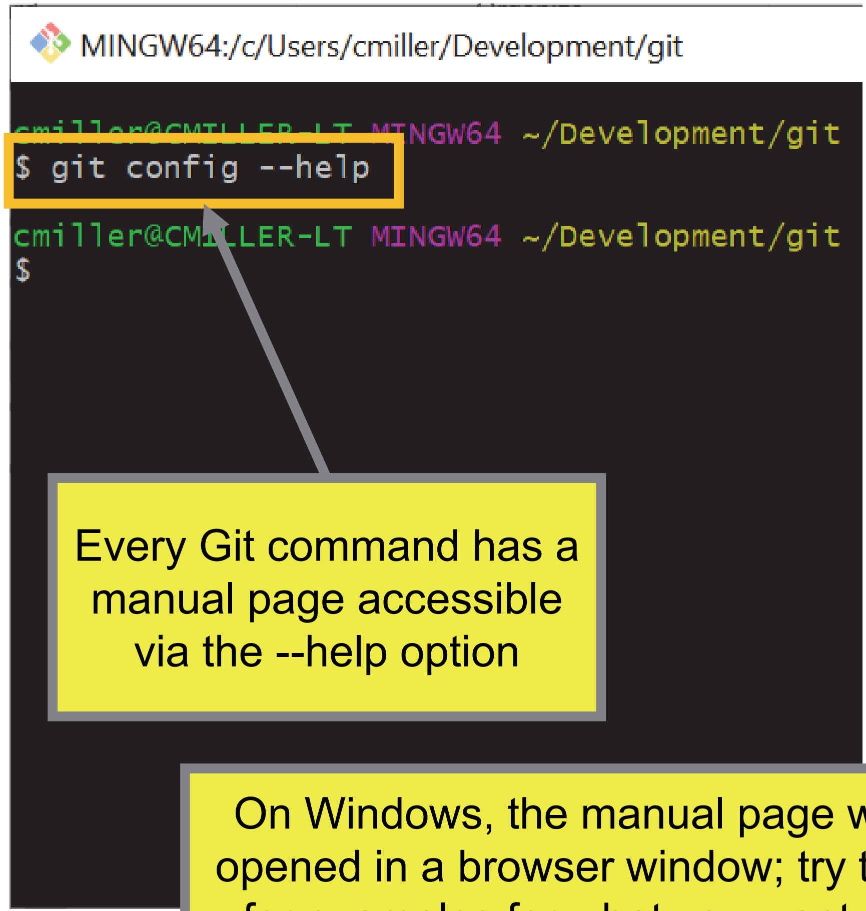
git-config - Get and set repository or global options

SYNOPSIS

```
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] name [value [value
git config [<file-option>] [--type=<type>] --add name value
git config [<file-option>] [--type=<type>] --replace-all name value [value_regex]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] --get name [value_r
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] --get-all name
[value_regex]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--name-only] --get-
name_regex [value_regex]
git config [<file-option>] [--type=<type>] [-z|--null] --get-urlmatch name URL
git config [<file-option>] --unset name [value_regex]
git config [<file-option>] --unset-all name [value_regex]
git config [<file-option>] --rename-section old_name new_name
git config [<file-option>] --remove-section name
git config [<file-option>] [--show-origin] [--show-scope] [-z|--null] [--name-only] -l | --list
git config [<file-option>] --get-color name [default]
git config [<file-option>] --get-colorbool name [stdout-is-tty]
git config [<file-option>] -e | --edit
```

DESCRIPTION

“How interesting! How can I learn more about how this command works?”

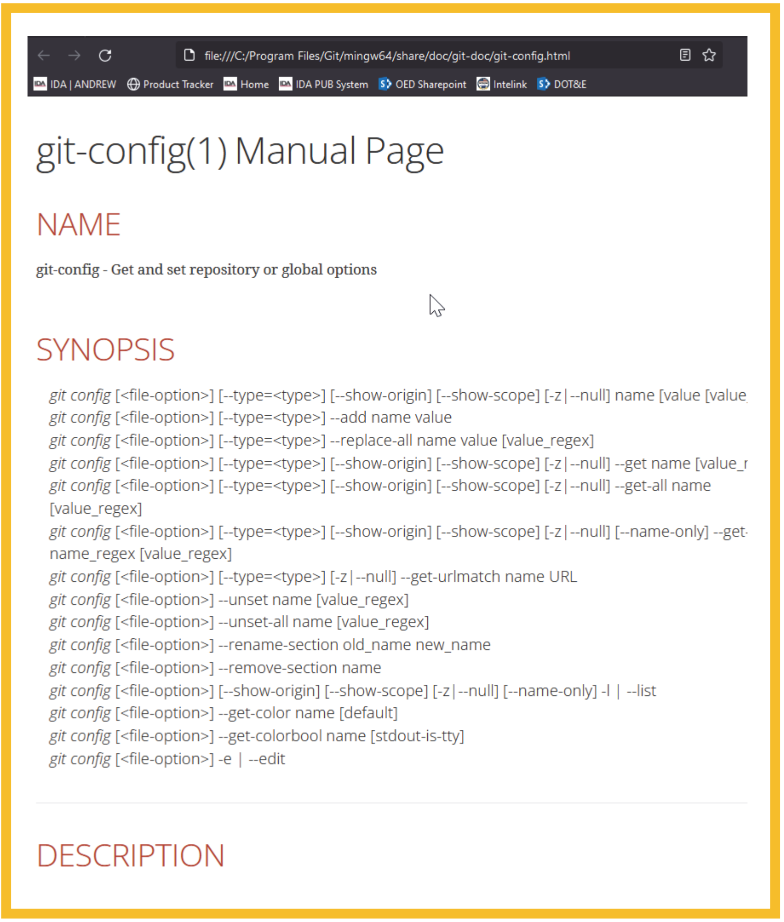


MINGW64:/c/Users/cmiller/Development/git

```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git config --help

cmiller@CMILLER-LT MINGW64 ~/Development/git
$
```

Every Git command has a manual page accessible via the --help option

On Windows, the manual page will be opened in a browser window; try to look for examples for what you want to do



file:///C:/Program Files/Git/mingw64/share/doc/git-doc/git-config.html
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E

git-config(1) Manual Page

NAME

git-config - Get and set repository or global options

SYNOPSIS

```
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] name [value [value
git config [<file-option>] [--type=<type>] --add name value
git config [<file-option>] [--type=<type>] --replace-all name value [value_regex]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] --get name [value_r
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] --get-all name
[value_regex]
git config [<file-option>] [--type=<type>] [--show-origin] [--show-scope] [-z|--null] [--name-only] --get-
name_regex [value_regex]
git config [<file-option>] [--type=<type>] [-z|--null] --get-urlmatch name URL
git config [<file-option>] --unset name [value_regex]
git config [<file-option>] --unset-all name [value_regex]
git config [<file-option>] --rename-section old_name new_name
git config [<file-option>] --remove-section name
git config [<file-option>] [--show-origin] [--show-scope] [-z|--null] [--name-only] -l | --list
git config [<file-option>] --get-color name [default]
git config [<file-option>] --get-colorbool name [stdout-is-tty]
git config [<file-option>] -e | --edit
```

DESCRIPTION

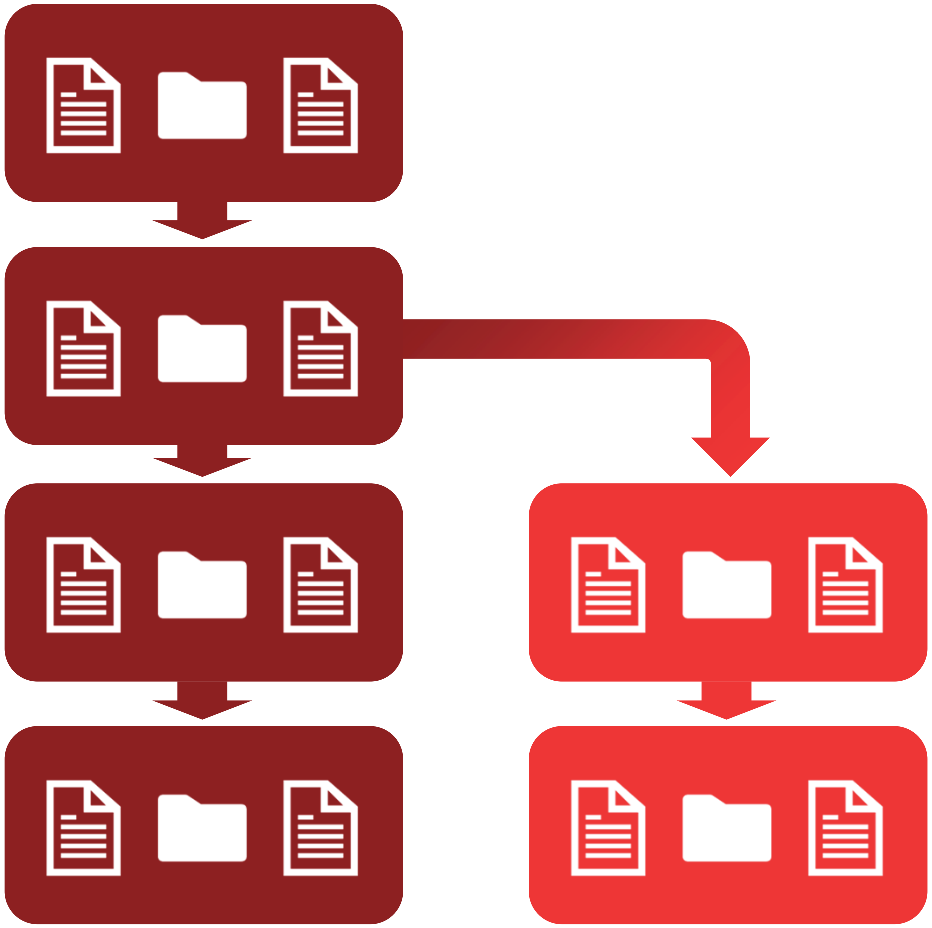
Git powers popular code sharing websites such as GitHub

git
GitHub
Bitbucket
GitLab

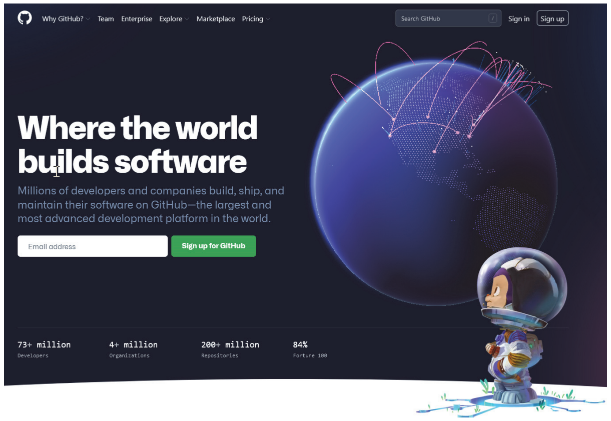
Repos

A Git repository is a collection of tracked files

- Git tracks the history of these files and their differences across time and between branches
- Repos generally contain entire projects
- Users access the project by accessing the repo, and the project is shared by sharing the repo
- Repos are as simple as folders containing files



Many web services host repos (but you don’t need to use these to use Git on your own computer)


Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search GitHub
Sign in
Sign up
Email address
Sign up for GitHub

Where the world builds software

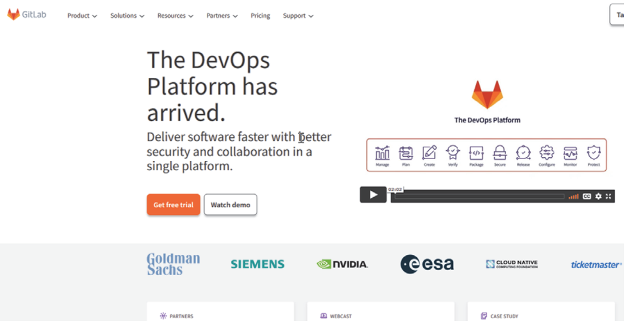
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

73+ million Developers

4+ million Organizations

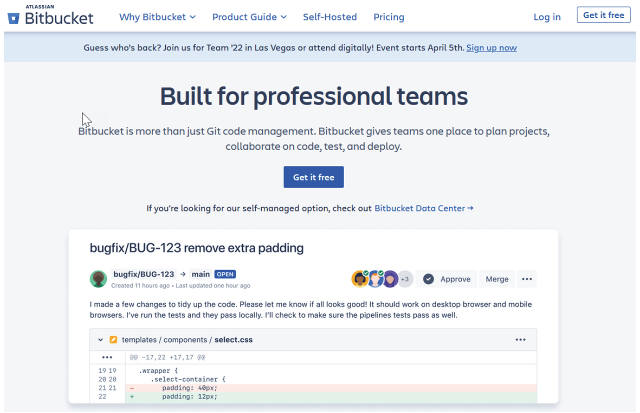
200+ million Repositories

84% Fortune 100


GitLab
Product
Solutions
Resources
Partners
Pricing
Support
Ta
The DevOps Platform
Manage
Plan
Create
Verify
Package
Secure
Release
Configure
Monitor
Protect
02:02
Get free trial
Watch demo
Goldman Sachs
SIEMENS
NVIDIA
esa
CLOUD NATIVE COMPUTING FOUNDATION
ticketmaster
PARTNERS
WEBCAST
CASE STUDY

The DevOps Platform has arrived.

Deliver software faster with better security and collaboration in a single platform.


ATLASSIAN
Bitbucket
Why Bitbucket
Product Guide
Self-Hosted
Pricing
Log in
Get it free
Get it free
OPEN
+3
Approve
Merge

Guess who’s back? Join us for Team ’22 in Las Vegas or attend digitally! Event starts April 5th. Sign up now

Built for professional teams

Bitbucket is more than just Git code management. Bitbucket gives teams one place to plan projects, collaborate on code, test, and deploy.

If you're looking for our self-managed option, check out Bitbucket Data Center →

bugfix/BUG-123 remove extra padding

bugfix/BUG-123 → main

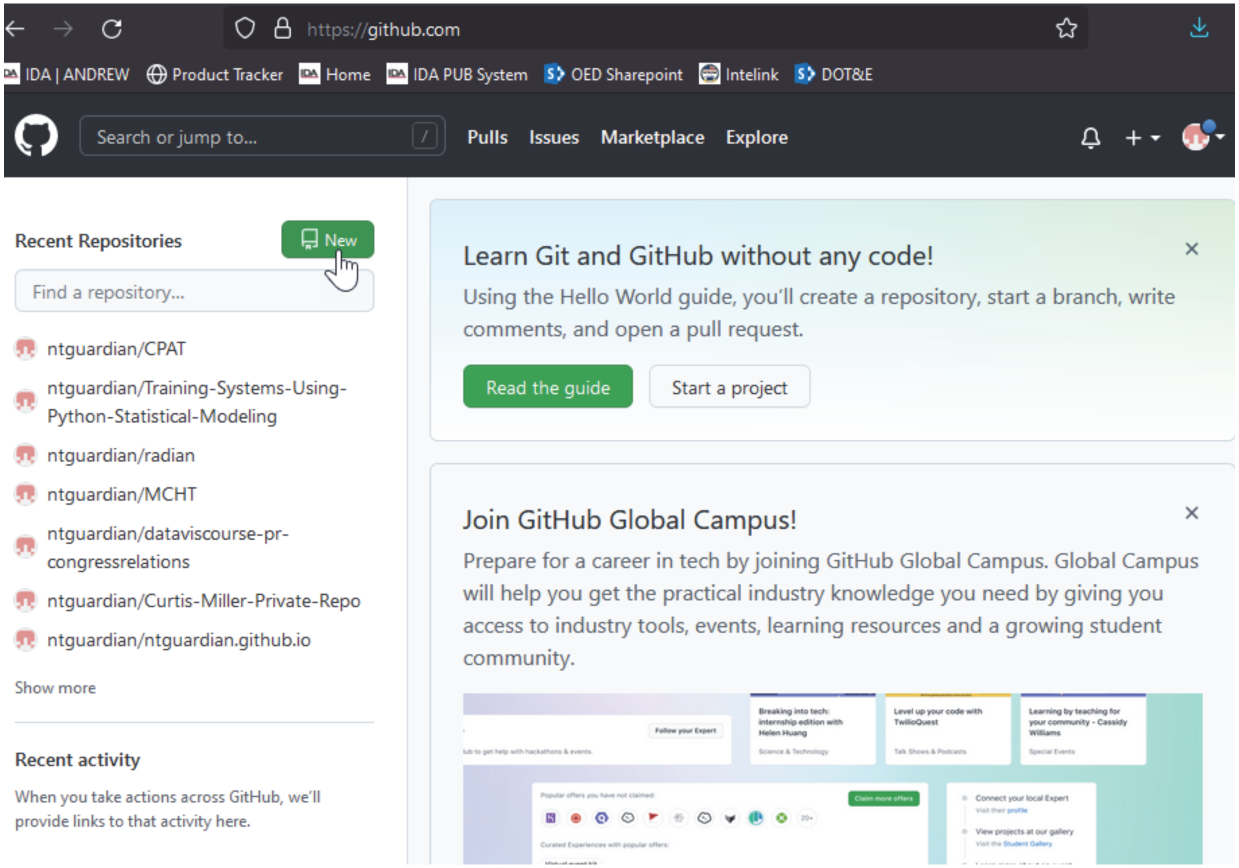
Created 11 hours ago • Last updated one hour ago

I made a few changes to tidy up the code. Please let me know if all looks good! It should work on desktop browser and mobile browsers. I’ve run the tests and they pass locally. I’ll check to make sure the pipelines tests pass as well.

templates / components / select.css

```
@@ -17,22 +17,17 @@
  .wrapper {
    .select-container {
-      padding: 40px;
+      padding: 12px;
```

We can create a repo on GitHub and then clone it to a personal computer


https://github.com
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E
Search or jump to...
Pulls
Issues
Marketplace
Explore
Recent Repositories
New
Find a repository...
Read the guide
Start a project
Show more

ntguardian/CPAT
ntguardian/Training-Systems-Using-Python-Statistical-Modeling
ntguardian/radian
ntguardian/MCHT
ntguardian/dataviscourse-pr-congressrelations
ntguardian/Curtis-Miller-Private-Repo
ntguardian/ntguardian.github.io

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

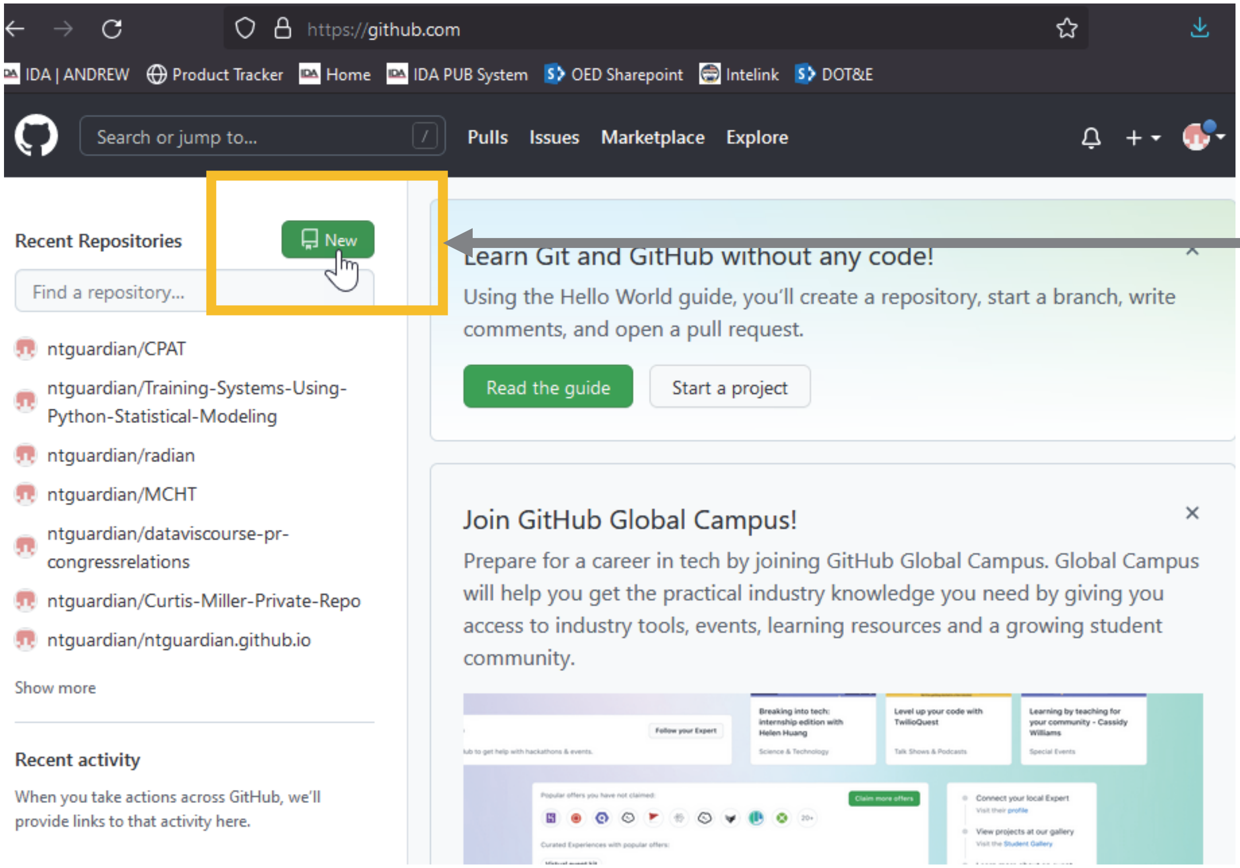
Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Join GitHub Global Campus!

Prepare for a career in tech by joining GitHub Global Campus. Global Campus will help you get the practical industry knowledge you need by giving you access to industry tools, events, learning resources and a growing student community.

We can create a repo on GitHub and then clone it to a personal computer


https://github.com
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E
Search or jump to...
Pulls
Issues
Marketplace
Explore
New
Read the guide
Start a project

Recent Repositories

Find a repository...

- ntguardian/CPAT
- ntguardian/Training-Systems-Using-Python-Statistical-Modeling
- ntguardian/radian
- ntguardian/MCHT
- ntguardian/dataviscourse-pr-congressrelations
- ntguardian/Curtis-Miller-Private-Repo
- ntguardian/ntguardian.github.io

Show more

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Learn Git and GitHub without any code!

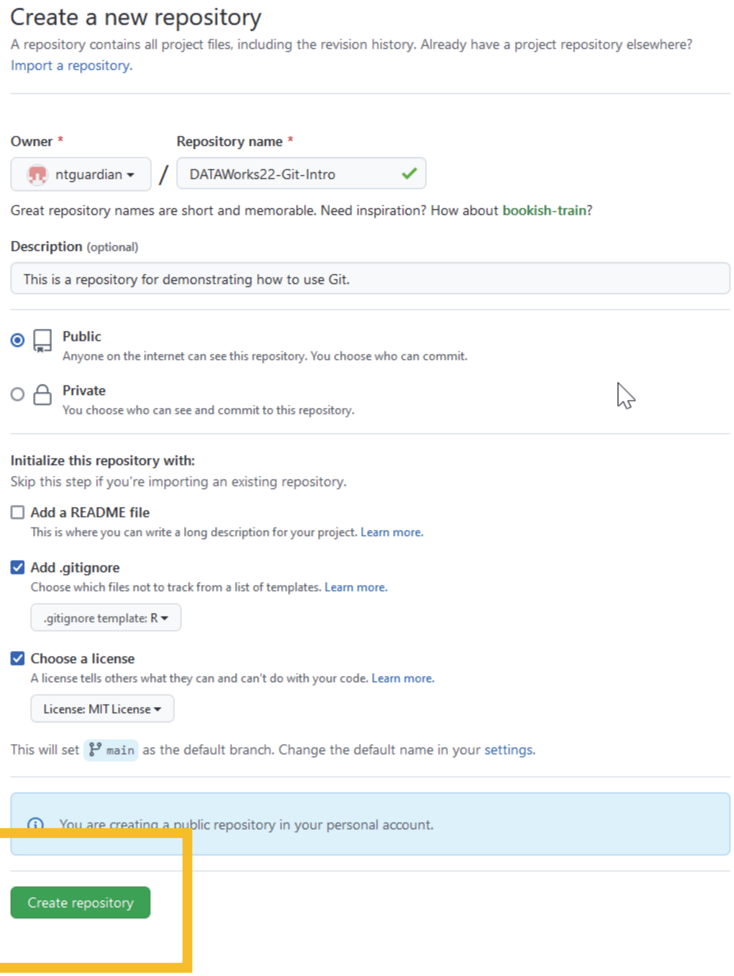
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Join GitHub Global Campus!

Prepare for a career in tech by joining GitHub Global Campus. Global Campus will help you get the practical industry knowledge you need by giving you access to industry tools, events, learning resources and a growing student community.

Click me to make a new repo on GitHub

Fill out the form to create a new repository



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * ntguardian / Repository name * DATAWorks22-Git-Intro ✓

Great repository names are short and memorable. Need inspiration? How about bookish-train?

Description (optional): This is a repository for demonstrating how to use Git.

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

☑ Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: R

☑ Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: MIT License

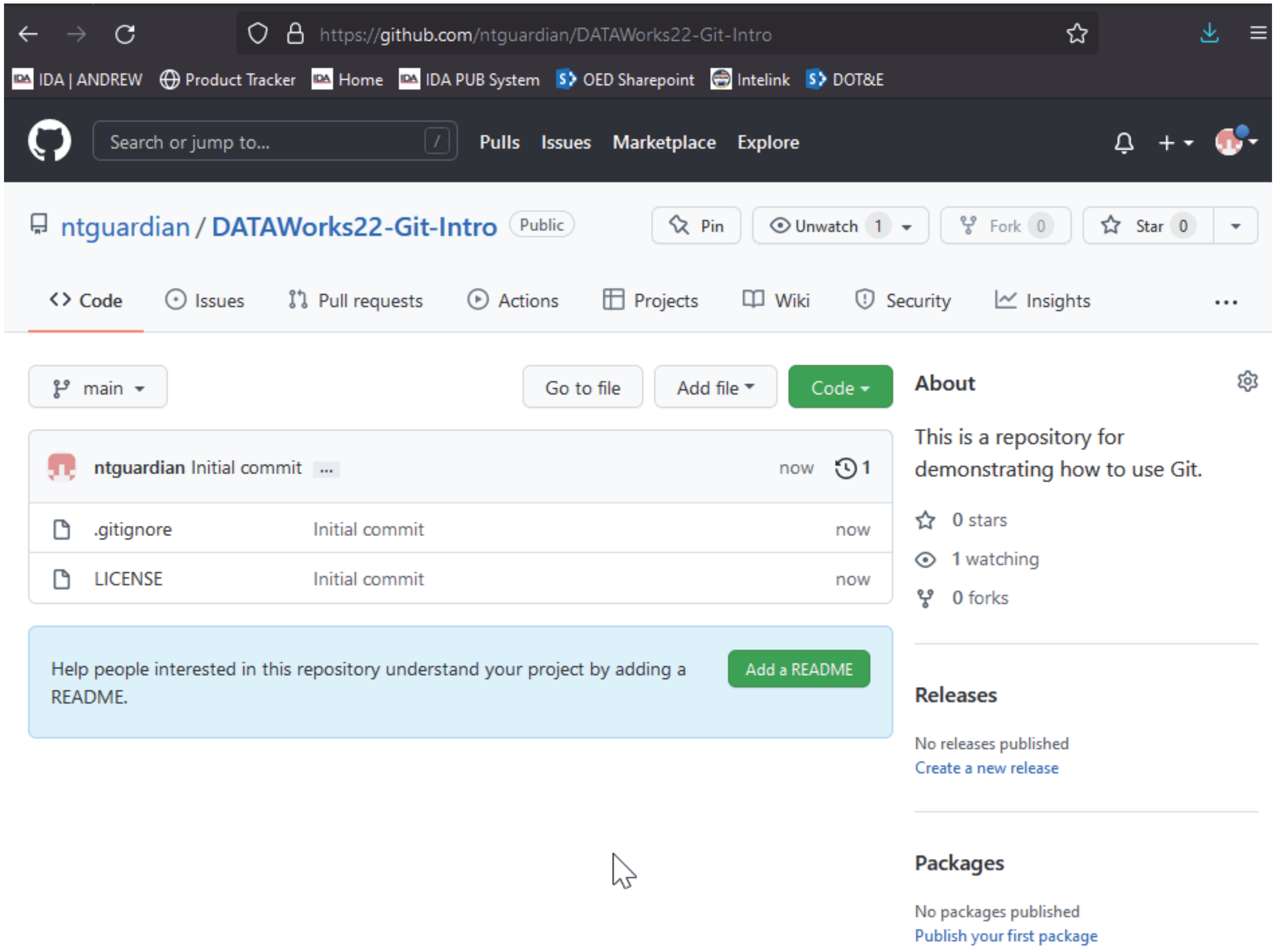
This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

Click me to make a new repo on GitHub

For real work, follow your organization guidelines on where to store data. For example, IDA employees need to use the IDA internal BitBucket for repo hosting, not GitHub (for sponsor work).

Now we can view our repo online



https://github.com/ntguardian/DATAWorks22-Git-Intro
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E
Search or jump to...
Pulls
Issues
Marketplace
Explore
ntguardian / DATAWorks22-Git-Intro
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
main
Go to file
Add file
Code
Add a README

ntguardian Initial commit ...		now 1
.gitignore	Initial commit	now
LICENSE	Initial commit	now

Help people interested in this repository understand your project by adding a README.

About

This is a repository for demonstrating how to use Git.

0 stars

1 watching

0 forks

Releases

No releases published

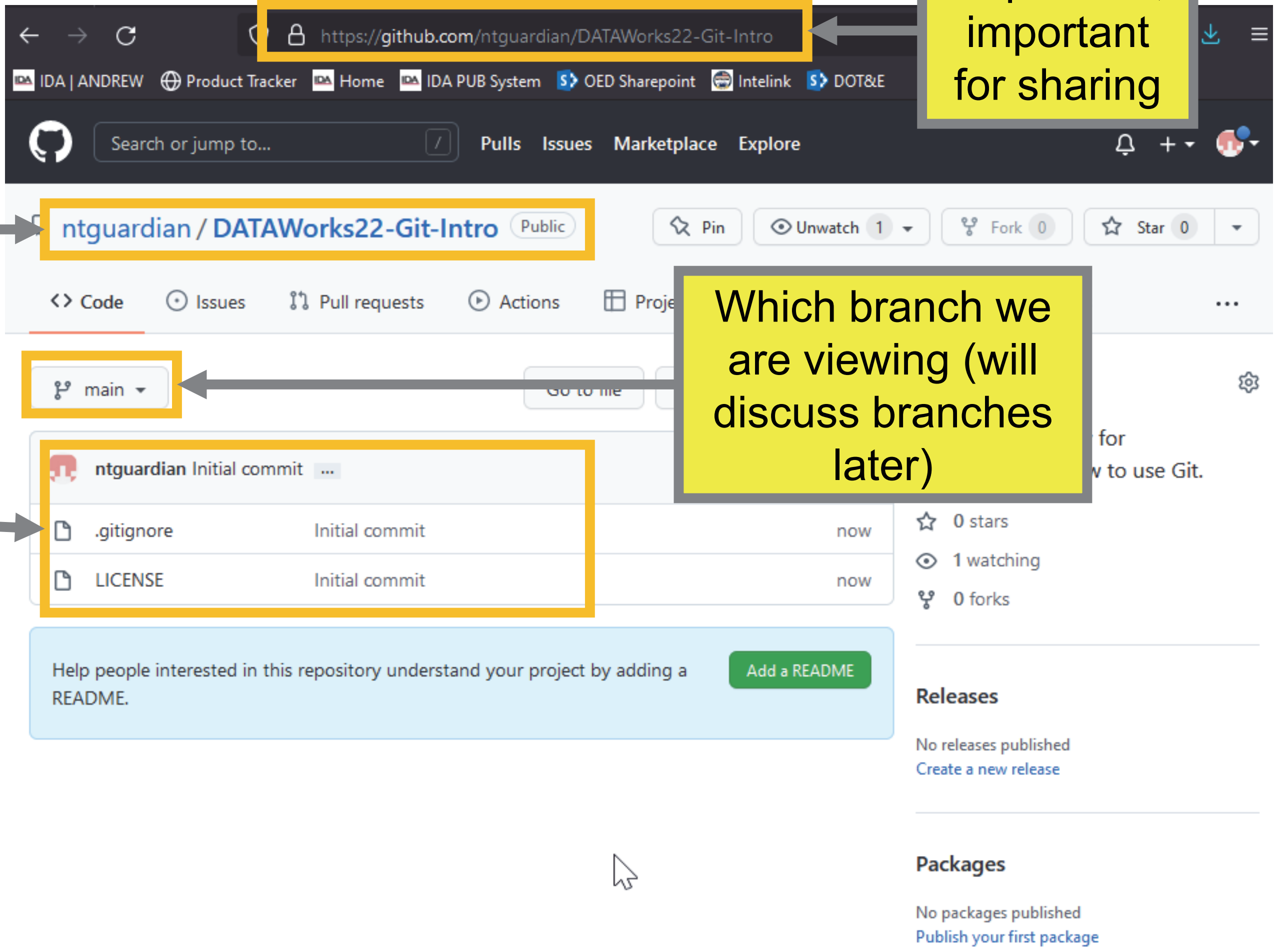
Create a new release

Packages

No packages published

Publish your first package

Now we can view our repo online


https://github.com/ntguardian/DATAWorks22-Git-Intro
IDA | ANDREW
Product Tracker
Home
IDA PUB System
OED Sharepoint
Intelink
DOT&E
Search or jump to...
Pulls
Issues
Marketplace
Explore
ntguardian / DATAWorks22-Git-Intro
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Proje
main
Go to file
for
to use Git.
0 stars
1 watching
0 forks
Add a README
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package

ntguardian Initial commit

.gitignore	Initial commit	now
LICENSE	Initial commit	now

Help people interested in this repository understand your project by adding a README.

Repo URL, important for sharing

Repo identifier (note connection to URL)

Which branch we are viewing (will discuss branches later)

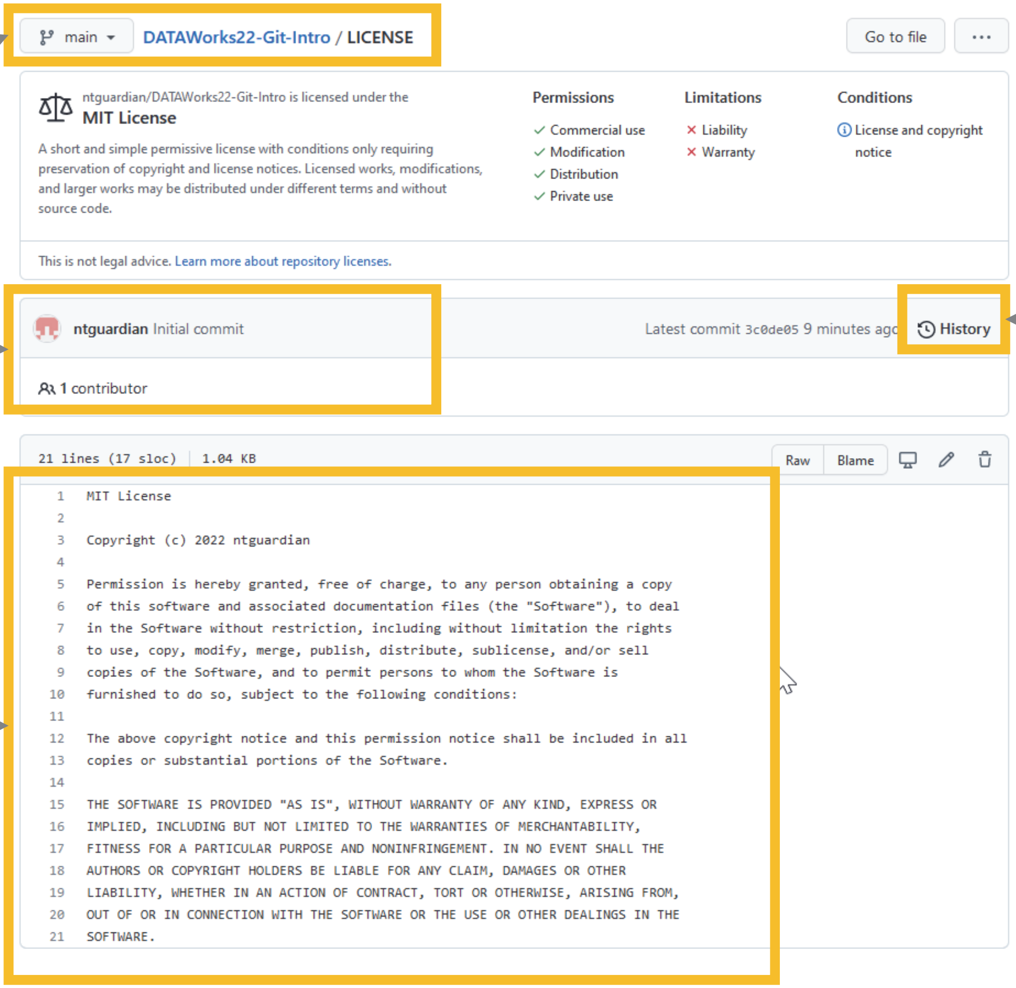
Some files already in the repo, created by GitHub (click to view)

We can view files online

The file we are viewing, including what branch

Who works on the file and recent activity

File contents



ntguardian/DATAWorks22-Git-Intro is licensed under the
MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Limitations	Conditions
✓ Commercial use	× Liability	License and copyright notice
✓ Modification	× Warranty	
✓ Distribution		
✓ Private use		

This is not legal advice. Learn more about repository licenses.

ntguardian Initial commit

Latest commit 3c0de05 9 minutes ago

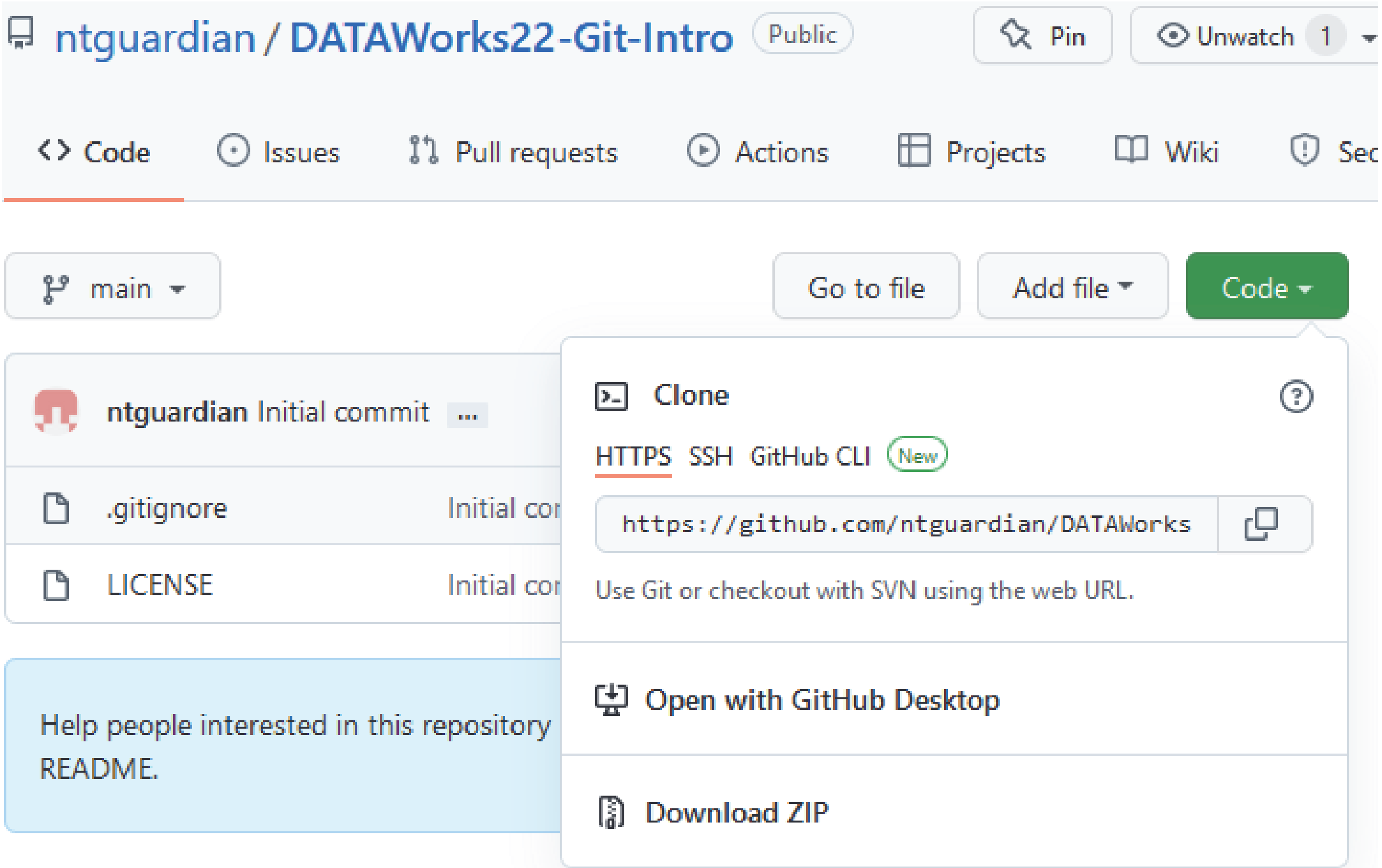
1 contributor

21 lines (17 sloc) | 1.04 KB

```
MIT License

Copyright (c) 2022 ntguardian

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Click this to see the file's commit history (will discuss commits soon)

To get this repo onto our computer, we need to clone it

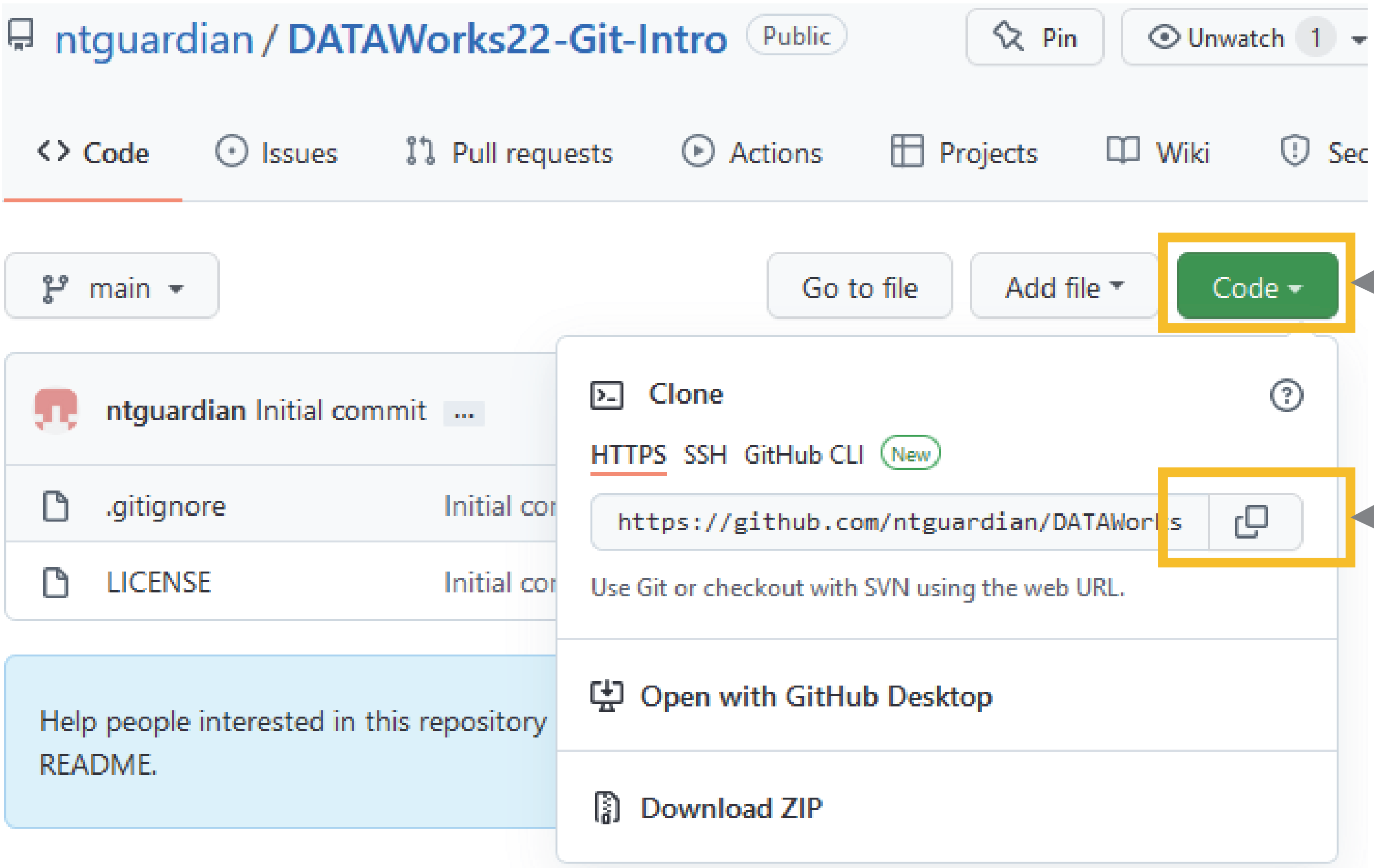

ntguardian / DATAWorks22-Git-Intro
Public
Pin
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Sec
main
Go to file
Add file
Code
ntguardian Initial commit
.gitignore
Initial co
LICENSE
Initial co
Clone
HTTPS
SSH
GitHub CLI
New
Open with GitHub Desktop
Download ZIP

https://github.com/ntguardian/DATAWorks

Use Git or checkout with SVN using the web URL.

Help people interested in this repository
README.

To get this repo onto our computer, we need to clone it


ntguardian / DATAWorks22-Git-Intro
Public
Pin
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Sec
main
Go to file
Add file
Code
ntguardian Initial commit ...
.gitignore
Initial cor
LICENSE
Initial cor
Clone
HTTPS
SSH
GitHub CLI
New
Open with GitHub Desktop
Download ZIP

`https://github.com/ntguardian/DATAWorl s`

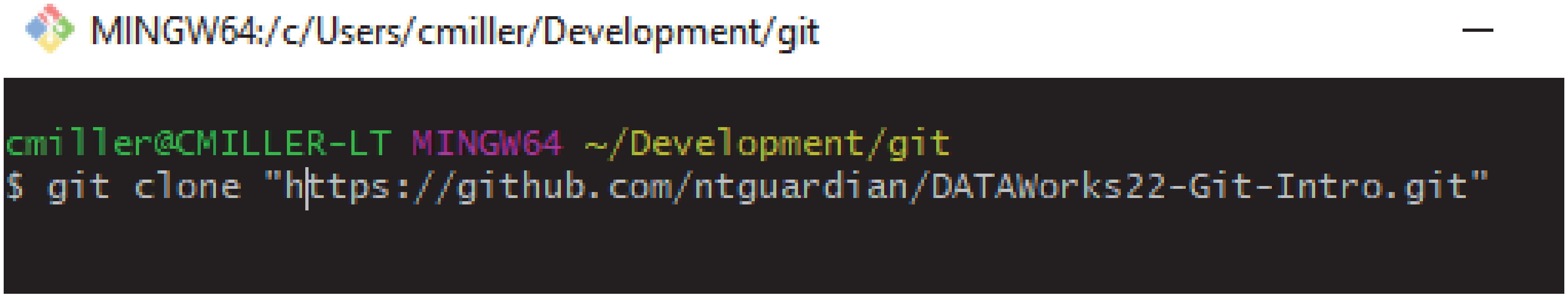
Use Git or checkout with SVN using the web URL.

Help people interested in this repository
README.

Click this to open a dialog to allow easy cloning

Click to copy the repo URL for cloning

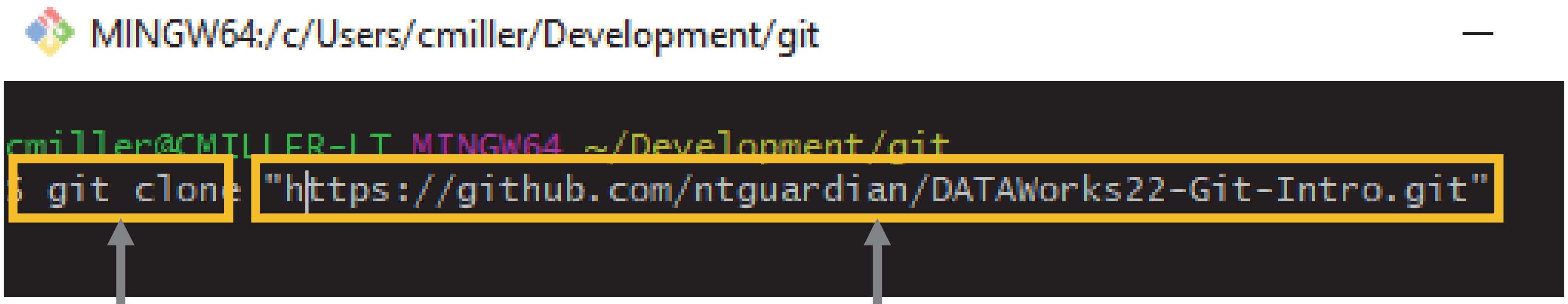
Use the command git clone to get a copy on your computer



MINGW64:/c/Users/cmiller/Development/git

```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git clone "https://github.com/ntguardian/DATAWorks22-Git-Intro.git"
```

Use the command git clone to get a copy on your computer



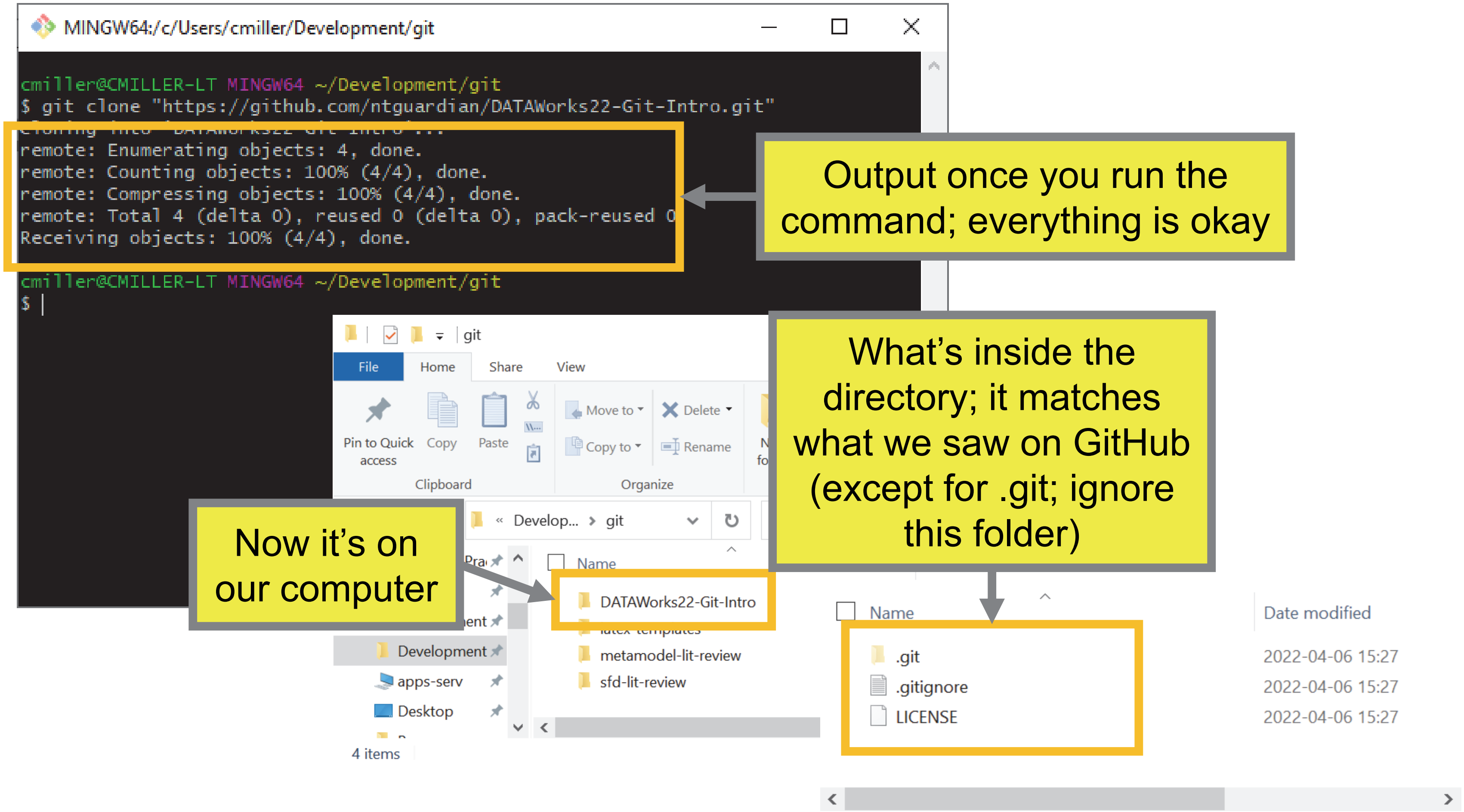
MINGW64:/c/Users/cmiller/Development/git

```
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git clone "https://github.com/ntguardian/DATAWorks22-Git-Intro.git"
```

The `git clone` command

The URL you copied; wrap in quotes to be safe

Once you run the command, you have a working copy of the repo on your computer



```
MINGW64:/c/Users/cmiller/Development/git

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git clone "https://github.com/ntguardian/DATAWorks22-Git-Intro.git"
Cloning into 'DATAWorks22-Git-Intro'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

cmiller@CMILLER-LT MINGW64 ~/Development/git
$ |
```

git
File
Home
Share
View
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
Clipboard
Organize
« Develop... › git
Development
apps-serv
Desktop
Name

- DATAWorks22-Git-Intro
- latex-templates
- metamodel-lit-review
- sfd-lit-review

4 items

Name	Date modified
.git	2022-04-06 15:27
.gitignore	2022-04-06 15:27
LICENSE	2022-04-06 15:27

Output once you run the command; everything is okay

What's inside the directory; it matches what we saw on GitHub (except for .git; ignore this folder)

Now it's on our computer

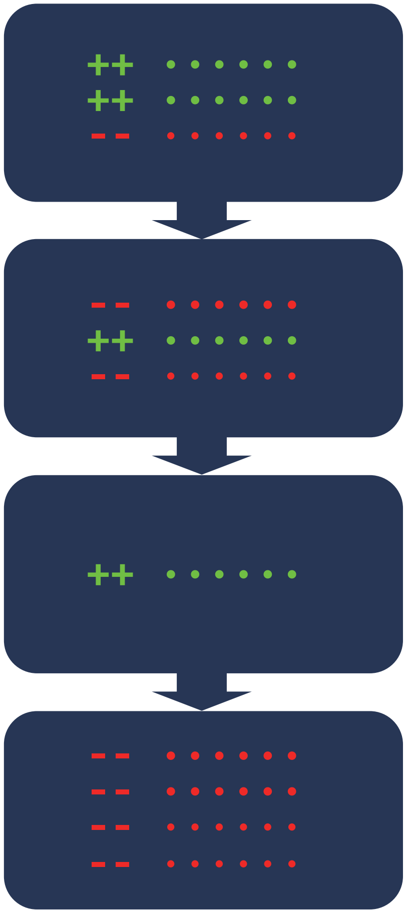
Summary: Making and getting repos

Command	Description	Examples
`git init`	Create a new repo	`git init`
`git clone`	Clone an existing repository	`git clone “https://github.com/someuser/someproject”`

Tracking Data

Git tracks changes via commits

- A Git commit is the basic unit by which Git tracks changes to tracked files
- The Git user adds files for Git to track and tells Git when to track changes made to a file
- For text files, Git tracks only the difference between the current file and the most recent version
- Users can “check out” previous commits to recover the (tracked) state of the project at the time of the commit



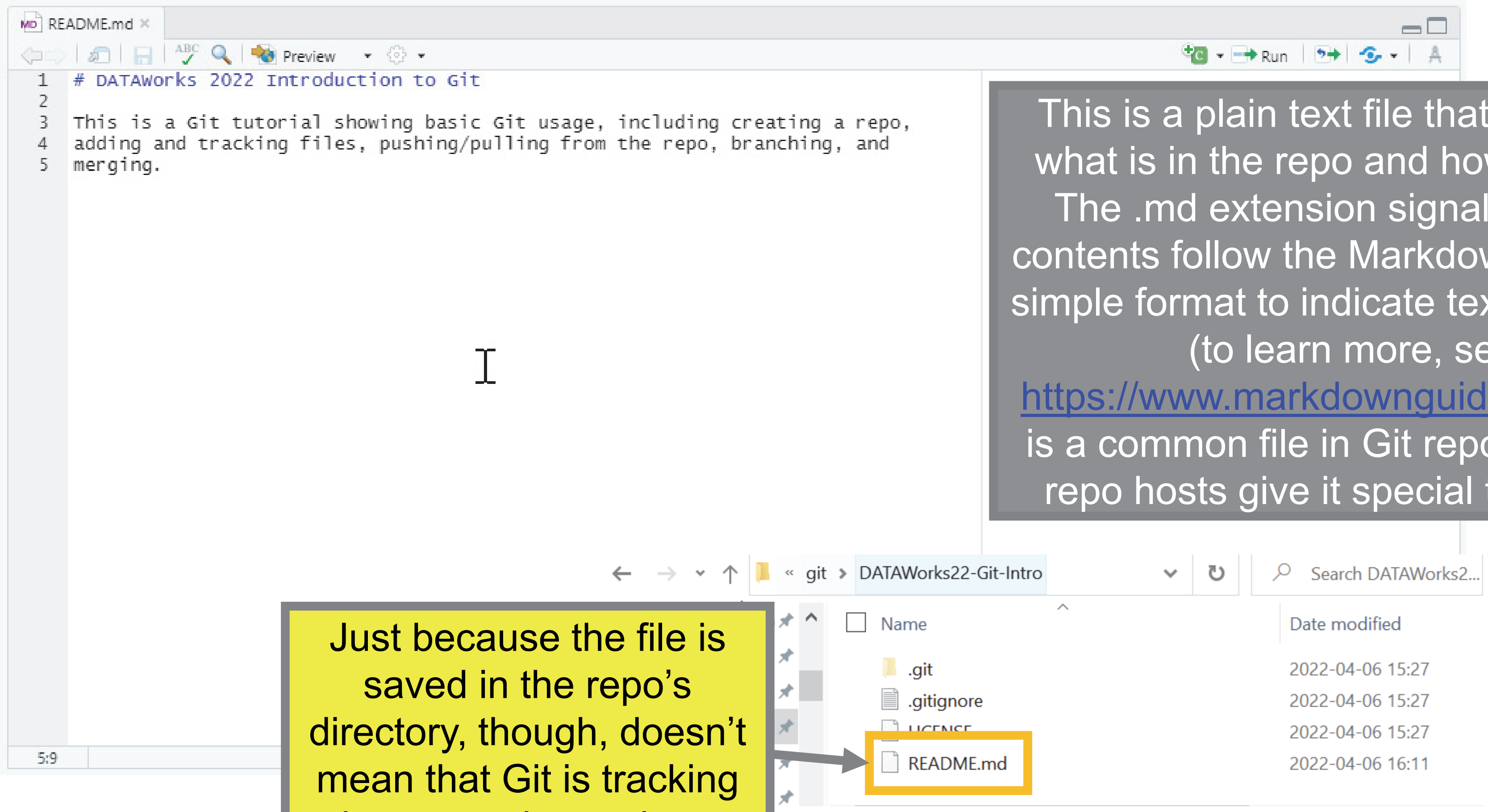
Using Git is like belaying* while mountain climbing; if you fall, you fall three feet, not three hundred feet

- With Git, you can track changes as they are made
- If you make an immediate mistake, it can be undone
- If you made a mistake several weeks ago that is not obvious until now, Git can help find where the error happened and help address it
- If a deleted file turns out to be useful, you can retrieve it
- You can branch to try out an idea without losing the original files known to work
- You can easily manage multiple versions of a product simultaneously

Wikimedia Commons

*Belaying is climbing a rock wall with a rope. The climber is attached to a rope that is either held by someone at the top of the cliff or looped around something sturdy while a partner stands on the ground with the other end of the rope. As the climber progresses, their partner keeps the rope taut; should the climber let go of the wall, they will only fall as far as the slack in the rope allows, which keeps the climber safe and ensures not too much progress is lost.

Our Git repo should have a README.md file; let’s make one!



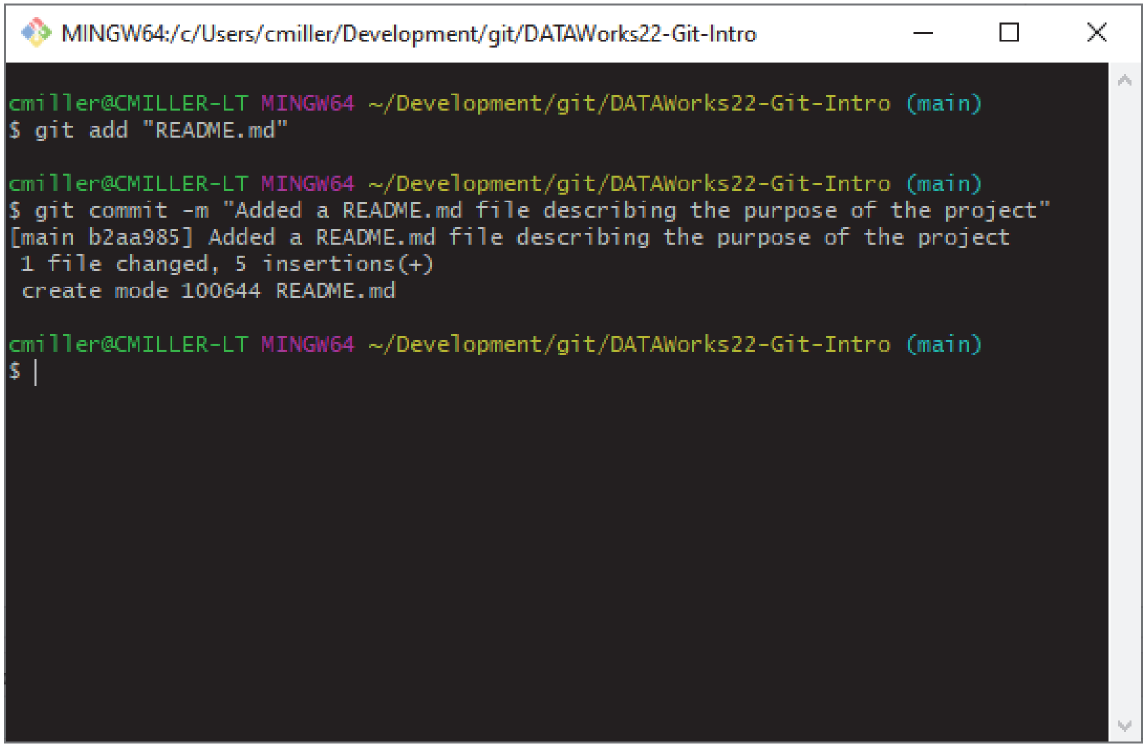
```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.
```

Name	Date modified
.git	2022-04-06 15:27
.gitignore	2022-04-06 15:27
LICENSE	2022-04-06 15:27
README.md	2022-04-06 16:11

This is a plain text file that describes what is in the repo and how to use it. The .md extension signals that the contents follow the Markdown format, a simple format to indicate text formatting (to learn more, see https://www.markdownguide.org). This is a common file in Git repos so some repo hosts give it special treatment.

Just because the file is saved in the repo’s directory, though, doesn’t mean that Git is tracking it; we need to make a commit that adds it.

Use `git commit` to create a commit that adds the file and describes its contents



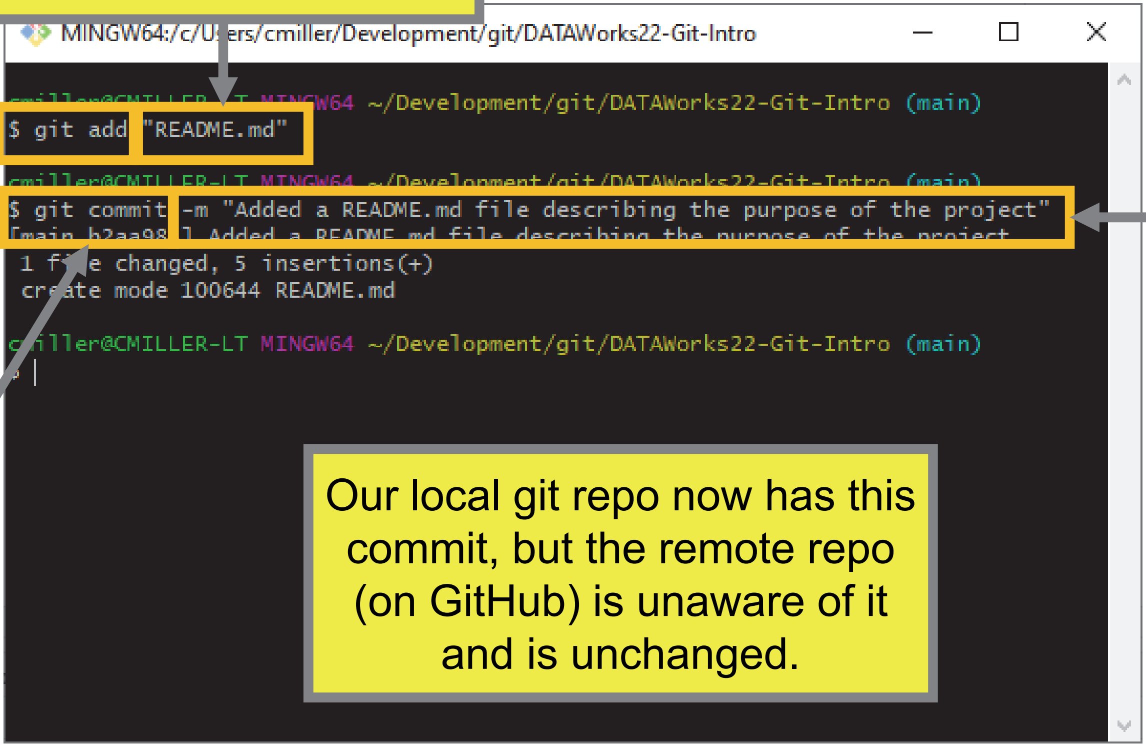
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add "README.md"

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Added a README.md file describing the purpose of the project"
[main b2aa985] Added a README.md file describing the purpose of the project
 1 file changed, 5 insertions(+)
 create mode 100644 README.md

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ 
```

Use `git commit` to create a commit that adds the file and describes its contents

`git add` is the command for adding changes to a commit to track

Name the file with the changes to track



MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add "README.md"

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Added a README.md file describing the purpose of the project"
[main b2aa98 ] Added a README.md file describing the purpose of the project
 1 file changed, 5 insertions(+)
 create mode 100644 README.md

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ |
```

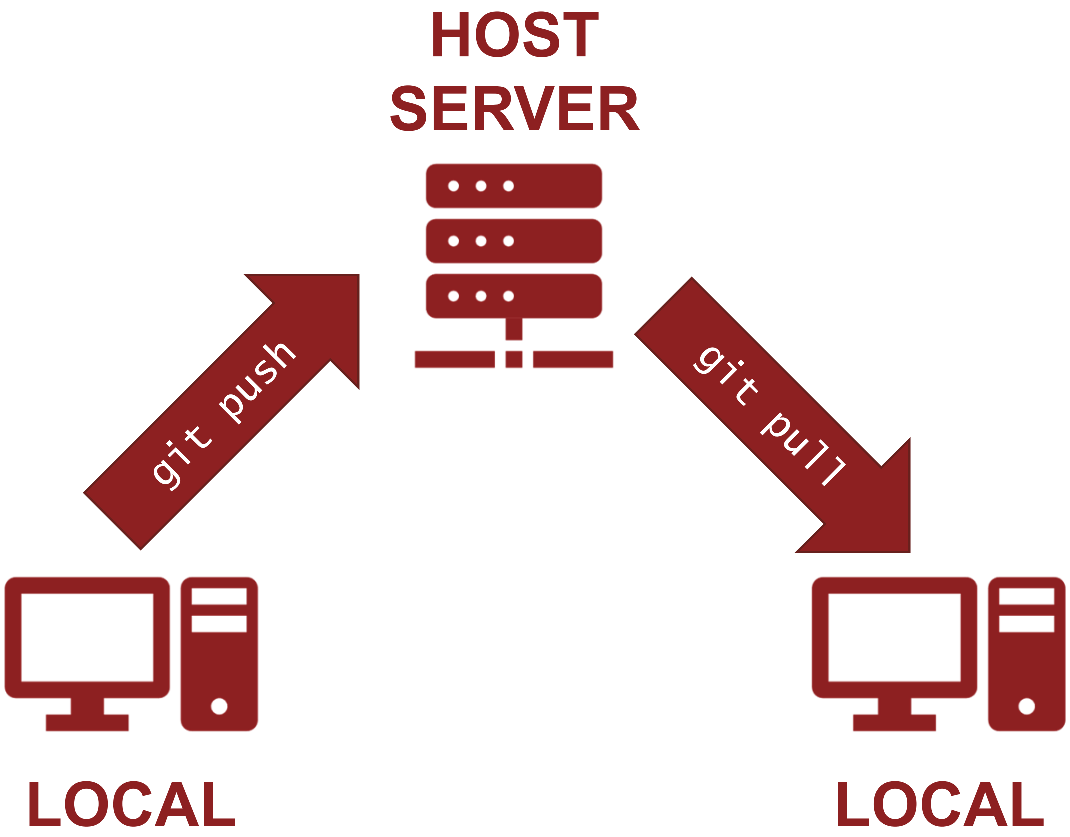
The `-m` option is for adding a message summarizing what changes were made in the commit (keep it short)

`git commit` is the command for creating a commit, a “save point” containing a set of changes made to the files in a repo

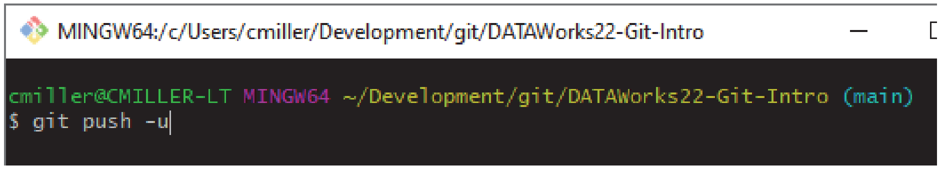
Our local git repo now has this commit, but the remote repo (on GitHub) is unaware of it and is unchanged.

Use git push/git pull to keep your local repo copy up-to-date

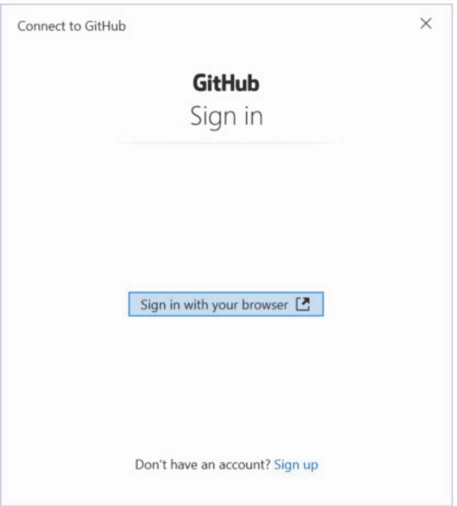
- The commands `git push` and `git pull` keep local copies of git repositories up-to-date with the external host of the repository
- After committing changes, use git push to propagate those changes to the host (and thus to everyone else with an up-to-date repo)
- Push changes at least at the end of the day when work is done
- Use git pull to make the local repo up-to-date with whatever is on the server
- Pull changes at the start of the day before starting work

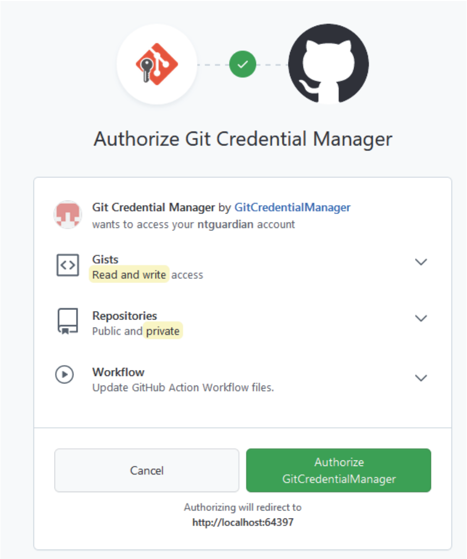


Pushing to GitHub for the first time may require authentication


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push -u
```


Connect to GitHub
GitHub
Sign in
Sign in with your browser
Don't have an account? Sign up

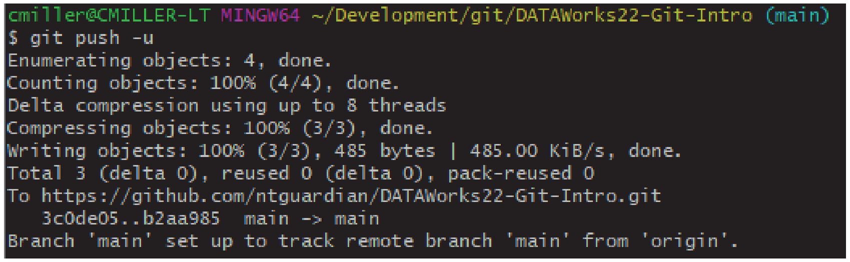

Authorize Git Credential Manager

Git Credential Manager by GitCredentialManager wants to access your ntguardian account

- Gists: Read and write access
- Repositories: Public and private
- Workflow: Update GitHub Action Workflow files.

Cancel
Authorize GitCredentialManager

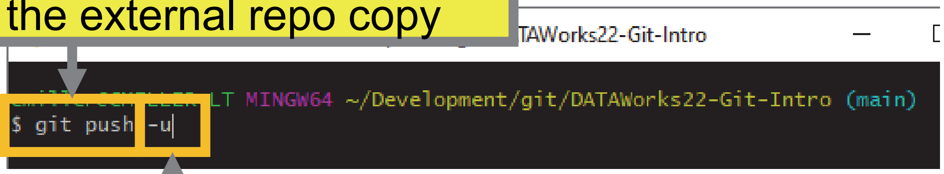
Authorizing will redirect to http://localhost:64397



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push -u
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 485 bytes | 485.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   3c0de05..b2aa985  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

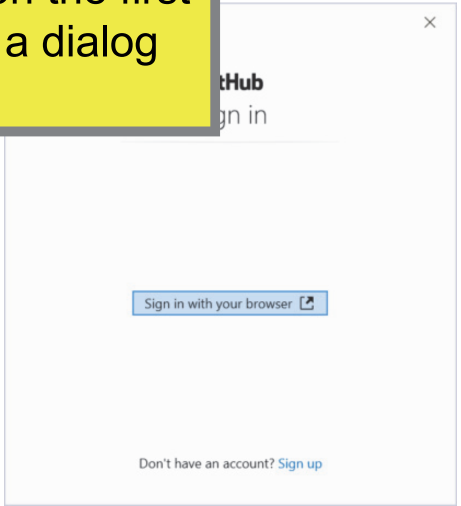
Pushing to GitHub for the first time may require authentication

`git push` is the command for propagating changes to the external repo copy

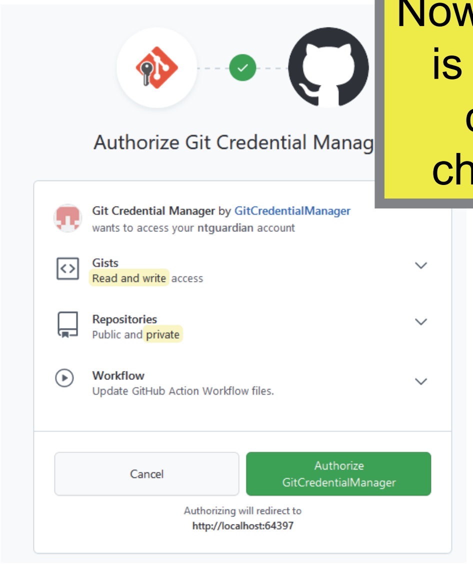

TAWorks22-Git-Intro

```
LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push -u
```

The -u option is needed for authentication the first time, opening a dialog box


tHub
gn in
Sign in with your browser
Don't have an account? Sign up

Follow the instructions…


Authorize Git Credential Manag
Cancel
Authorize GitCredentialManager

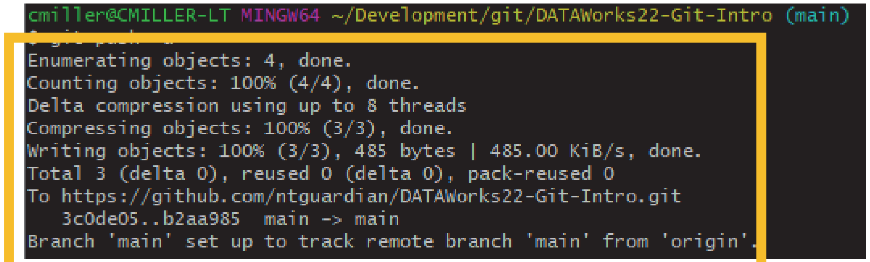
Git Credential Manager by GitCredentialManager wants to access your ntguardian account

Gists: Read and write access

Repositories: Public and private

Workflow: Update GitHub Action Workflow files.

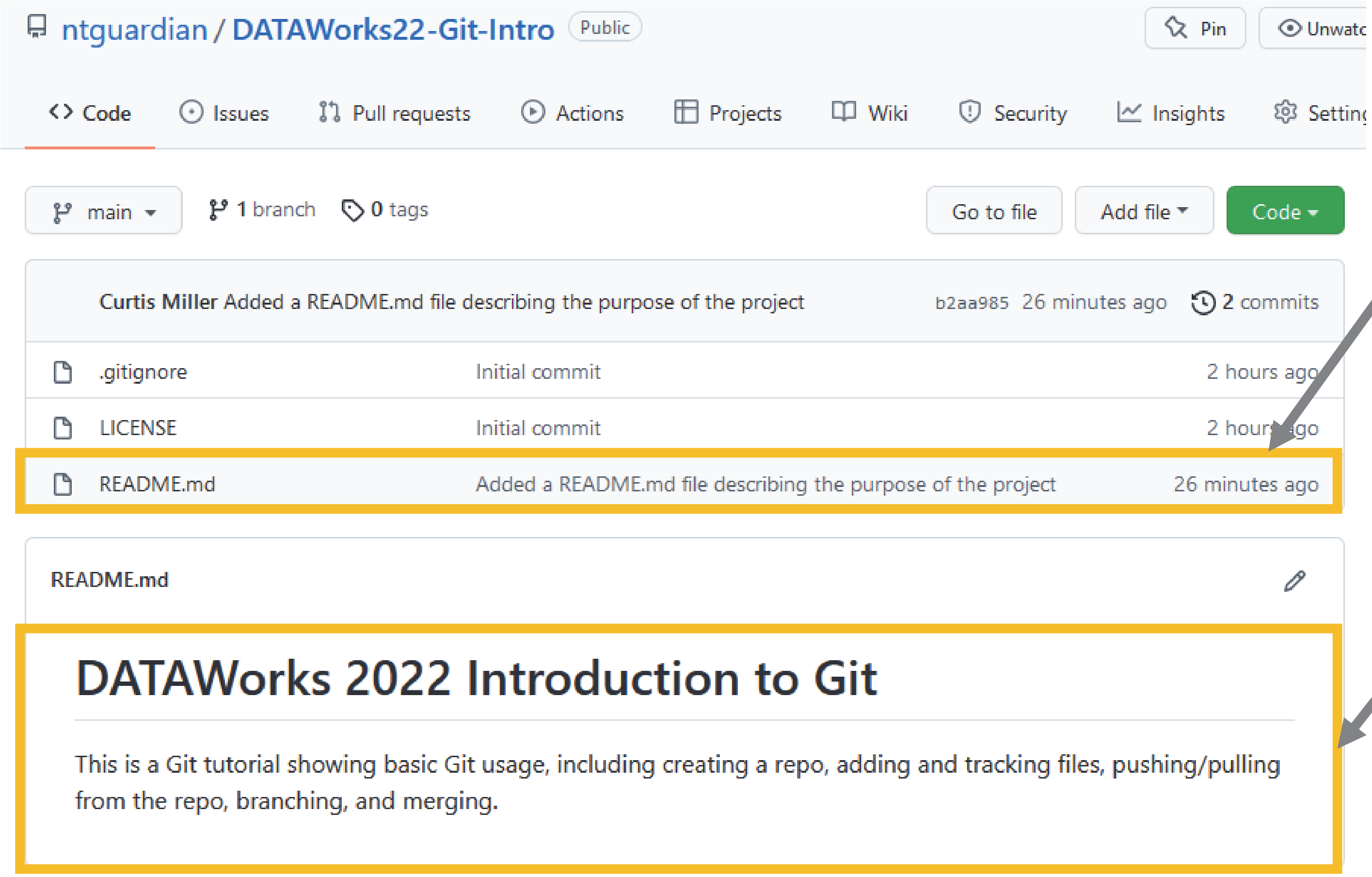
Authorizing will redirect to http://localhost:64397



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push -u
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 485 bytes | 485.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   3c0de05..b2aa985  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Now the command is done and we can see our changes online

The changes are now online


ntguardian / DATAWorks22-Git-Intro
Public
Pin
Unwatc
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settin
main
1 branch
0 tags
Go to file
Add file
Code

Curtis Miller Added a README.md file describing the purpose of the project b2aa985 26 minutes ago 2 commits

.gitignore	Initial commit	2 hours ago
LICENSE	Initial commit	2 hours ago
README.md	Added a README.md file describing the purpose of the project	26 minutes ago

README.md

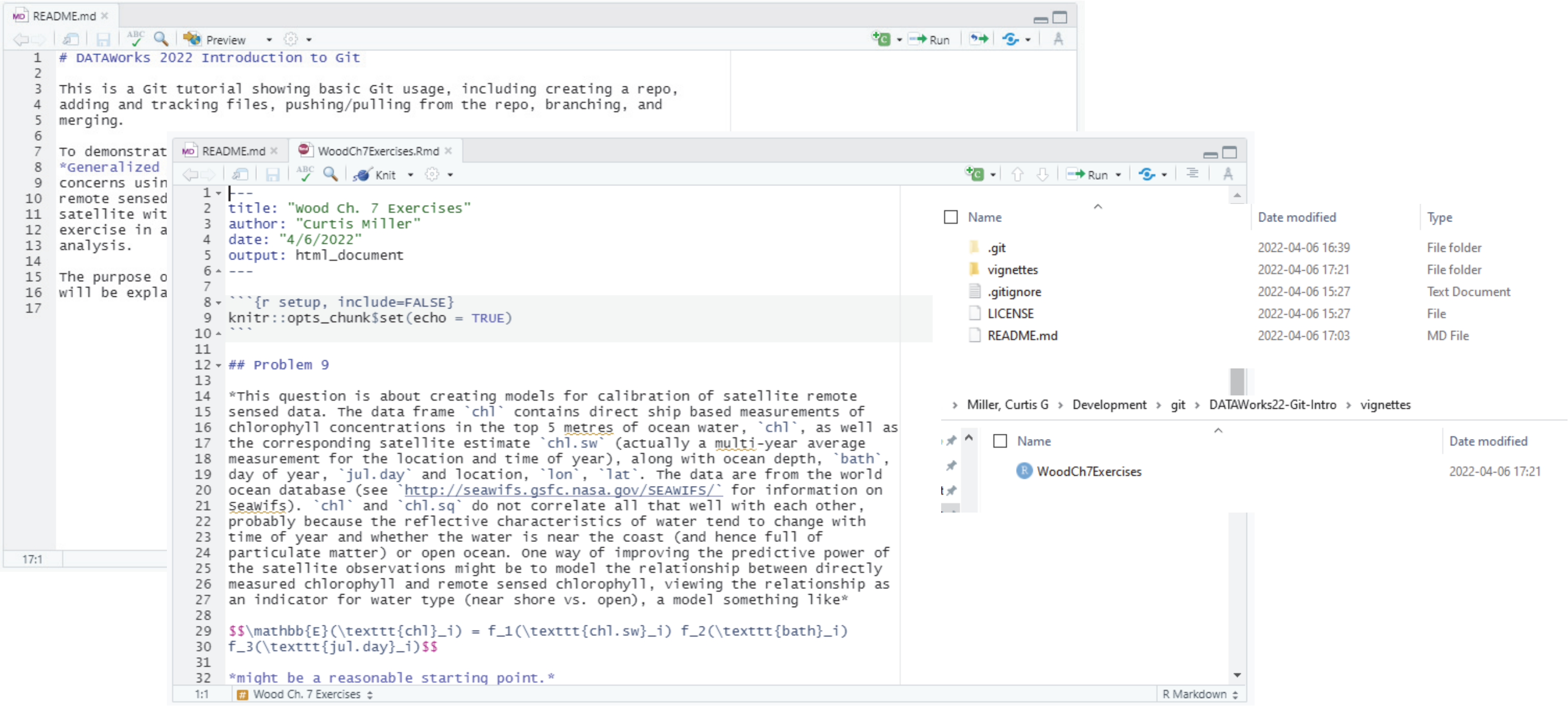
DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

This file is in the repo now; GitHub shows the time since the last commit, and the message associated with that commit

GitHub looks for README.md files and renders them specially; these files describe your project to others

Let’s make more changes!



```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.

To demonstrat
*Generalized
concerns usin
remote sensed
satellite wit
exercise in a
analysis.

The purpose o
will be expla
```

~~~
---
title: "Wood Ch. 7 Exercises"
author: "Curtis Miller"
date: "4/6/2022"
output: html_document
---

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

## Problem 9

*This question is about creating models for calibration of satellite remote
sensed data. The data frame `chl` contains direct ship based measurements of
chlorophyll concentrations in the top 5 metres of ocean water, `chl`, as well as
the corresponding satellite estimate `chl.sw` (actually a multi-year average
measurement for the location and time of year), along with ocean depth, `bath`,
day of year, `jul.day` and location, `lon`, `lat`. The data are from the world
ocean database (see `http://seawifs.gsfc.nasa.gov/SEAWIFS/` for information on
SeaWifs). `chl` and `chl.sq` do not correlate all that well with each other,
probably because the reflective characteristics of water tend to change with
time of year and whether the water is near the coast (and hence full of
particulate matter) or open ocean. One way of improving the predictive power of
the satellite observations might be to model the relationship between directly
measured chlorophyll and remote sensed chlorophyll, viewing the relationship as
an indicator for water type (near shore vs. open), a model something like*

$$\mathbb{E}(\texttt{chl}_i) = f_1(\texttt{chl.sw}_i) f_2(\texttt{bath}_i)
f_3(\texttt{jul.day}_i)$$

*might be a reasonable starting point.*
~~~

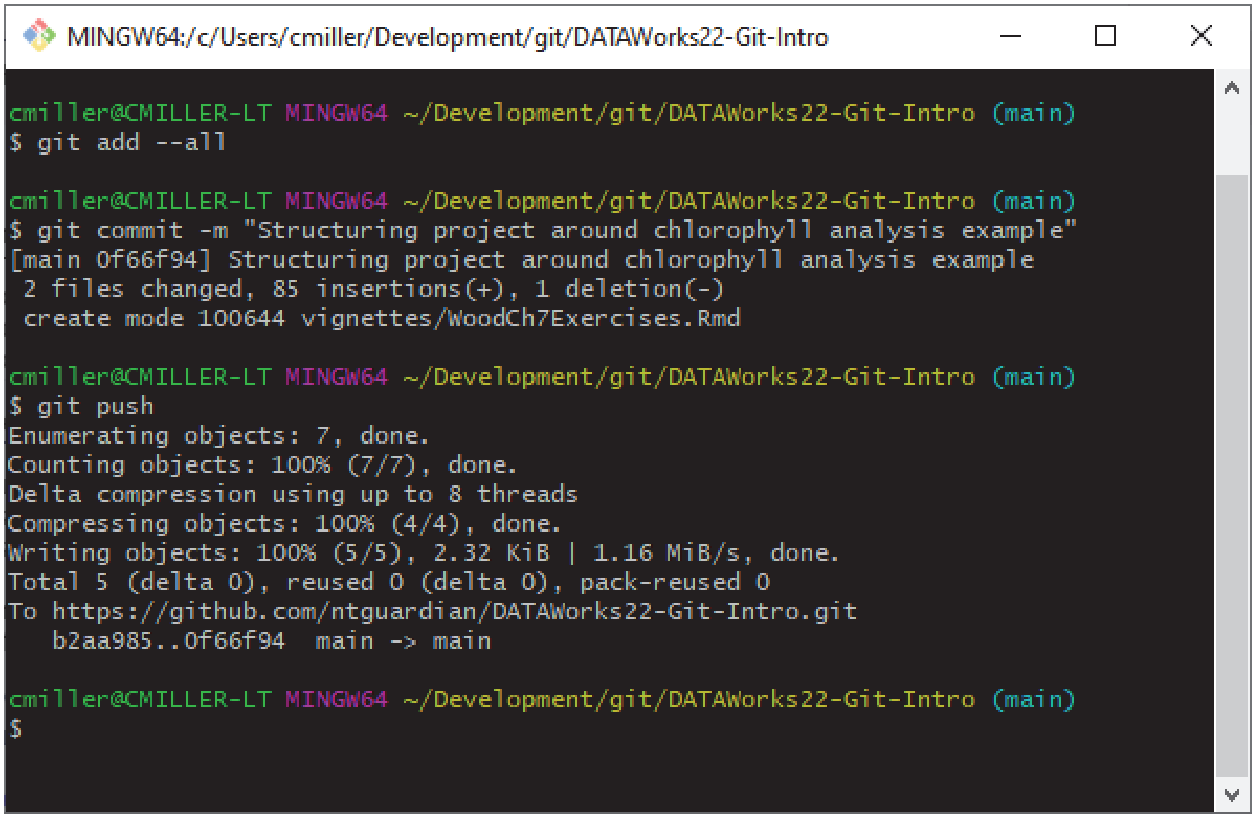
☐ Name	Date modified	Type
.git	2022-04-06 16:39	File folder
vignettes	2022-04-06 17:21	File folder
.gitignore	2022-04-06 15:27	Text Document
LICENSE	2022-04-06 15:27	File
README.md	2022-04-06 17:03	MD File

› Miller, Curtis G › Development › git › DATAWorks22-Git-Intro › vignettes

☐ Name	Date modified
WoodCh7Exercises	2022-04-06 17:21

We’re structuring the project around a statistical analysis of ocean chlorophyll data using generalized additive models. That will be described in a file stored in the vignettes directory. README.md has been changed to reflect this purpose.

Track them changes!



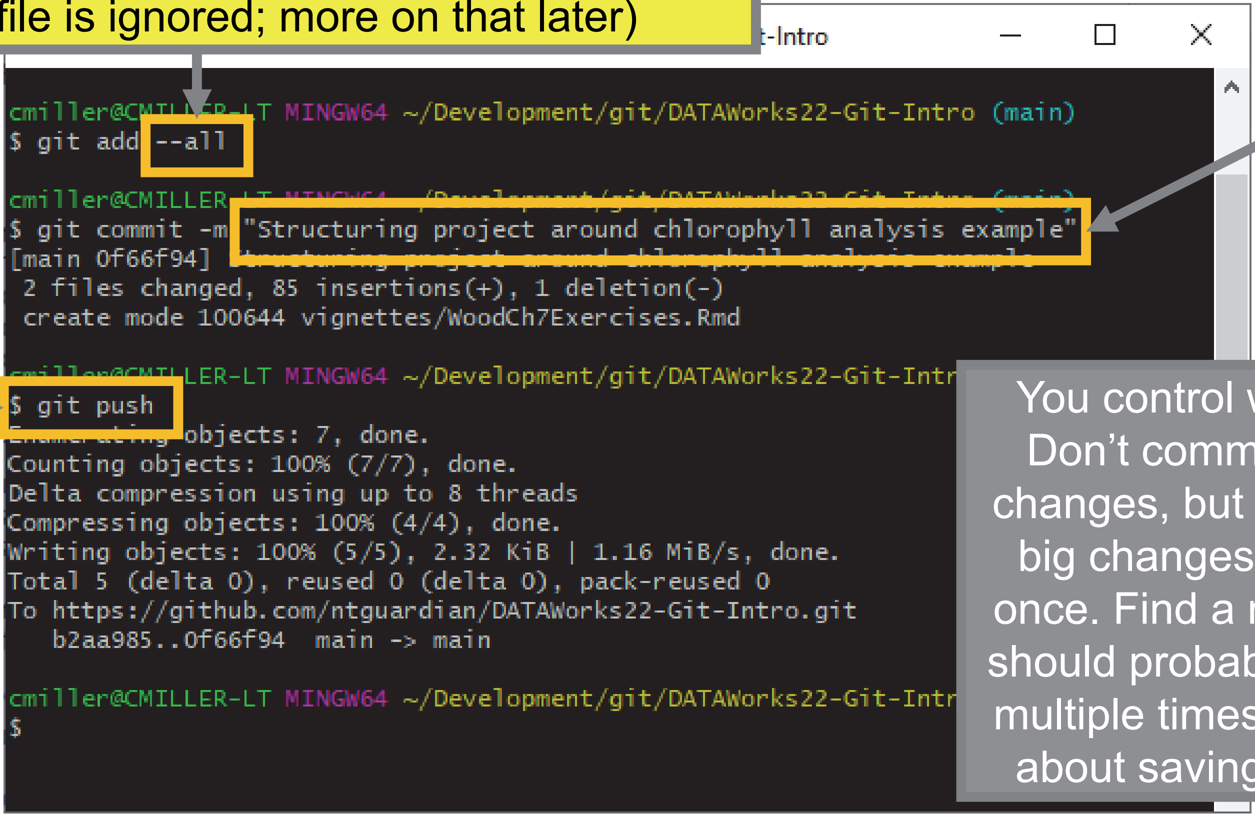
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Structuring project around chlorophyll analysis example"
[main 0f66f94] Structuring project around chlorophyll analysis example
 2 files changed, 85 insertions(+), 1 deletion(-)
 create mode 100644 vignettes/WoodCh7Exercises.Rmd

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 2.32 KiB | 1.16 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   b2aa985..0f66f94  main -> main

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$
```

Repeating these three commands will be 90% of your Git commands

Instead of listing files to track individually, use this option to add all changes made to any file in the repo (unless that file is ignored; more on that later)

Push your commands to the remote host; you don’t need to do this with every commit, but do it at least at the end of a workday, or when wanting to share immediately



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Structuring project around chlorophyll analysis example"
[main 0f66f94] Structuring project around chlorophyll analysis example
 2 files changed, 85 insertions(+), 1 deletion(-)
 create mode 100644 vignettes/WoodCh7Exercises.Rmd

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intr
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 2.32 KiB | 1.16 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   b2aa985..0f66f94  main -> main

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intr
$
```

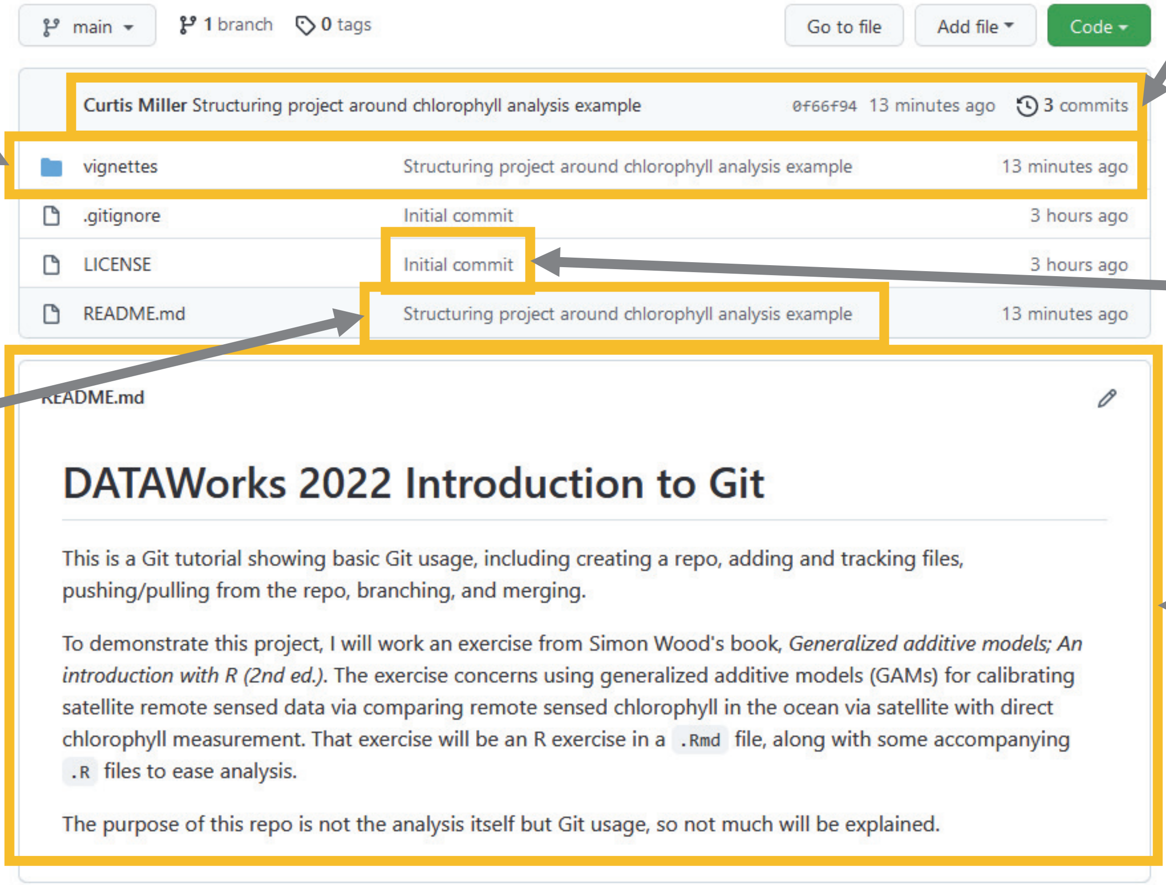
Commit message describing what change was made; please make your message informative

You control when you commit. Don’t commit a bunch of little changes, but don’t make several big changes and commit all at once. Find a middle ground. You should probably commit changes multiple times a day (don’t worry about saving space too much)

Notice what all has changed on GitHub

There are new files in the project

Now using the most recent commit message since this file was altered



Curtis Miller Structuring project around chlorophyll analysis example `0f66f94` 13 minutes ago 3 commits

vignettes	Structuring project around chlorophyll analysis example	13 minutes ago
.gitignore	Initial commit	3 hours ago
LICENSE	Initial commit	3 hours ago
README.md	Structuring project around chlorophyll analysis example	13 minutes ago

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

To demonstrate this project, I will work an exercise from Simon Wood's book, *Generalized additive models; An introduction with R (2nd ed.)*. The exercise concerns using generalized additive models (GAMs) for calibrating satellite remote sensed data via comparing remote sensed chlorophyll in the ocean via satellite with direct chlorophyll measurement. That exercise will be an R exercise in a `.Rmd` file, along with some accompanying `.R` files to ease analysis.

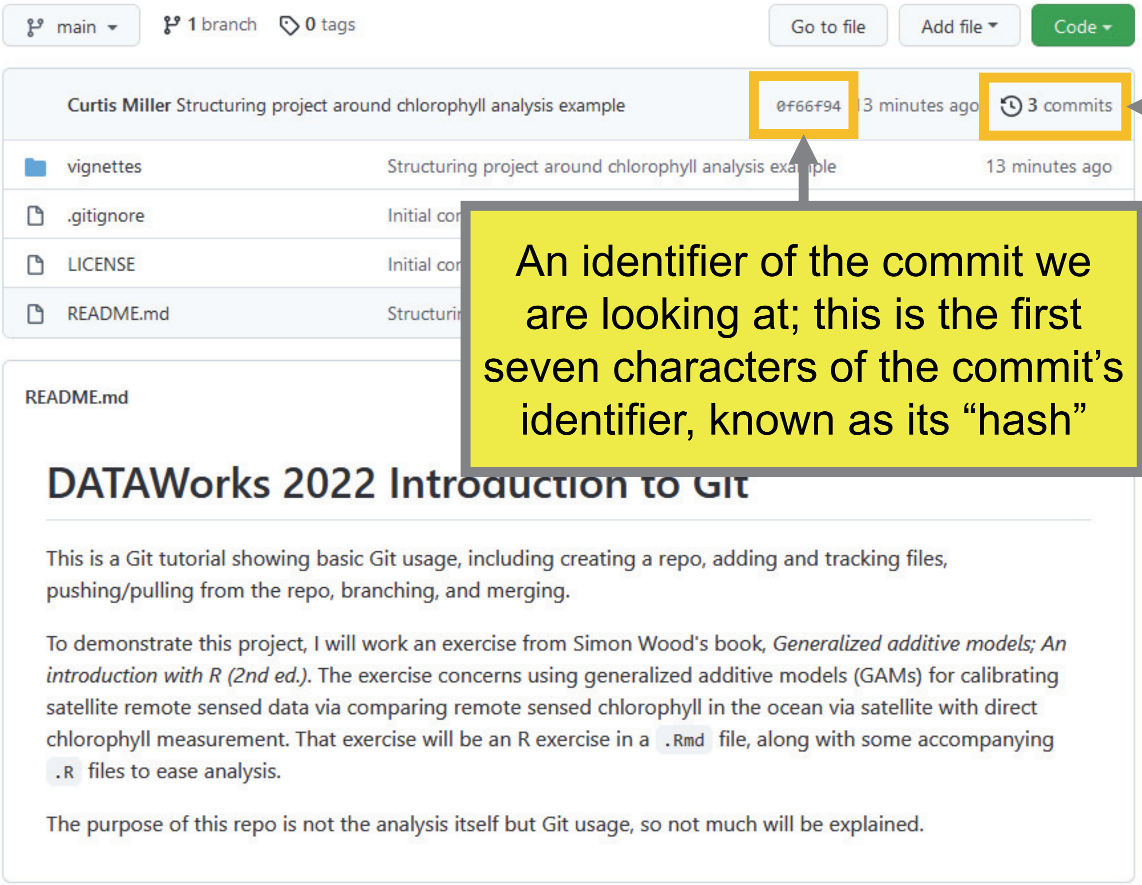
The purpose of this repo is not the analysis itself but Git usage, so not much will be explained.

Indicates the most up-to-date state of the project

This file has not been altered so nothing has changed

README.md now has the new text

Let’s see the history of the project so far



Curtis Miller Structuring project around chlorophyll analysis example `0f66f94` 13 minutes ago 3 commits

vignettes	Structuring project around chlorophyll analysis example	13 minutes ago
.gitignore	Initial co	
LICENSE	Initial co	
README.md	Structuri	

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

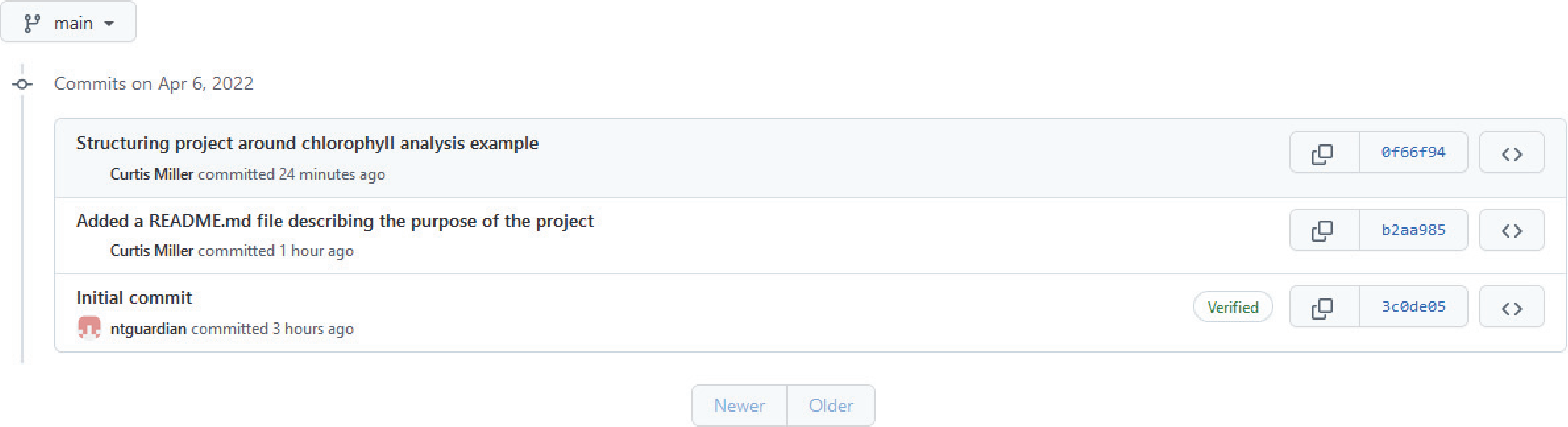
To demonstrate this project, I will work an exercise from Simon Wood's book, *Generalized additive models; An introduction with R (2nd ed.)*. The exercise concerns using generalized additive models (GAMs) for calibrating satellite remote sensed data via comparing remote sensed chlorophyll in the ocean via satellite with direct chlorophyll measurement. That exercise will be an R exercise in a `.Rmd` file, along with some accompanying `.R` files to ease analysis.

The purpose of this repo is not the analysis itself but Git usage, so not much will be explained.

Click this to see the history of the project

An identifier of the commit we are looking at; this is the first seven characters of the commit’s identifier, known as its “hash”

Our project has a short history



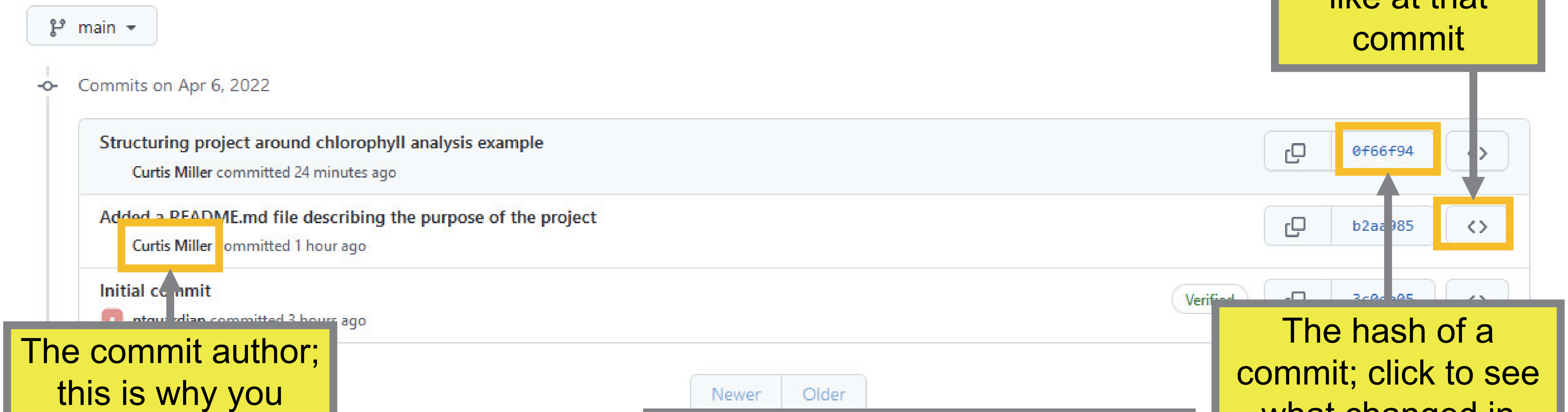
Commits on Apr 6, 2022

Structuring project around chlorophyll analysis example
Curtis Miller committed 24 minutes ago
0f66f94

Added a README.md file describing the purpose of the project
Curtis Miller committed 1 hour ago
b2aa985

Initial commit
ntguardian committed 3 hours ago
Verified
3c0de05

From here we can explore that history



Commits on Apr 6, 2022

Structuring project around chlorophyll analysis example
Curtis Miller committed 24 minutes ago
0f66f94

Added a README.md file describing the purpose of the project
Curtis Miller committed 1 hour ago
b2aa[illegible]985

Initial commit
ntguardian committed 3 hours ago
Verified

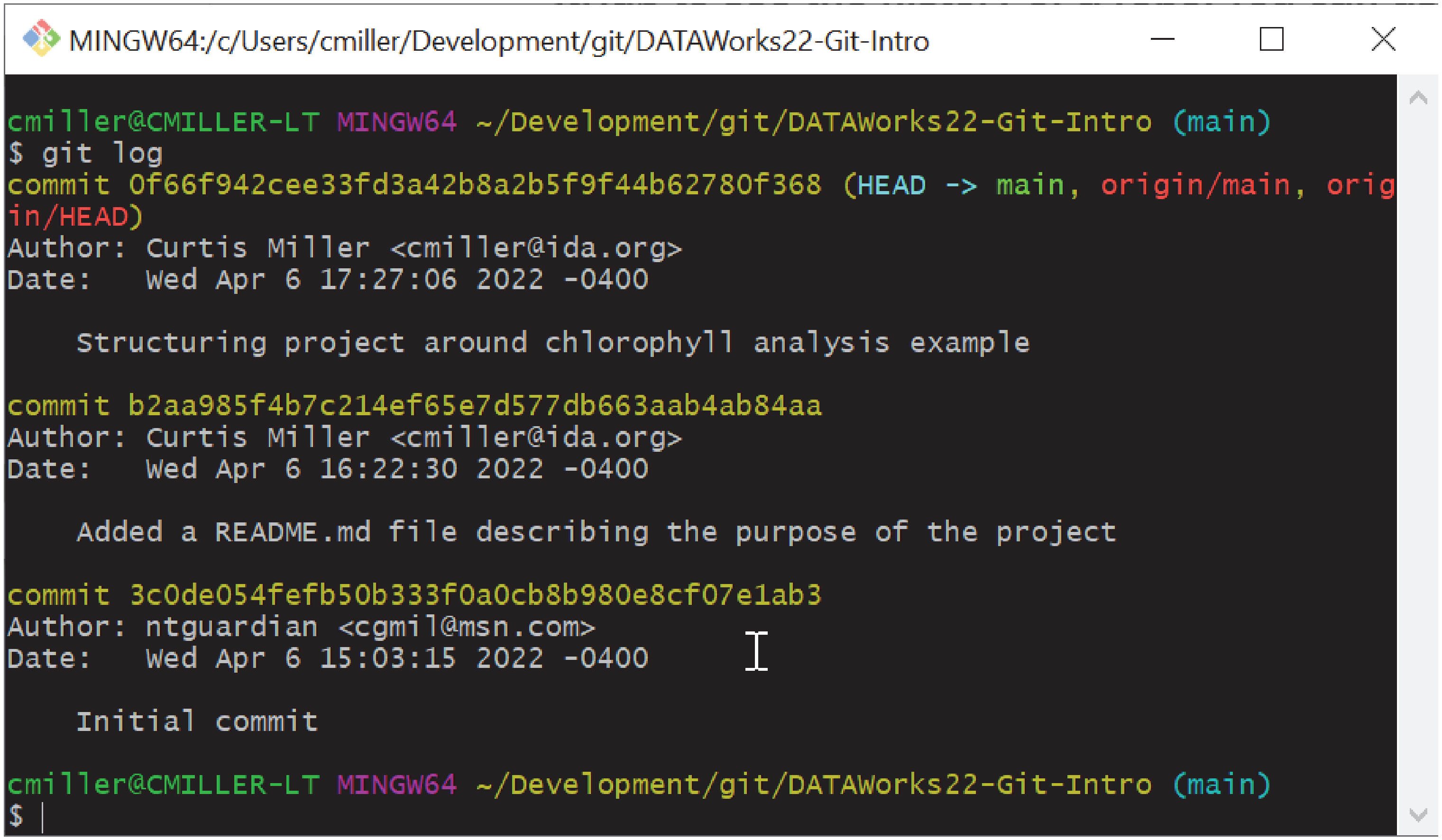
Click this to see what the project looked like at that commit

The commit author; this is why you needed to use `git config`

The hash of a commit; click to see what changed in that commit

None of the features seen here are exclusive to GitHub; other repo hosts have their own versions of this, and Git itself can allow looking at this information.

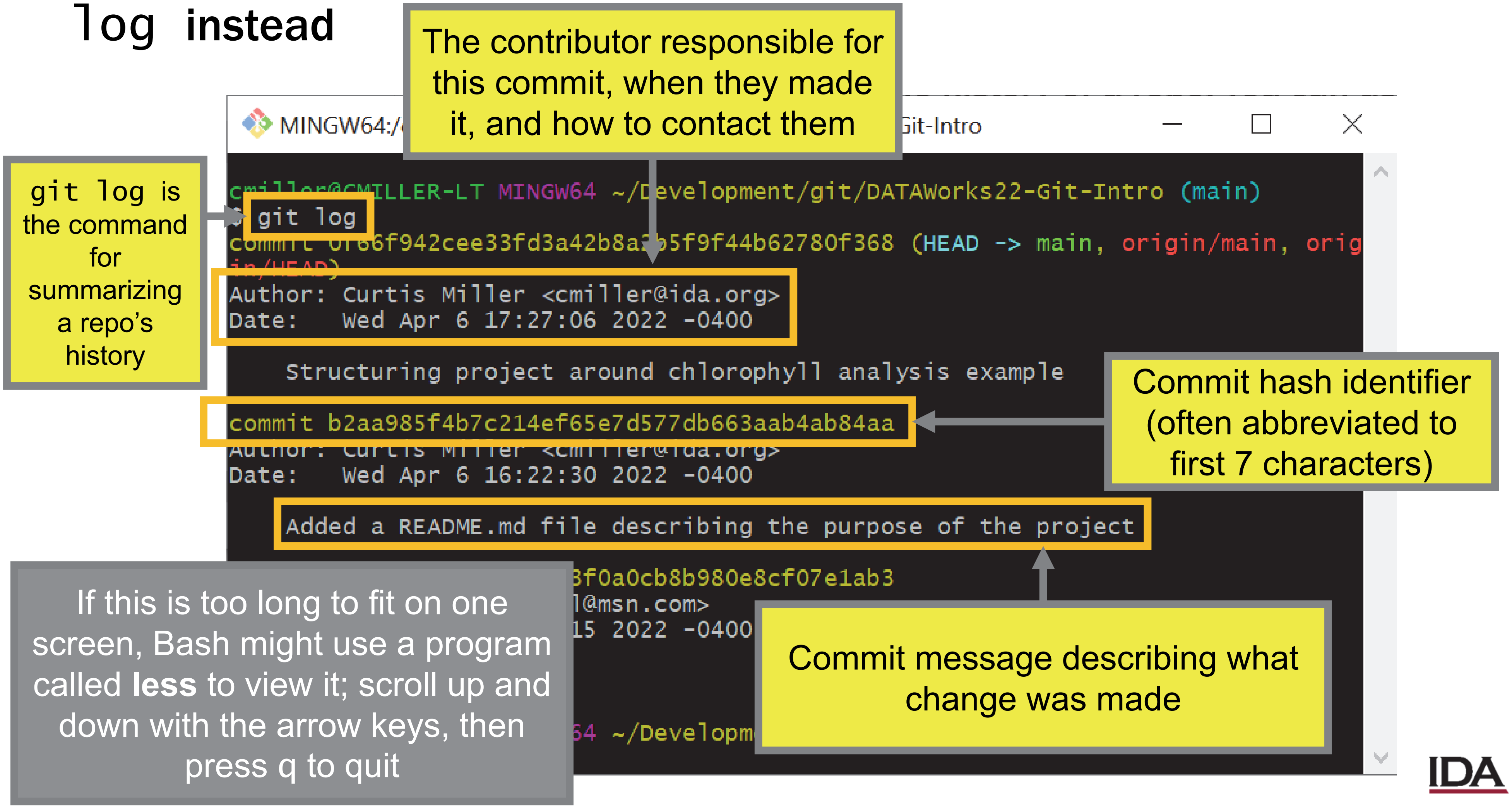
You don’t need GitHub to see the history of a repo; you can use `git log` instead



MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git log
commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368 (HEAD -> main, origin/main, orig
in/HEAD)
Author: Curtis Miller <cmiller@ida.org>
Date:   Wed Apr 6 17:27:06 2022 -0400

    Structuring project around chlorophyll analysis example

commit b2aa985f4b7c214ef65e7d577db663aab4ab84aa
Author: Curtis Miller <cmiller@ida.org>
Date:   Wed Apr 6 16:22:30 2022 -0400

    Added a README.md file describing the purpose of the project

commit 3c0de054fefb50b333f0a0cb8b980e8cf07e1ab3
Author: ntguardian <cgmil@msn.com>
Date:   Wed Apr 6 15:03:15 2022 -0400

    Initial commit

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ 
```

You don’t need GitHub to see the history of a repo; you can use `git log` instead



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git log
commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368 (HEAD -> main, origin/main, orig
in/HEAD)
Author: Curtis Miller <cmiller@ida.org>
Date:   Wed Apr 6 17:27:06 2022 -0400

    Structuring project around chlorophyll analysis example

commit b2aa985f4b7c214ef65e7d577db663aab4ab84aa
Author: Curtis Miller <cmiller@ida.org>
Date:   Wed Apr 6 16:22:30 2022 -0400

    Added a README.md file describing the purpose of the project

                        3f0a0cb8b980e8cf07e1ab3
                        l@msn.com>
                        15 2022 -0400


                        64 ~/Developm
```

The contributor responsible for this commit, when they made it, and how to contact them

`git log` is the command for summarizing a repo’s history

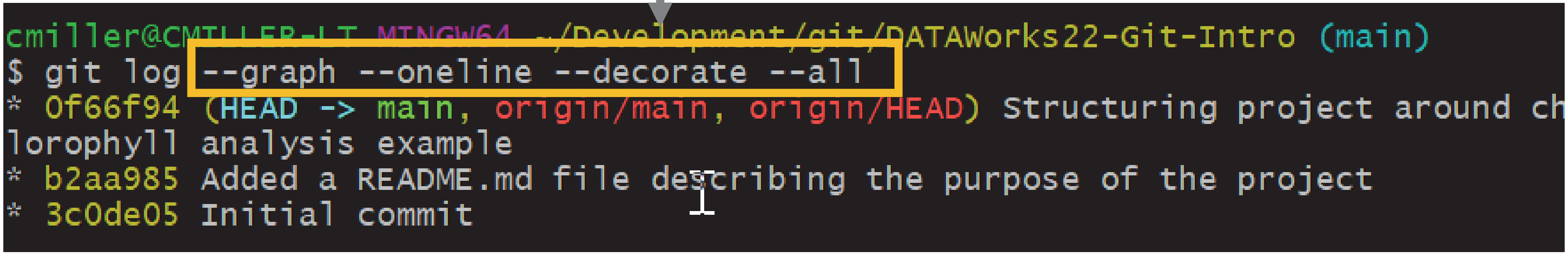
Commit hash identifier (often abbreviated to first 7 characters)

Commit message describing what change was made

If this is too long to fit on one screen, Bash might use a program called **less** to view it; scroll up and down with the arrow keys, then press q to quit

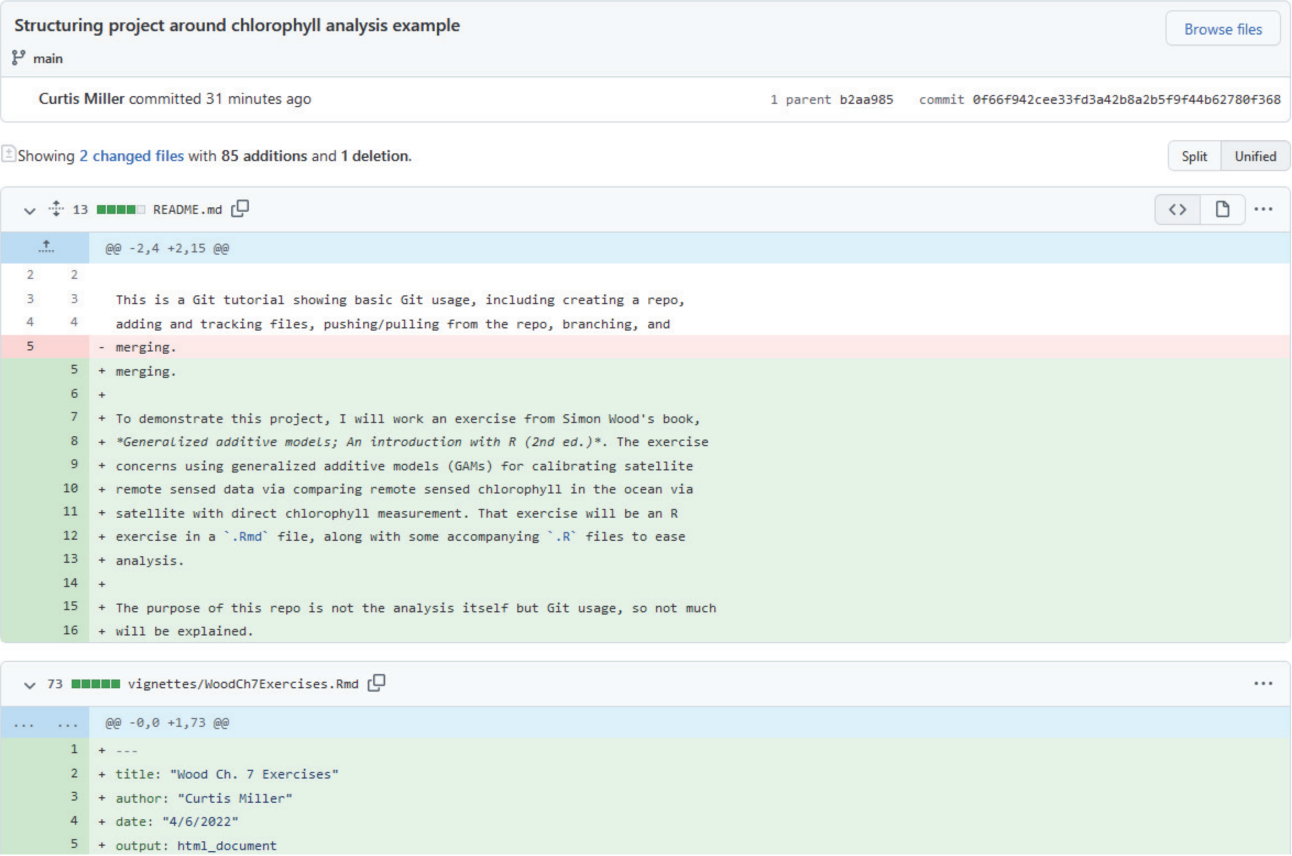
Options you can pass to `git log` can alter the output and make it easier to see at a glance

Using these options can make a more digestible history of a project; see `git log --help` for more



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git log --graph --oneline --decorate --all
* 0f66f94 (HEAD -> main, origin/main, origin/HEAD) Structuring project around ch
lorophyll analysis example
* b2aa985 Added a README.md file describing the purpose of the project
* 3c0de05 Initial commit
```

A commit lists precisely what the difference between it and the previous commit is; for text files, Git tracks *differences only*



Structuring project around chlorophyll analysis example

main

Curtis Miller committed 31 minutes ago

`1 parent b2aa985` `commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368`

Showing 2 changed files with **85 additions** and **1 deletion.**

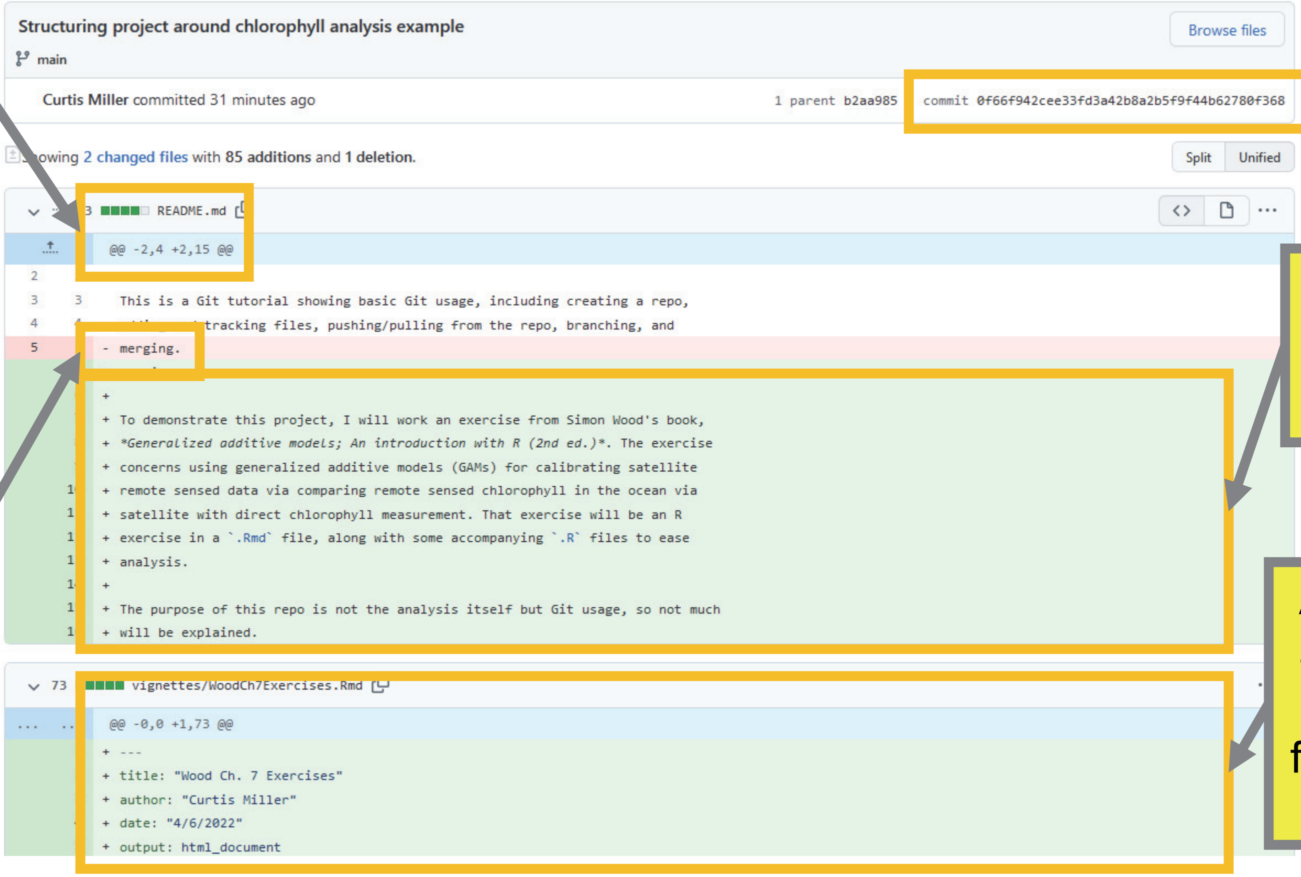
```
13 README.md
@@ -2,4 +2,15 @@

  This is a Git tutorial showing basic Git usage, including creating a repo,
  adding and tracking files, pushing/pulling from the repo, branching, and
- merging.
+ merging.
+
+ To demonstrate this project, I will work an exercise from Simon Wood's book,
+ *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
+ concerns using generalized additive models (GAMs) for calibrating satellite
+ remote sensed data via comparing remote sensed chlorophyll in the ocean via
+ satellite with direct chlorophyll measurement. That exercise will be an R
+ exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
+ analysis.
+
+ The purpose of this repo is not the analysis itself but Git usage, so not much
+ will be explained.
```

```
73 vignettes/WoodCh7Exercises.Rmd
@@ -0,0 +1,73 @@
+ ---
+ title: "Wood Ch. 7 Exercises"
+ author: "Curtis Miller"
+ date: "4/6/2022"
+ output: html_document
```

We are looking at what’s known as a Git diff (short for “difference”)

The file that was changed, with descriptor of where the change was made

Deleted lines are shown in red; if a line was modified, the old version is deleted (red) and the new version added (green)



Structuring project around chlorophyll analysis example

main

Curtis Miller committed 31 minutes ago

`1 parent b2aa985`

`commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368`

owing 2 changed files with **85 additions** and **1 deletion.**

```
3 README.md
@@ -2,4 +2,15 @@

  This is a Git tutorial showing basic Git usage, including creating a repo,
  tracking files, pushing/pulling from the repo, branching, and
- merging.
+
+ To demonstrate this project, I will work an exercise from Simon Wood's book,
+ *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
+ concerns using generalized additive models (GAMs) for calibrating satellite
+ remote sensed data via comparing remote sensed chlorophyll in the ocean via
+ satellite with direct chlorophyll measurement. That exercise will be an R
+ exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
+ analysis.
+
+ The purpose of this repo is not the analysis itself but Git usage, so not much
+ will be explained.
```

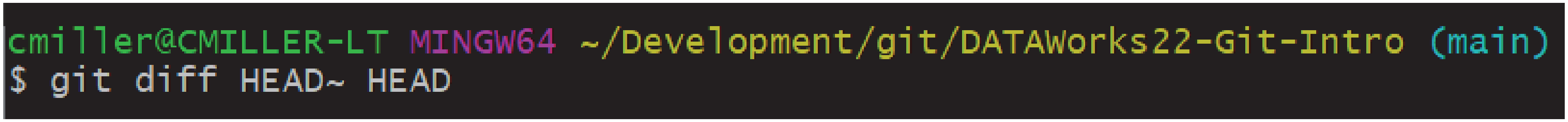
```
73 vignettes/WoodCh7Exercises.Rmd
@@ -0,0 +1,73 @@
+ ---
+ title: "Wood Ch. 7 Exercises"
+ author: "Curtis Miller"
+ date: "4/6/2022"
+ output: html_document
```

The full hash identifier of the commit

Additions to a file are shown in green

A whole new file is all additions, so all green (a deleted file would be all red for all deletions)

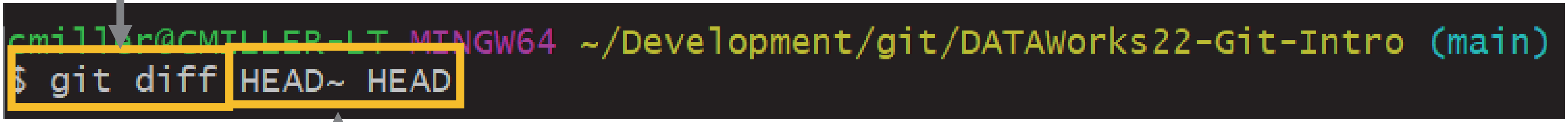
We can get the same information using `git diff`



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git diff HEAD~ HEAD
```

`git diff` can compare many entities Git tracks

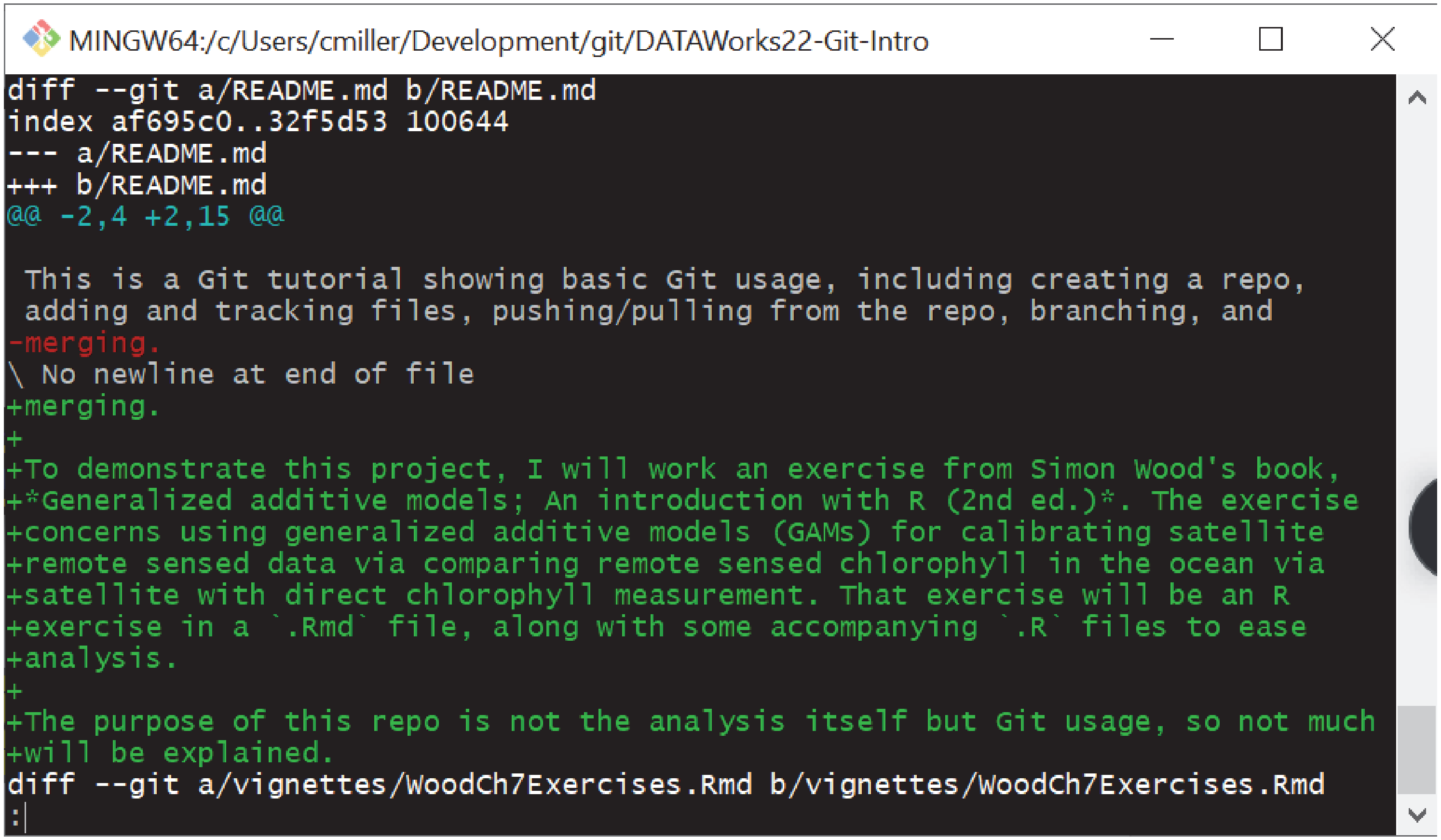
`git diff` is the command for comparing files tracked by Git



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git diff HEAD~ HEAD
```

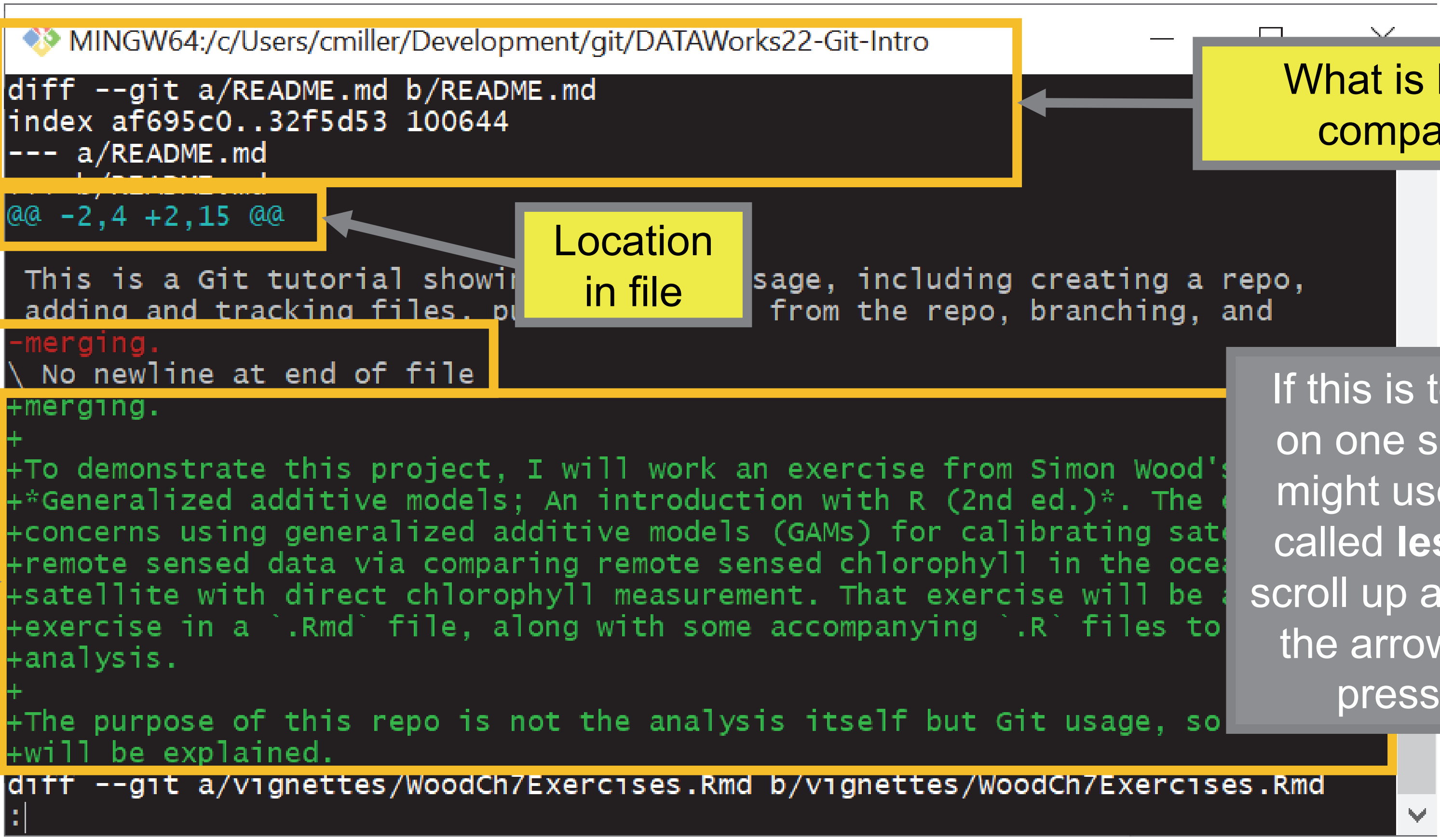
Determines what to compare; in this case, the current commit with the one prior

Using `git diff` we can see changes made in commits to files


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@

 This is a Git tutorial showing basic Git usage, including creating a repo,
 adding and tracking files, pushing/pulling from the repo, branching, and
-merging.
\ No newline at end of file
+merging.
+
+To demonstrate this project, I will work an exercise from Simon Wood's book,
+*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
+concerns using generalized additive models (GAMs) for calibrating satellite
+remote sensed data via comparing remote sensed chlorophyll in the ocean via
+satellite with direct chlorophyll measurement. That exercise will be an R
+exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
+analysis.
+
+The purpose of this repo is not the analysis itself but Git usage, so not much
+will be explained.
diff --git a/vignettes/WoodCh7Exercises.Rmd b/vignettes/WoodCh7Exercises.Rmd
:
```

For plain text files, Git tracks only the changes, and doesn't copy whole files if it doesn't need to


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@

 This is a Git tutorial showi                 sage, including creating a repo,
 adding and tracking files, pu                 from the repo, branching, and
-merging.
\ No newline at end of file
+merging.
+
+To demonstrate this project, I will work an exercise from Simon Wood's
+*Generalized additive models; An introduction with R (2nd ed.)*. The
+concerns using generalized additive models (GAMs) for calibrating sate
+remote sensed data via comparing remote sensed chlorophyll in the ocea
+satellite with direct chlorophyll measurement. That exercise will be a
+exercise in a `.Rmd` file, along with some accompanying `.R` files to
+analysis.
+
+The purpose of this repo is not the analysis itself but Git usage, so
+will be explained.
diff --git a/vignettes/WoodCh7Exercises.Rmd b/vignettes/WoodCh7Exercises.Rmd
:
```

What is being compared

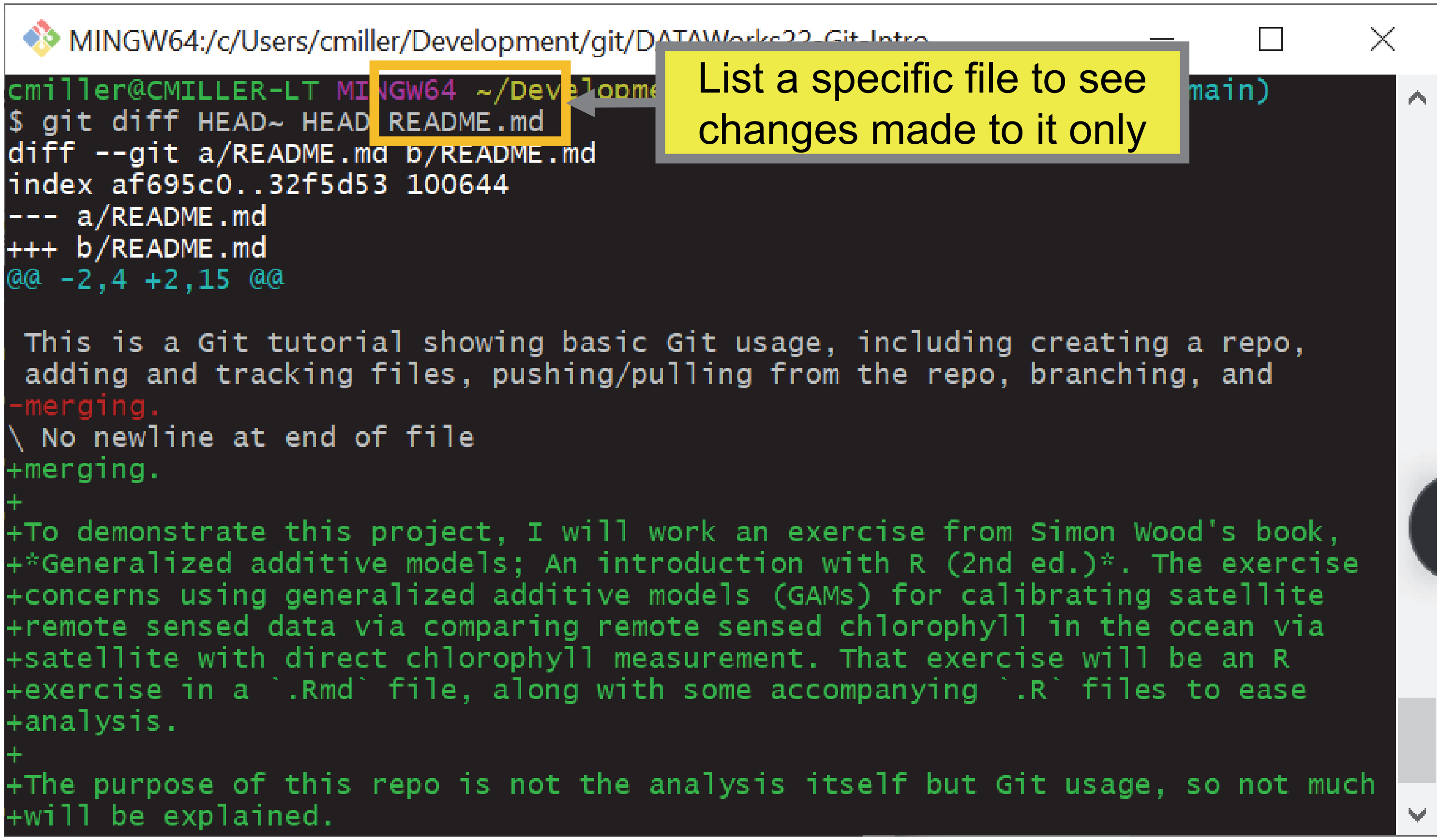
Location in file

Red: what was removed

Green: what was added

If this is too long to fit on one screen, Bash might use a program called **less** to view it; scroll up and down with the arrow keys, then press q to quit

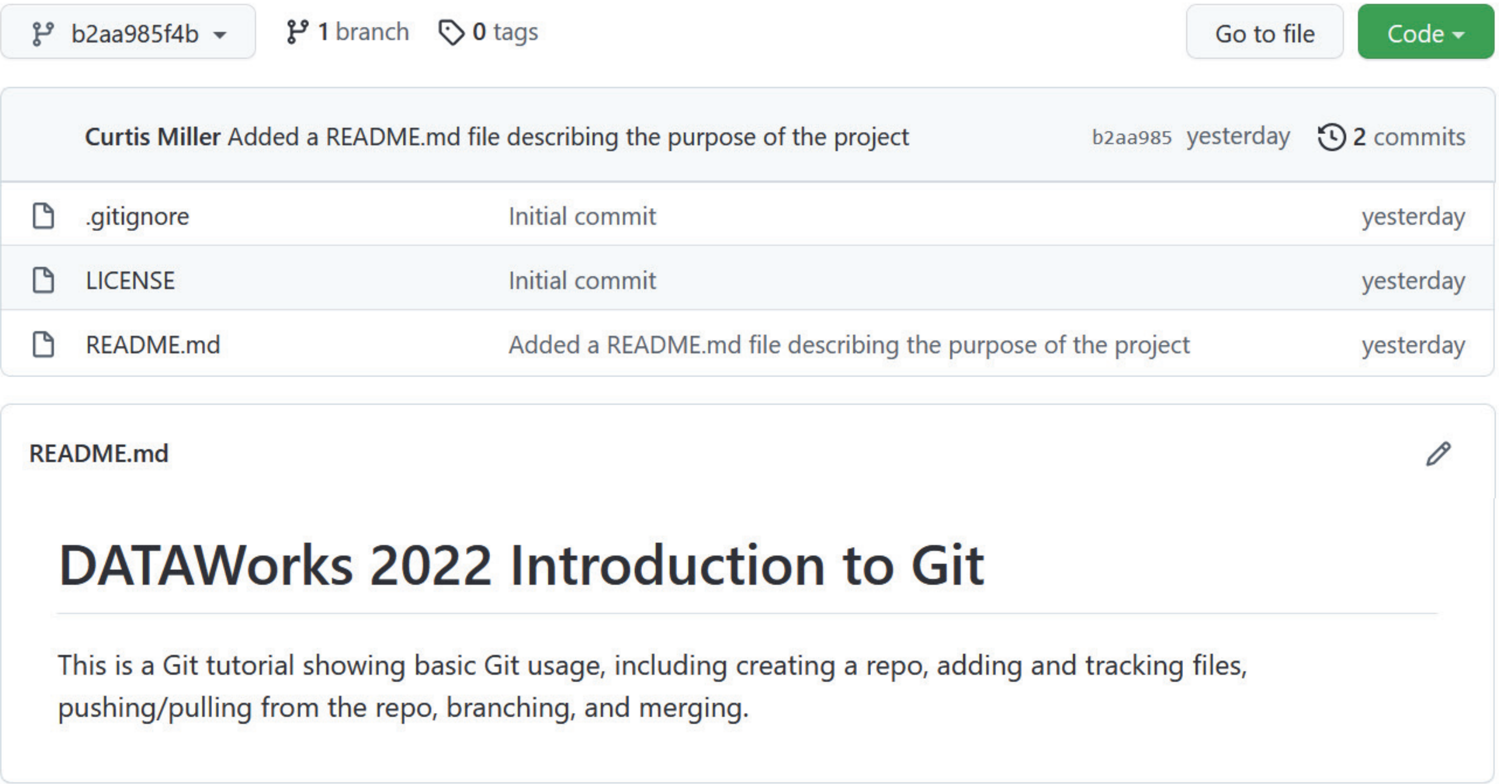
We can also see what changed in a specific file


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Developme                                       main)
$ git diff HEAD~ HEAD README.md
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@

 This is a Git tutorial showing basic Git usage, including creating a repo,
 adding and tracking files, pushing/pulling from the repo, branching, and
-merging.
\ No newline at end of file
+merging.
+
+To demonstrate this project, I will work an exercise from Simon Wood's book,
+*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
+concerns using generalized additive models (GAMs) for calibrating satellite
+remote sensed data via comparing remote sensed chlorophyll in the ocean via
+satellite with direct chlorophyll measurement. That exercise will be an R
+exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
+analysis.
+
+The purpose of this repo is not the analysis itself but Git usage, so not much
+will be explained.
```

List a specific file to see changes made to it only

We can also see what a repo looked like at the moment a commit was made



Curtis Miller Added a README.md file describing the purpose of the project b2aa985 yesterday 2 commits

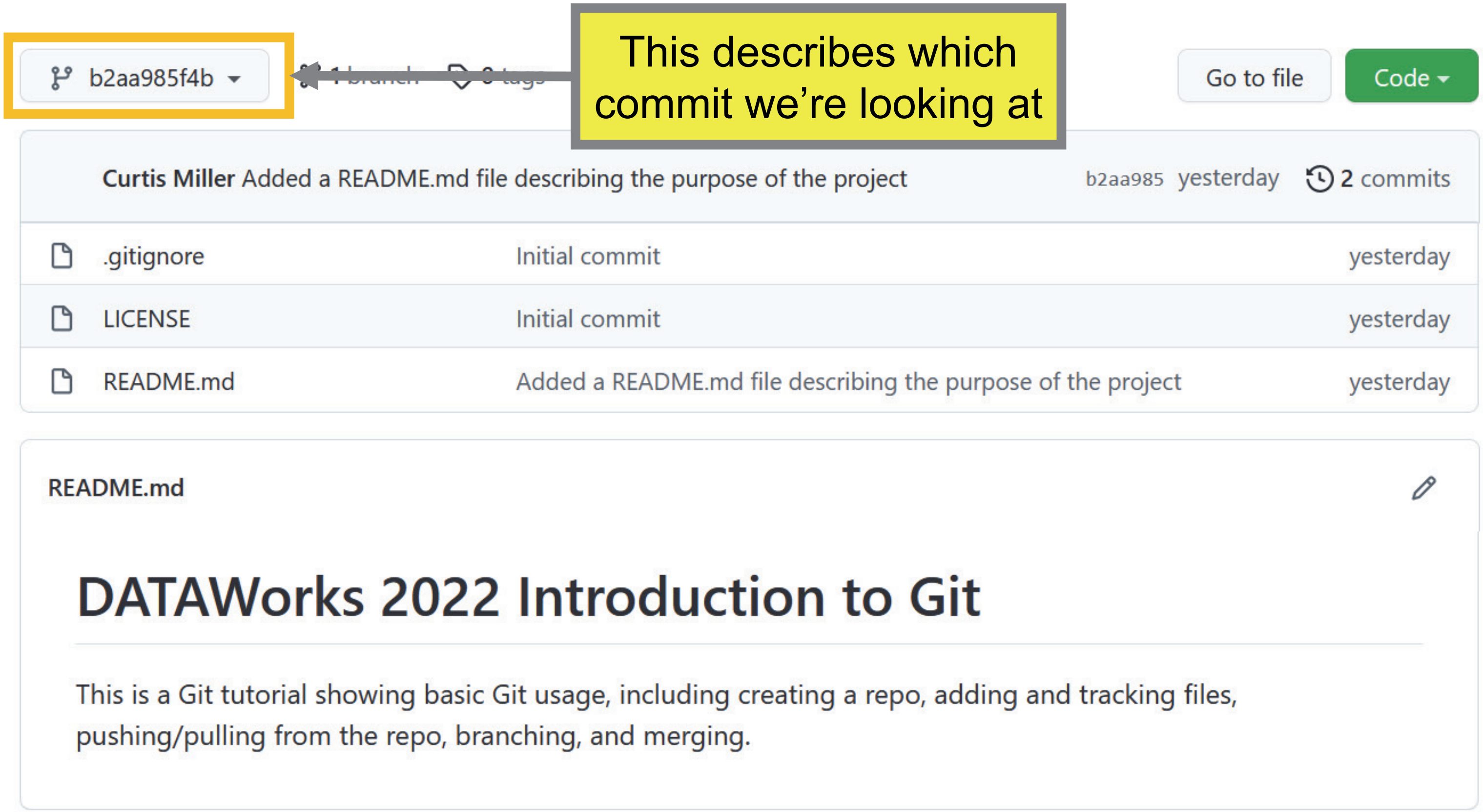
.gitignore	Initial commit	yesterday
LICENSE	Initial commit	yesterday
README.md	Added a README.md file describing the purpose of the project	yesterday

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

This includes not only changes made but the entire state of the project



This describes which commit we’re looking at

Curtis Miller Added a README.md file describing the purpose of the project b2aa985 yesterday 2 commits

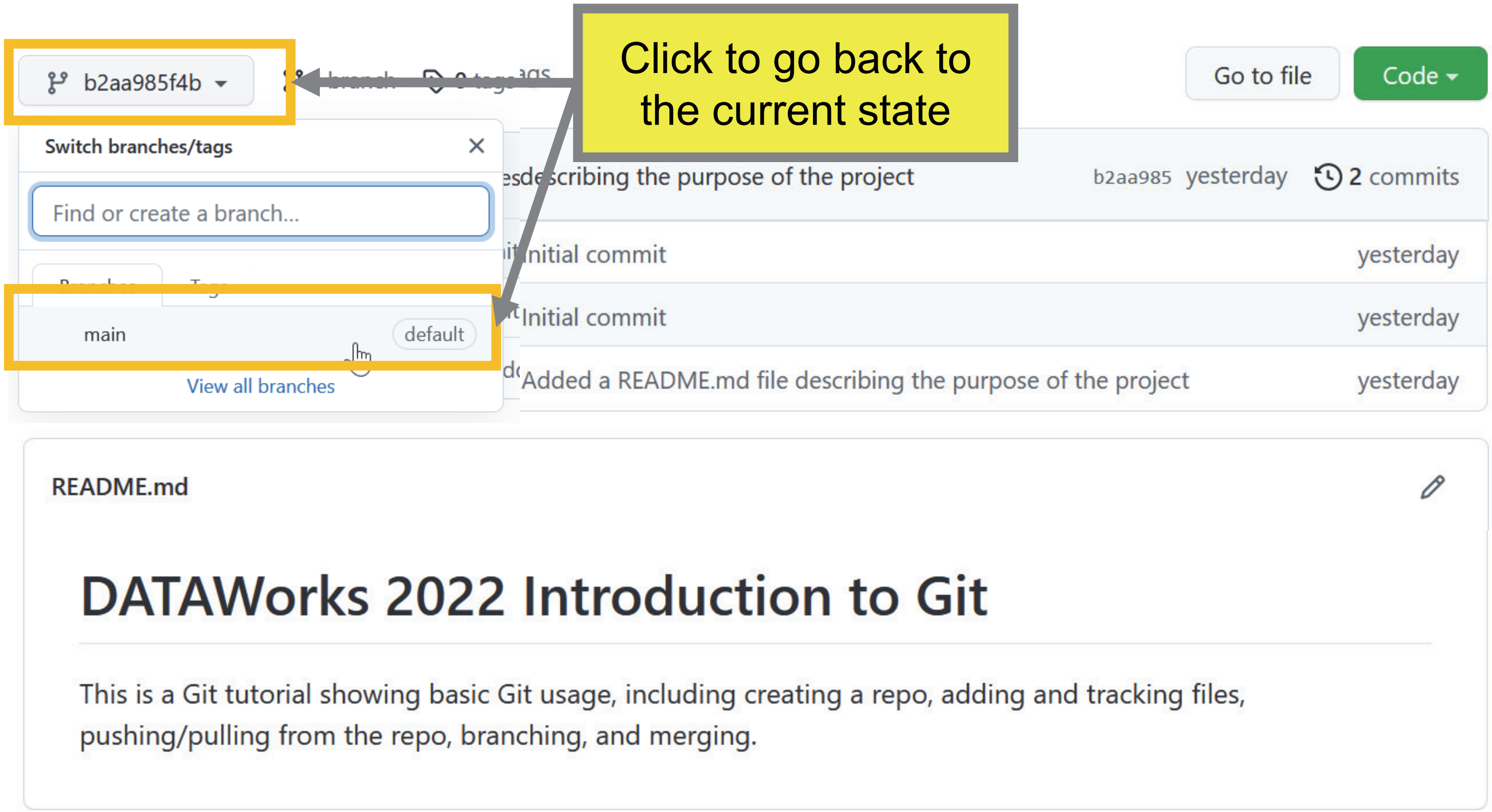
.gitignore	Initial commit	yesterday
LICENSE	Initial commit	yesterday
README.md	Added a README.md file describing the purpose of the project	yesterday

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

We can also see what a repo looked like at the moment a commit was made


b2aa985f4b
Go to file
Code
Switch branches/tags
Find or create a branch...
main
default
View all branches
README.md

Click to go back to the current state

describing the purpose of the project b2aa985 yesterday 2 commits

Initial commit yesterday

Initial commit yesterday

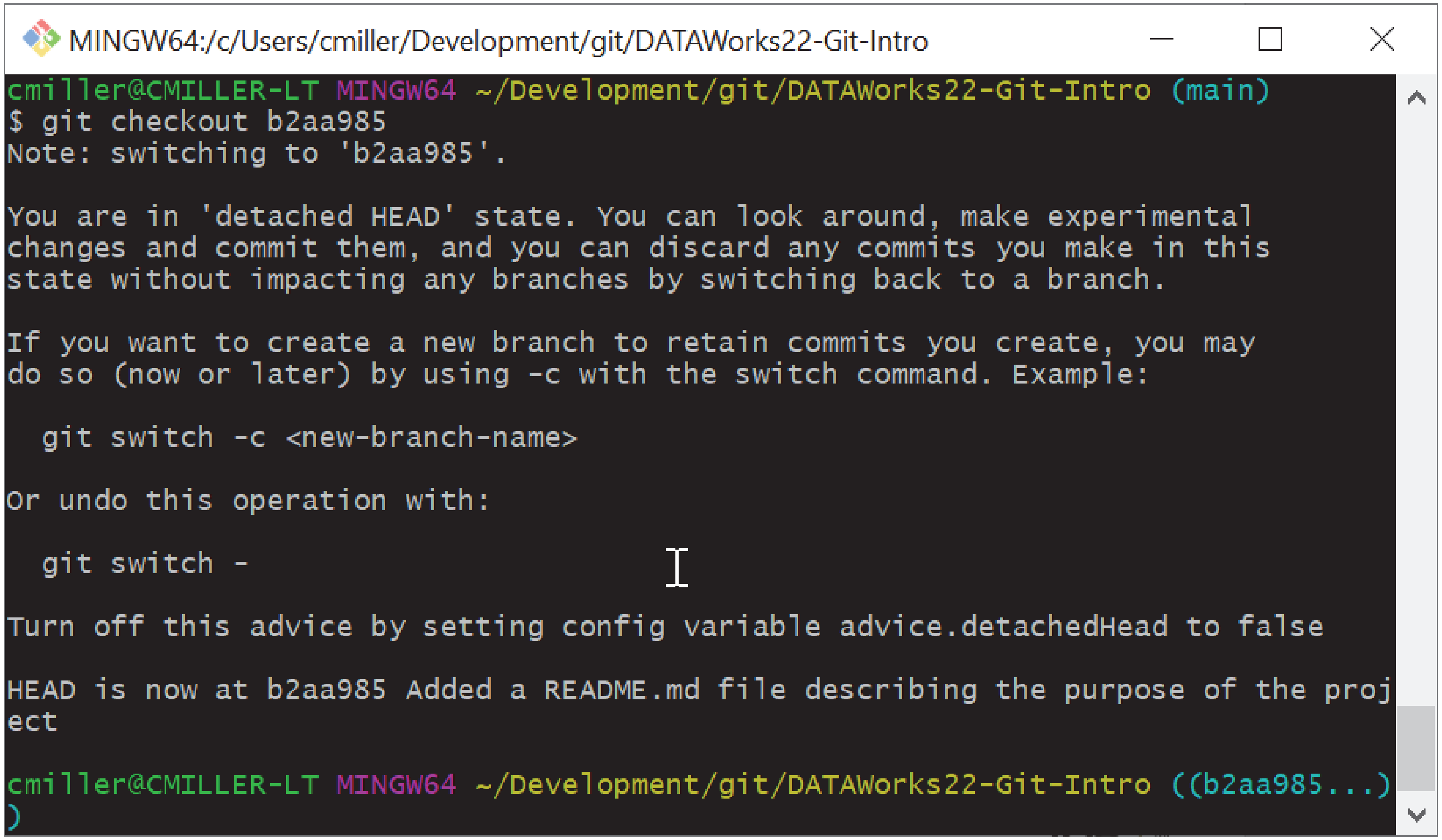
Added a README.md file describing the purpose of the project yesterday

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling from the repo, branching, and merging.

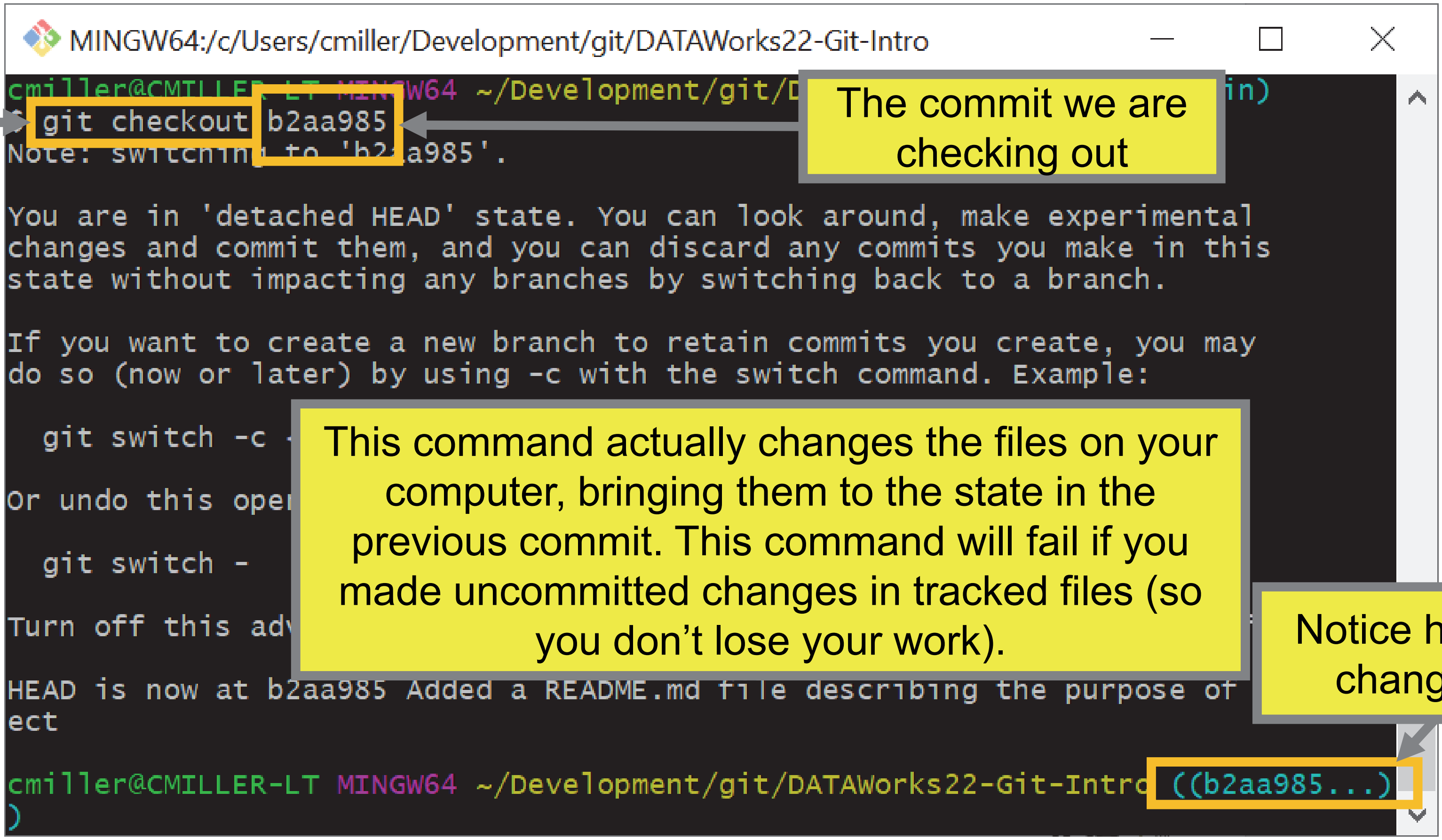
We can do this without GitHub using `git checkout`



MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git checkout b2aa985
Note: switching to 'b2aa985'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at b2aa985 Added a README.md file describing the purpose of the proj
ect

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro ((b2aa985...)
)
```

This will bring a repo’s files to a previously tracked state


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/D                    in)
  git checkout b2aa985
Note: switching to 'b2aa985'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c

Or undo this oper

  git switch -

Turn off this adv

HEAD is now at b2aa985 Added a README.md file describing the purpose of
ect

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro ((b2aa985...)
)
```

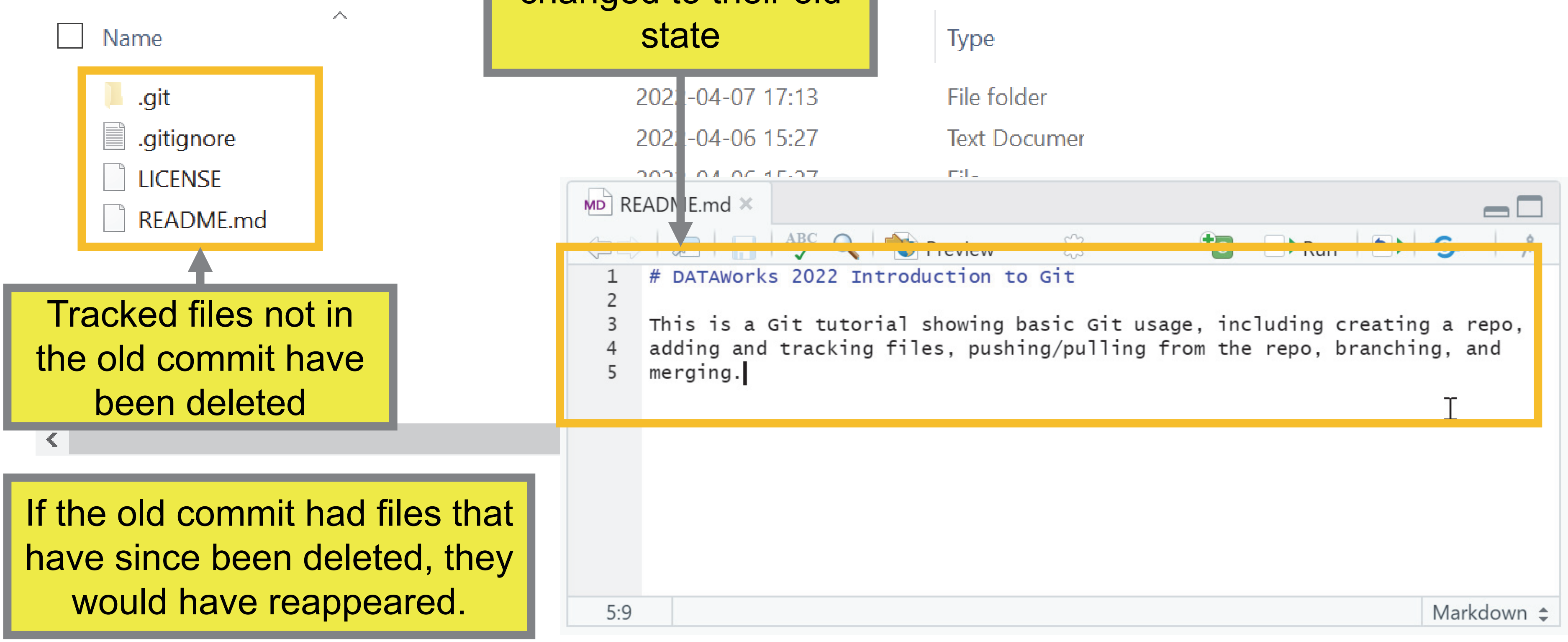
`git checkout` gets different versions of the repo, including old commits and branches

The commit we are checking out

This command actually changes the files on your computer, bringing them to the state in the previous commit. This command will fail if you made uncommitted changes in tracked files (so you don’t lose your work).

Notice how this changed…

The repo’s contents on our hard drive will change when checking out a commit


Name
.git
.gitignore
LICENSE
README.md
Type
202 -04-07 17:13
202 -04-06 15:27
File folder
Text Documer
README.md
Preview
Run
5:9
Markdown

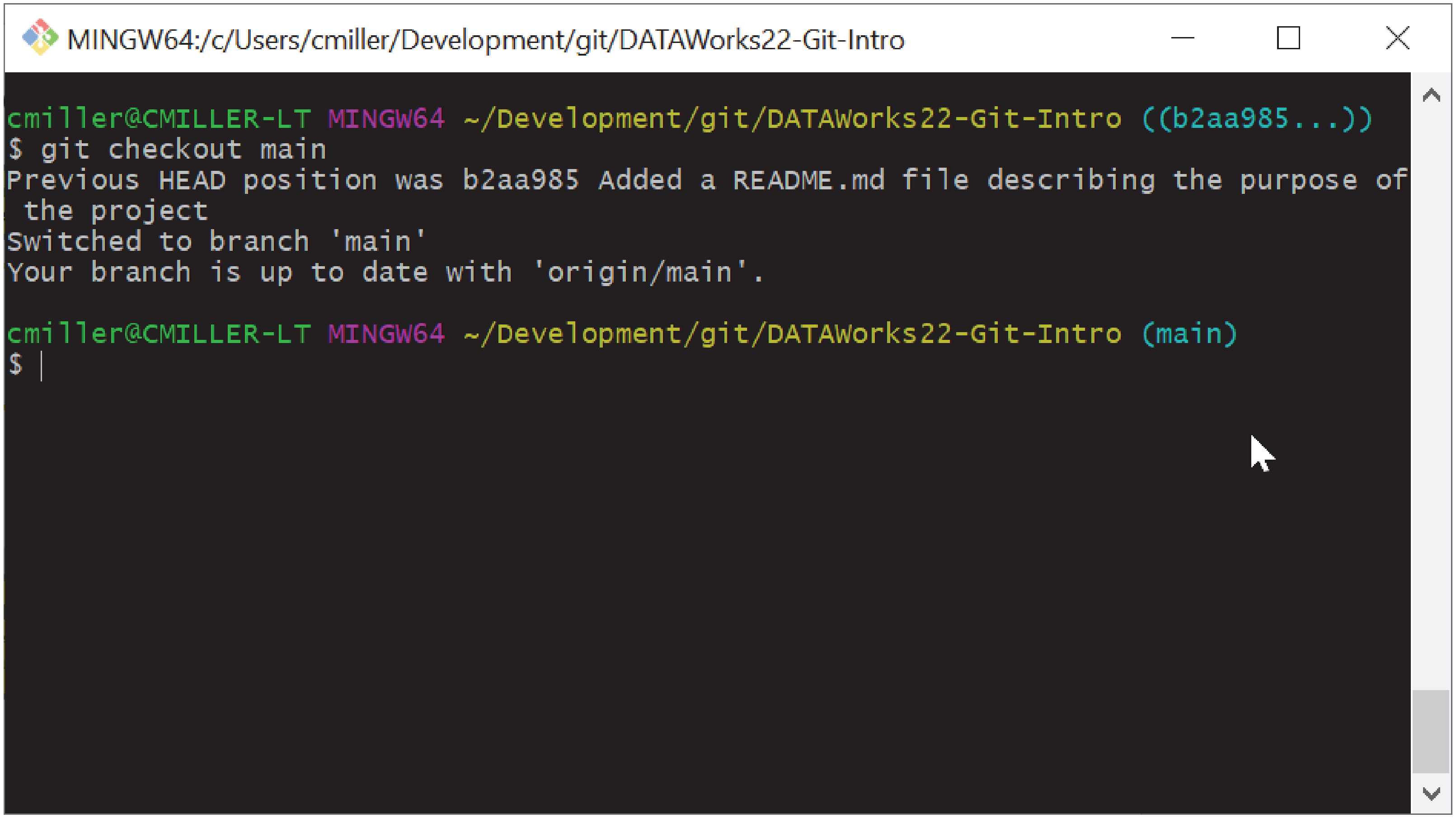
Files have been changed to their old state

```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.
```

Tracked files not in the old commit have been deleted

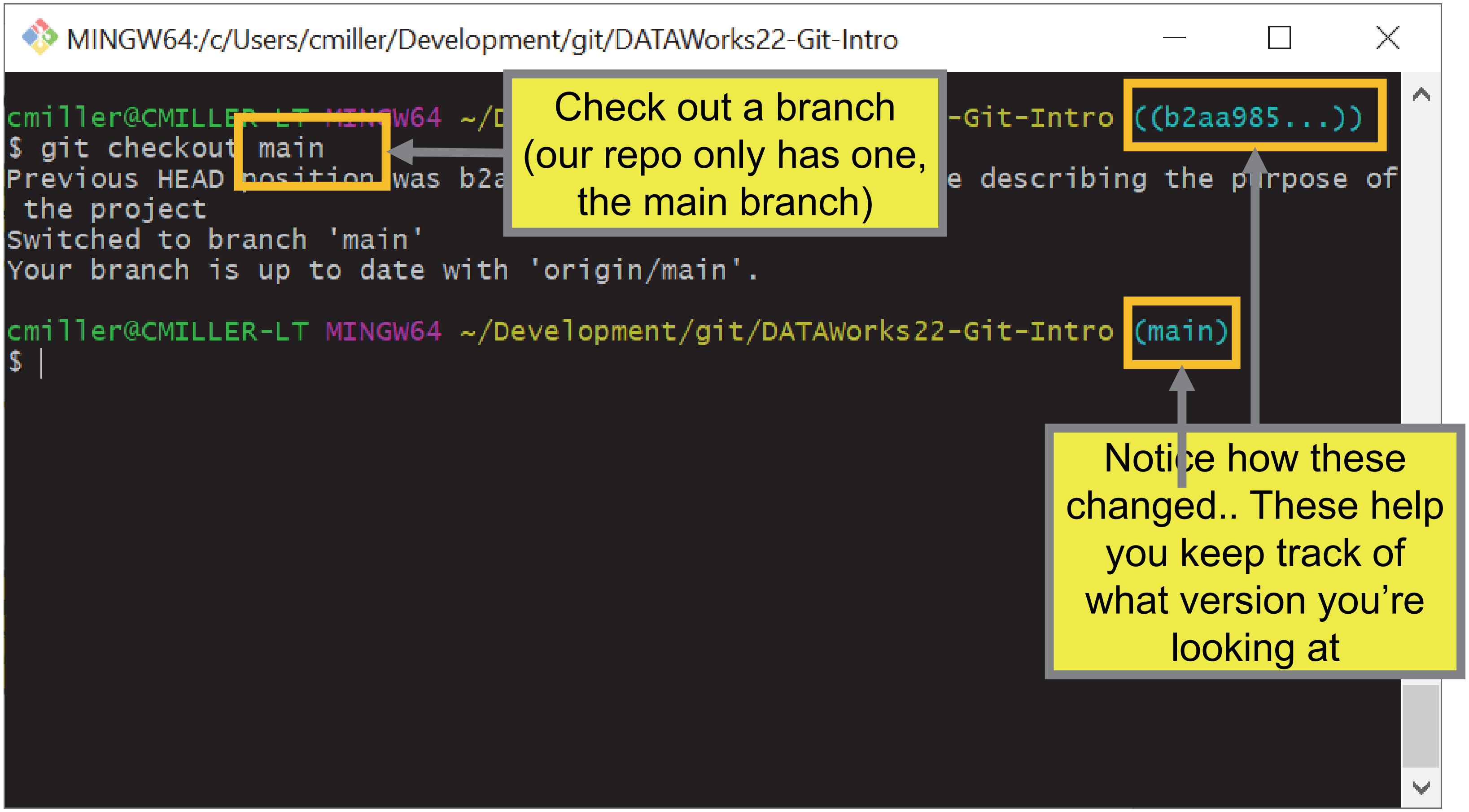
If the old commit had files that have since been deleted, they would have reappeared.

“So how do I go back to my most recent commit?”


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro ((b2aa985...))
$ git checkout main
Previous HEAD position was b2aa985 Added a README.md file describing the purpose of
 the project
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ 
```

You check out the main branch

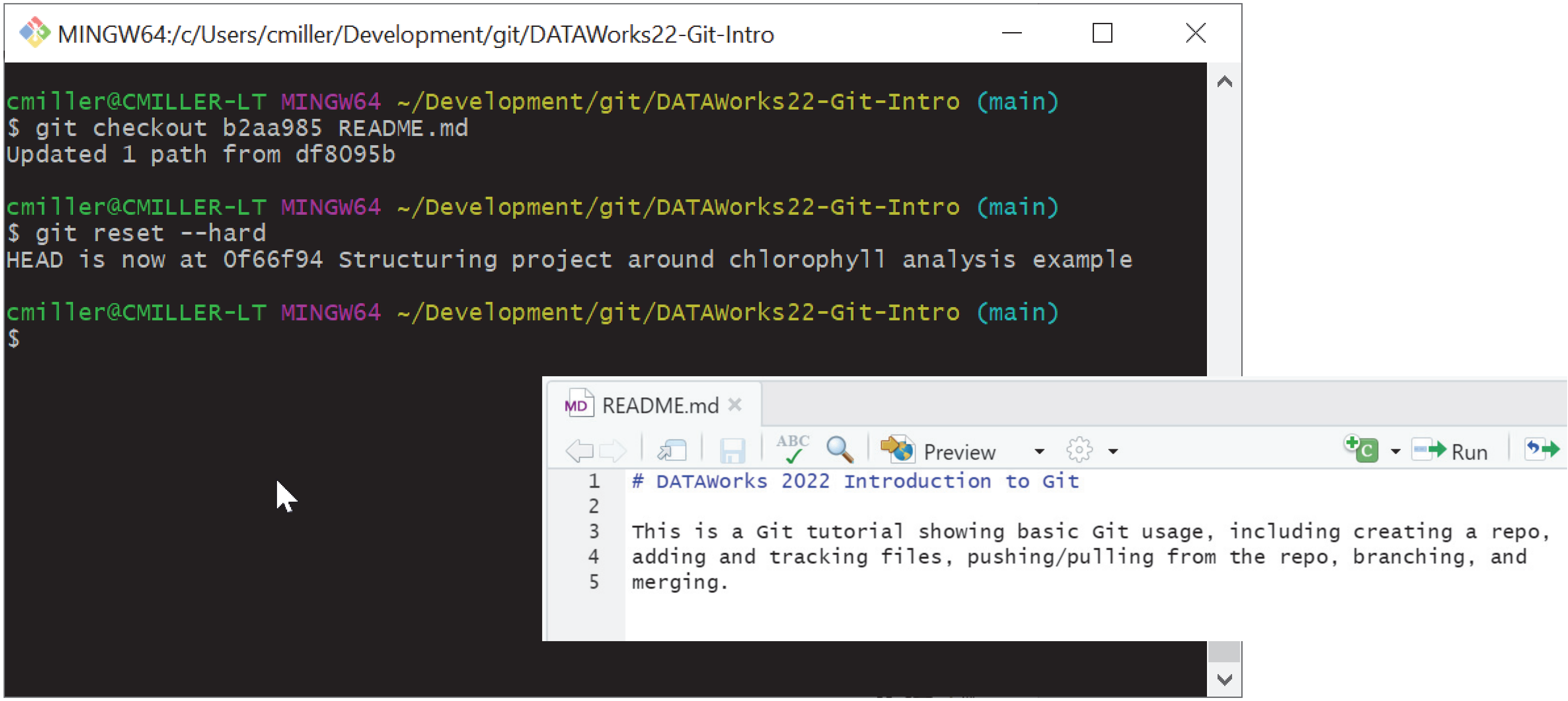

MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/D                           -Git-Intro ((b2aa985...))
$ git checkout main
Previous HEAD position was b2a                           e describing the purpose of
 the project
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ 
```

Check out a branch (our repo only has one, the main branch)

Notice how these changed.. These help you keep track of what version you're looking at

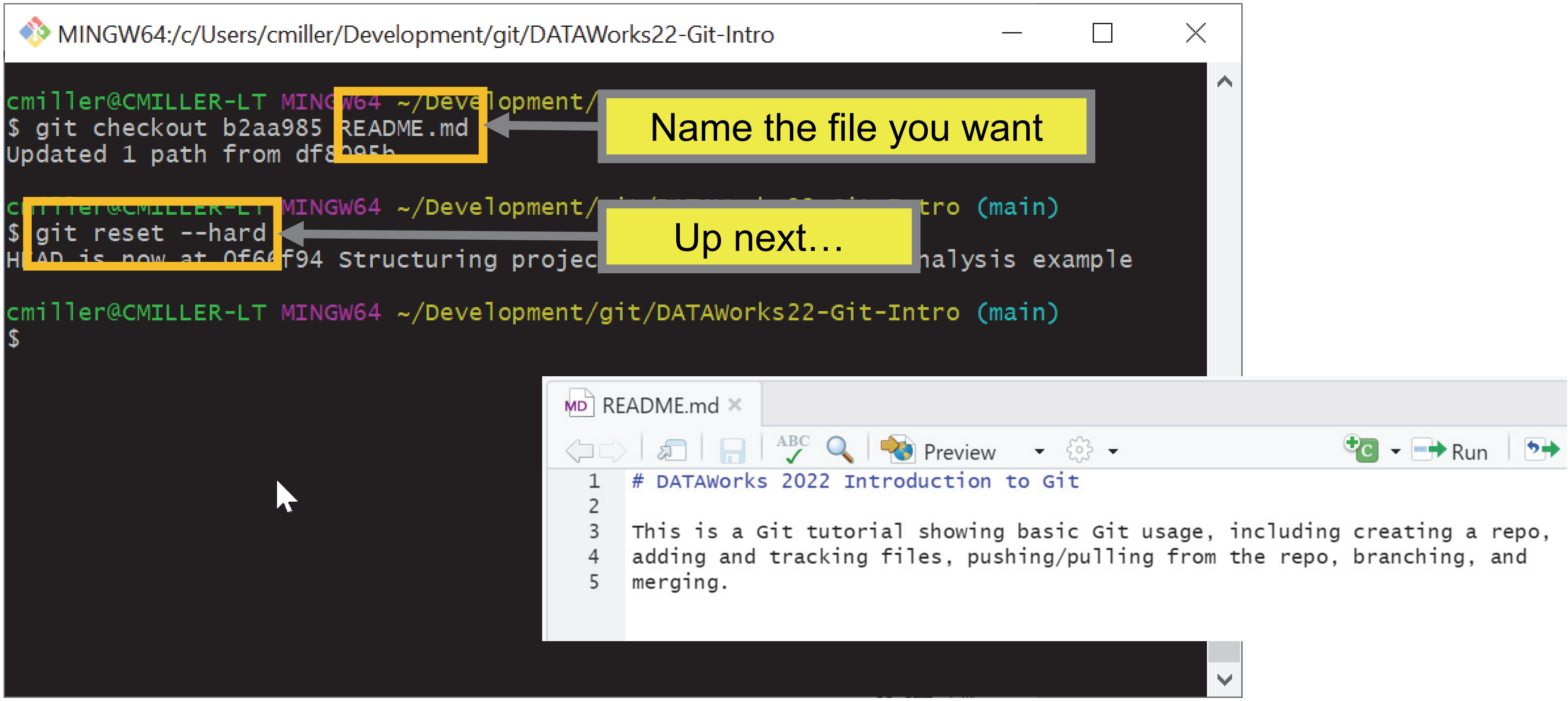
“Well, I only want one file from that commit.”


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
README.md
Preview
Run

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git checkout b2aa985 README.md
Updated 1 path from df8095b

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring project around chlorophyll analysis example

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$
```

```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.
```

Just say what file you want


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
README.md
Preview
Run

```
cmiller@CMILLER-LT MINGW64 ~/Development/
$ git checkout b2aa985 README.md
Updated 1 path from df8095b

cmiller@CMILLER-LT MINGW64 ~/Development/                    tro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring projec                    nalysis example

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$
```

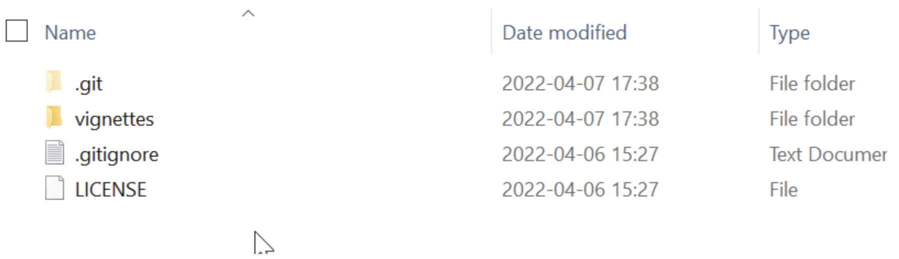
Name the file you want

Up next…

```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.
```

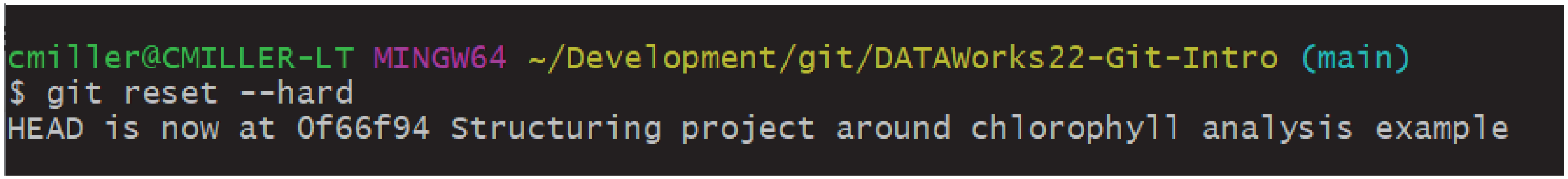
“Oh no! I deleted an important file! I want it back!”

1



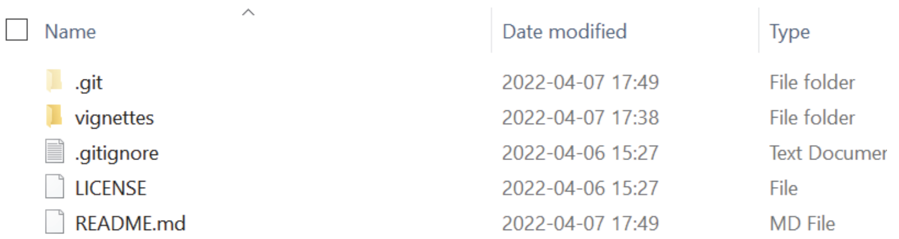
Name	Date modified	Type
.git	2022-04-07 17:38	File folder
vignettes	2022-04-07 17:38	File folder
.gitignore	2022-04-06 15:27	Text Documer
LICENSE	2022-04-06 15:27	File

2



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring project around chlorophyll analysis example
```

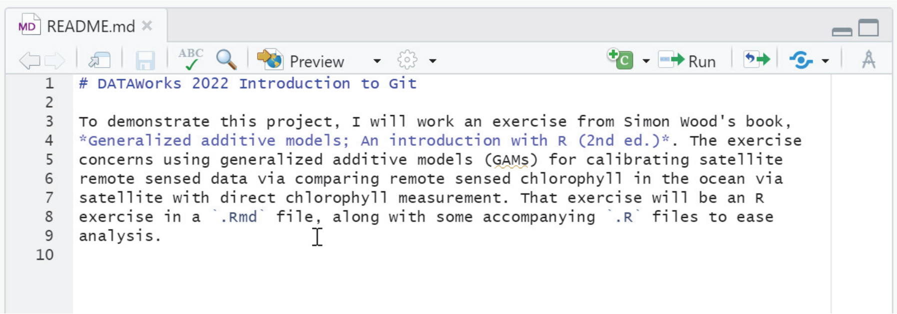
3



Name	Date modified	Type
.git	2022-04-07 17:49	File folder
vignettes	2022-04-07 17:38	File folder
.gitignore	2022-04-06 15:27	Text Documer
LICENSE	2022-04-06 15:27	File
README.md	2022-04-07 17:49	MD File

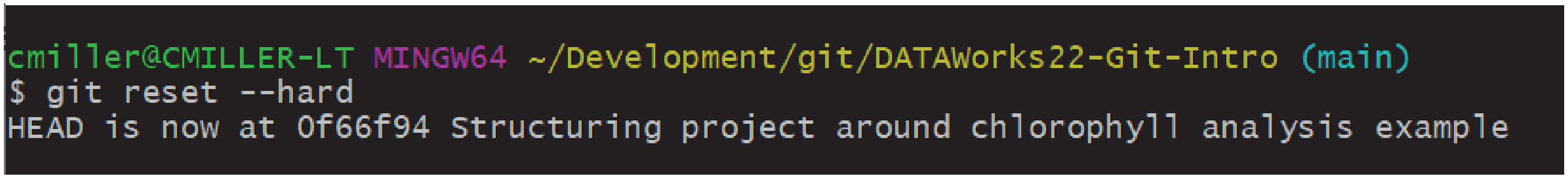
“Oh no! I changed and saved a file but all my changes were stupid! I want the old one back!”

1



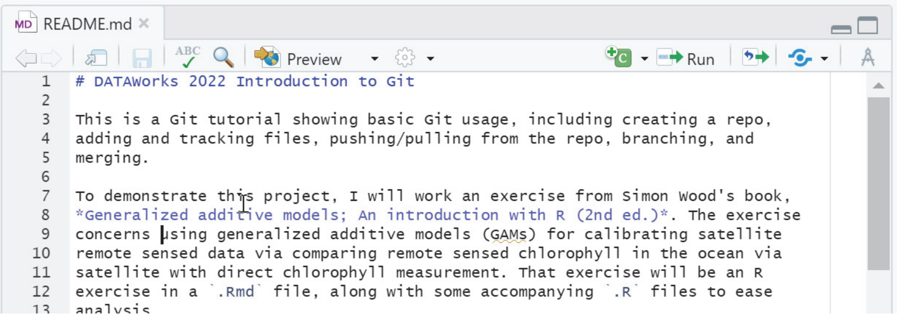
```
# DATAWorks 2022 Introduction to Git

To demonstrate this project, I will work an exercise from Simon Wood's book,
*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
concerns using generalized additive models (GAMs) for calibrating satellite
remote sensed data via comparing remote sensed chlorophyll in the ocean via
satellite with direct chlorophyll measurement. That exercise will be an R
exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
analysis.
```

2



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring project around chlorophyll analysis example
```

3



```
# DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo,
adding and tracking files, pushing/pulling from the repo, branching, and
merging.

To demonstrate this project, I will work an exercise from Simon Wood's book,
*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
concerns using generalized additive models (GAMs) for calibrating satellite
remote sensed data via comparing remote sensed chlorophyll in the ocean via
satellite with direct chlorophyll measurement. That exercise will be an R
exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
analysis
```

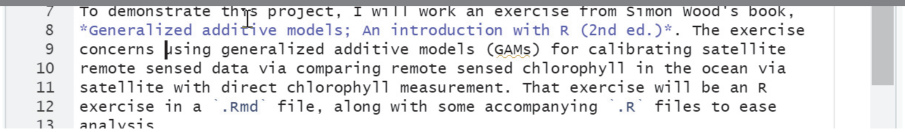
“Oh no! I changed and saved a file but all my changes were stupid! I want the old one back!”

WARNING!!!

`git reset --hard` will undo EVERYTHING! Be sure there are no changes in tracked files you actually want to keep!

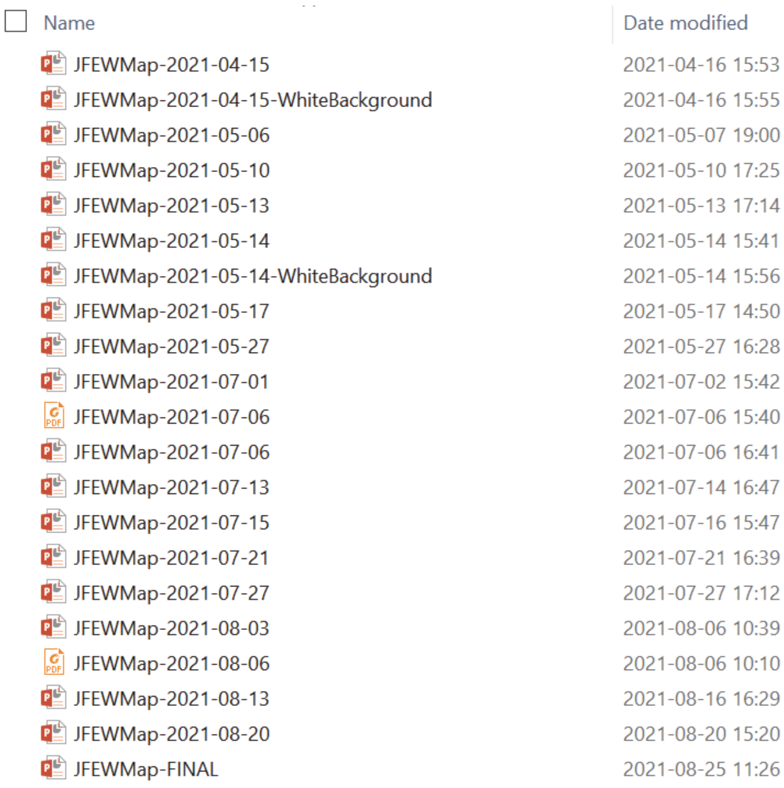
THIS IS THE ONLY WARNING YOU WILL EVER GET BEFORE LOSING WORK THIS WAY!

2



```
To demonstrate this project, I will work an exercise from Simon Wood's book,
*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
concerns using generalized additive models (GAMs) for calibrating satellite
remote sensed data via comparing remote sensed chlorophyll in the ocean via
satellite with direct chlorophyll measurement. That exercise will be an R
exercise in a `.Rmd` file, along with some accompanying `.R` files to ease
analysis
```

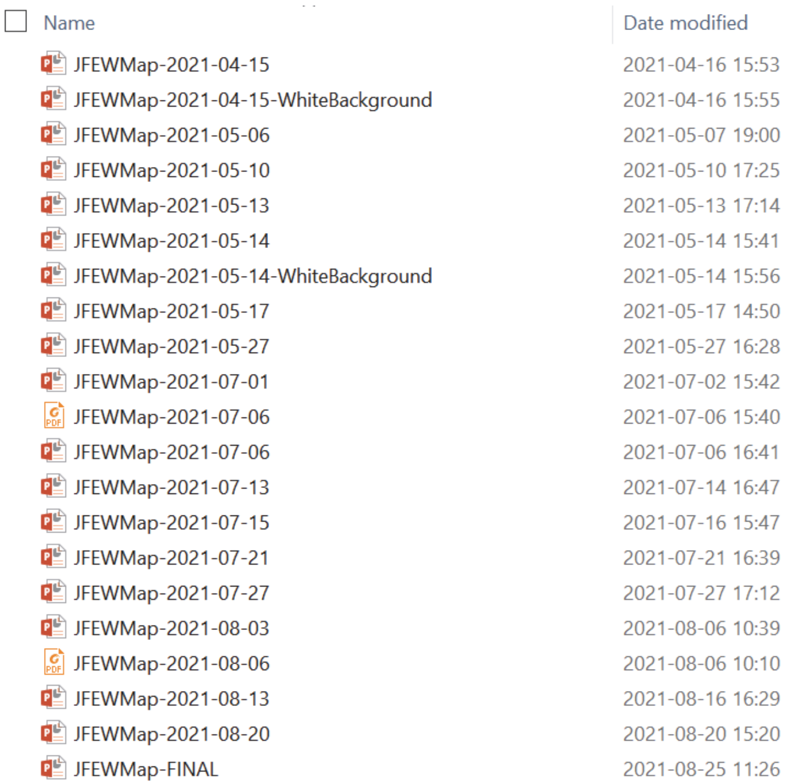
When using Git, you can now drop some terrible habits, like...



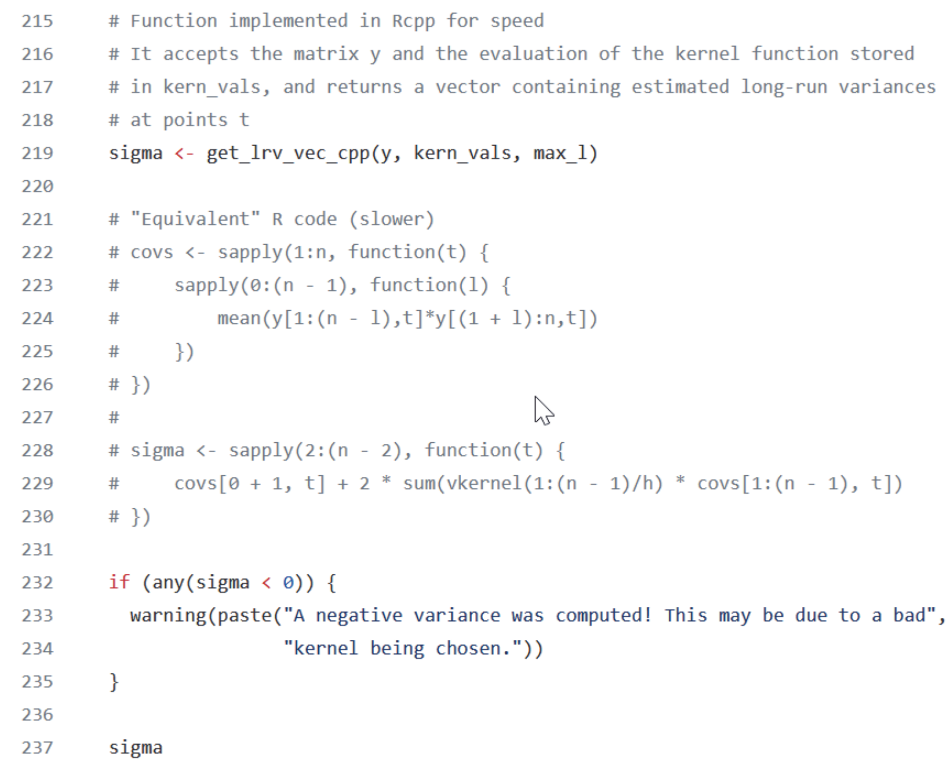
☐ Name	Date modified
JFEWMap-2021-04-15	2021-04-16 15:53
JFEWMap-2021-04-15-WhiteBackground	2021-04-16 15:55
JFEWMap-2021-05-06	2021-05-07 19:00
JFEWMap-2021-05-10	2021-05-10 17:25
JFEWMap-2021-05-13	2021-05-13 17:14
JFEWMap-2021-05-14	2021-05-14 15:41
JFEWMap-2021-05-14-WhiteBackground	2021-05-14 15:56
JFEWMap-2021-05-17	2021-05-17 14:50
JFEWMap-2021-05-27	2021-05-27 16:28
JFEWMap-2021-07-01	2021-07-02 15:42
JFEWMap-2021-07-06	2021-07-06 15:40
JFEWMap-2021-07-06	2021-07-06 16:41
JFEWMap-2021-07-13	2021-07-14 16:47
JFEWMap-2021-07-15	2021-07-16 15:47
JFEWMap-2021-07-21	2021-07-21 16:39
JFEWMap-2021-07-27	2021-07-27 17:12
JFEWMap-2021-08-03	2021-08-06 10:39
JFEWMap-2021-08-06	2021-08-06 10:10
JFEWMap-2021-08-13	2021-08-16 16:29
JFEWMap-2021-08-20	2021-08-20 15:20
JFEWMap-FINAL	2021-08-25 11:26

Using filenames to keep a history…

When using Git, you can now drop some terrible habits, like...



☐ Name	Date modified
JFEWMap-2021-04-15	2021-04-16 15:53
JFEWMap-2021-04-15-WhiteBackground	2021-04-16 15:55
JFEWMap-2021-05-06	2021-05-07 19:00
JFEWMap-2021-05-10	2021-05-10 17:25
JFEWMap-2021-05-13	2021-05-13 17:14
JFEWMap-2021-05-14	2021-05-14 15:41
JFEWMap-2021-05-14-WhiteBackground	2021-05-14 15:56
JFEWMap-2021-05-17	2021-05-17 14:50
JFEWMap-2021-05-27	2021-05-27 16:28
JFEWMap-2021-07-01	2021-07-02 15:42
JFEWMap-2021-07-06	2021-07-06 15:40
JFEWMap-2021-07-06	2021-07-06 16:41
JFEWMap-2021-07-13	2021-07-14 16:47
JFEWMap-2021-07-15	2021-07-16 15:47
JFEWMap-2021-07-21	2021-07-21 16:39
JFEWMap-2021-07-27	2021-07-27 17:12
JFEWMap-2021-08-03	2021-08-06 10:39
JFEWMap-2021-08-06	2021-08-06 10:10
JFEWMap-2021-08-13	2021-08-16 16:29
JFEWMap-2021-08-20	2021-08-20 15:20
JFEWMap-FINAL	2021-08-25 11:26

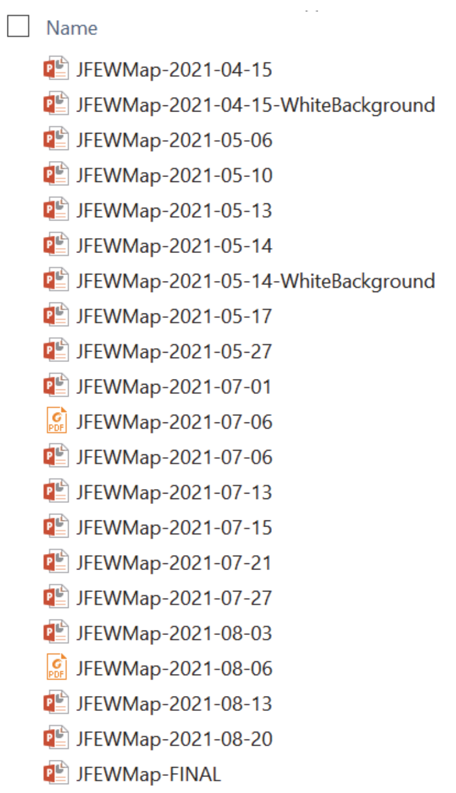


```r
# Function implemented in Rcpp for speed
# It accepts the matrix y and the evaluation of the kernel function stored
# in kern_vals, and returns a vector containing estimated long-run variances
# at points t
sigma <- get_lrv_vec_cpp(y, kern_vals, max_l)

# "Equivalent" R code (slower)
# covs <- sapply(1:n, function(t) {
#     sapply(0:(n - 1), function(l) {
#         mean(y[1:(n - l),t]*y[(1 + l):n,t])
#     })
# })
#
# sigma <- sapply(2:(n - 2), function(t) {
#     covs[0 + 1, t] + 2 * sum(vkernel(1:(n - 1)/h) * covs[1:(n - 1), t])
# })

if (any(sigma < 0)) {
  warning(paste("A negative variance was computed! This may be due to a bad",
                "kernel being chosen."))
}

sigma
```

Using filenames to keep a history…

Commenting out/keeping old useless code…

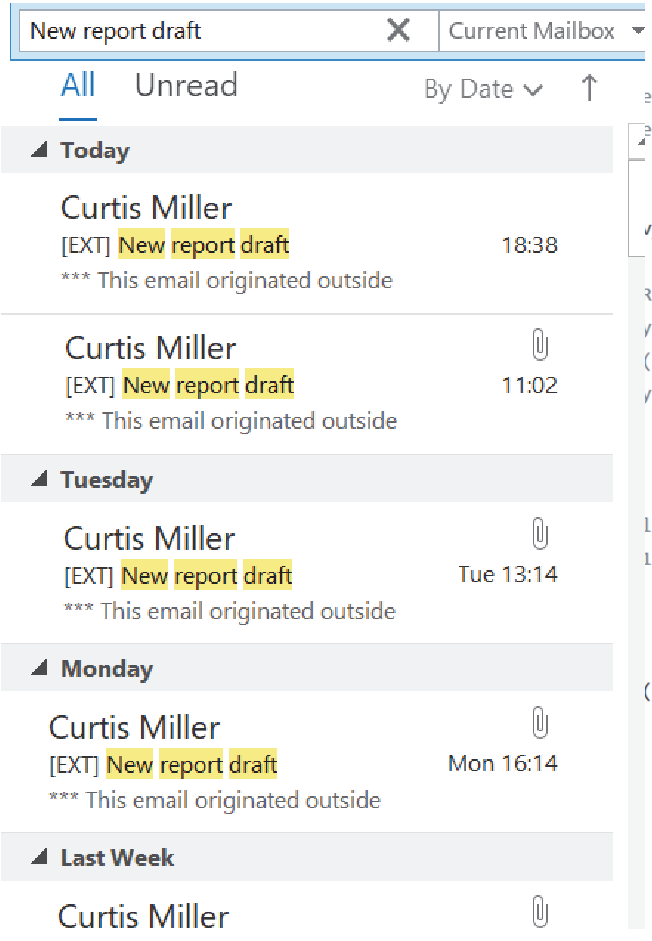
When using Git, you can now drop some terrible habits, like...



☐ Name

- JFEWMap-2021-04-15
- JFEWMap-2021-04-15-WhiteBackground
- JFEWMap-2021-05-06
- JFEWMap-2021-05-10
- JFEWMap-2021-05-13
- JFEWMap-2021-05-14
- JFEWMap-2021-05-14-WhiteBackground
- JFEWMap-2021-05-17
- JFEWMap-2021-05-27
- JFEWMap-2021-07-01
- JFEWMap-2021-07-06
- JFEWMap-2021-07-06
- JFEWMap-2021-07-13
- JFEWMap-2021-07-15
- JFEWMap-2021-07-21
- JFEWMap-2021-07-27
- JFEWMap-2021-08-03
- JFEWMap-2021-08-06
- JFEWMap-2021-08-13
- JFEWMap-2021-08-20
- JFEWMap-FINAL

Using filenames to keep a history…



Today

Curtis Miller
[EXT] New report draft 18:38
*** This email originated outside

Curtis Miller
[EXT] New report draft 11:02
*** This email originated outside

Tuesday

Curtis Miller
[EXT] New report draft Tue 13:14
*** This email originated outside

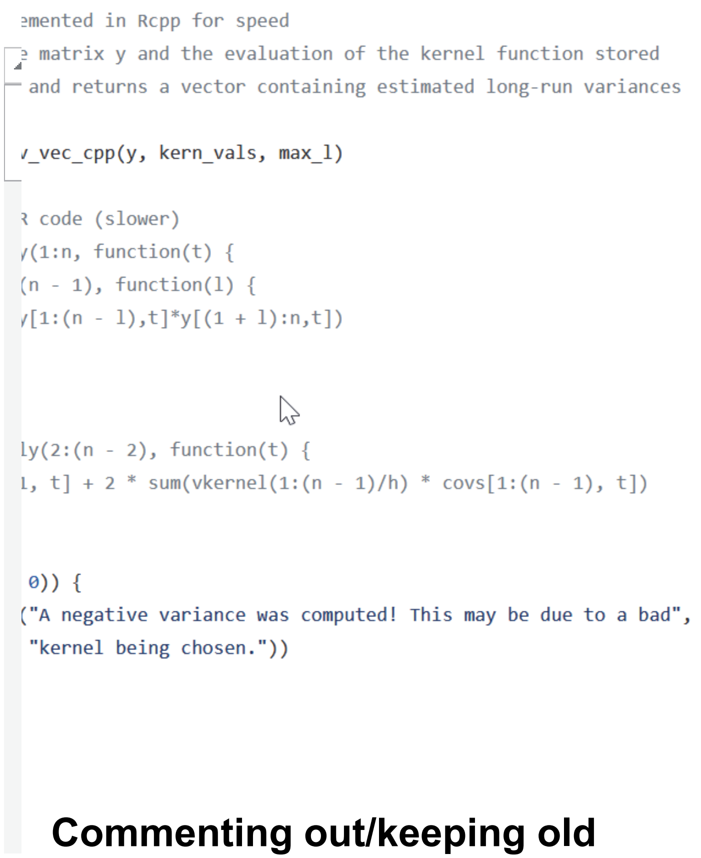
Monday

Curtis Miller
[EXT] New report draft Mon 16:14
*** This email originated outside

Last Week

Curtis Miller

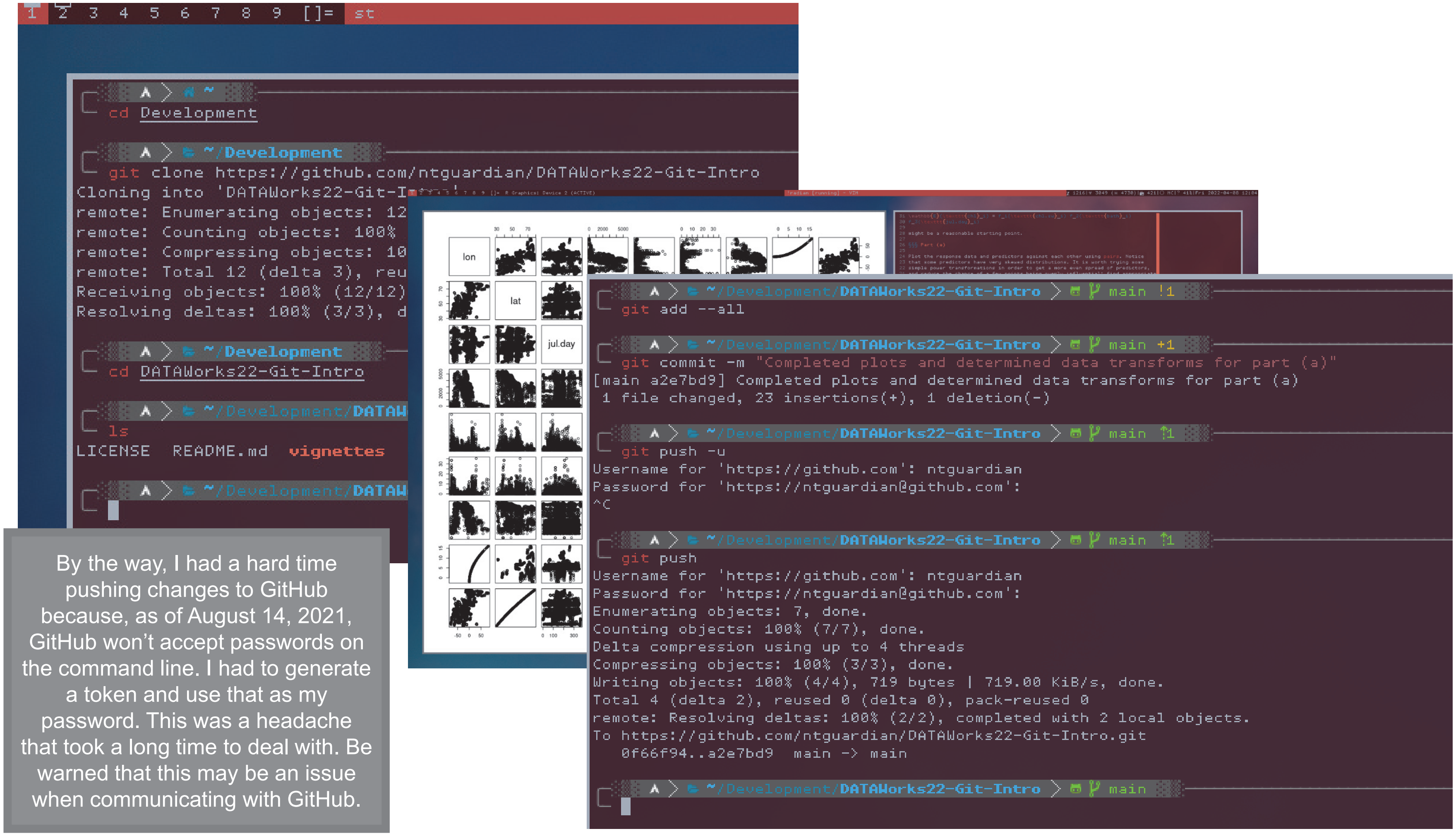
Using e-mail to manage collaboration/records…



```
emented in Rcpp for speed
 matrix y and the evaluation of the kernel function stored
 and returns a vector containing estimated long-run variances

_vec_cpp(y, kern_vals, max_l)

 code (slower)
(1:n, function(t) {
(n - 1), function(l) {
[1:(n - 1),t]*y[(1 + l):n,t])



ly(2:(n - 2), function(t) {
, t] + 2 * sum(vkernel(1:(n - 1)/h) * covs[1:(n - 1), t])



0)) {
("A negative variance was computed! This may be due to a bad",
 "kernel being chosen."))
```

Commenting out/keeping old useless code…

A collaborator just did some stuff on their own computer…

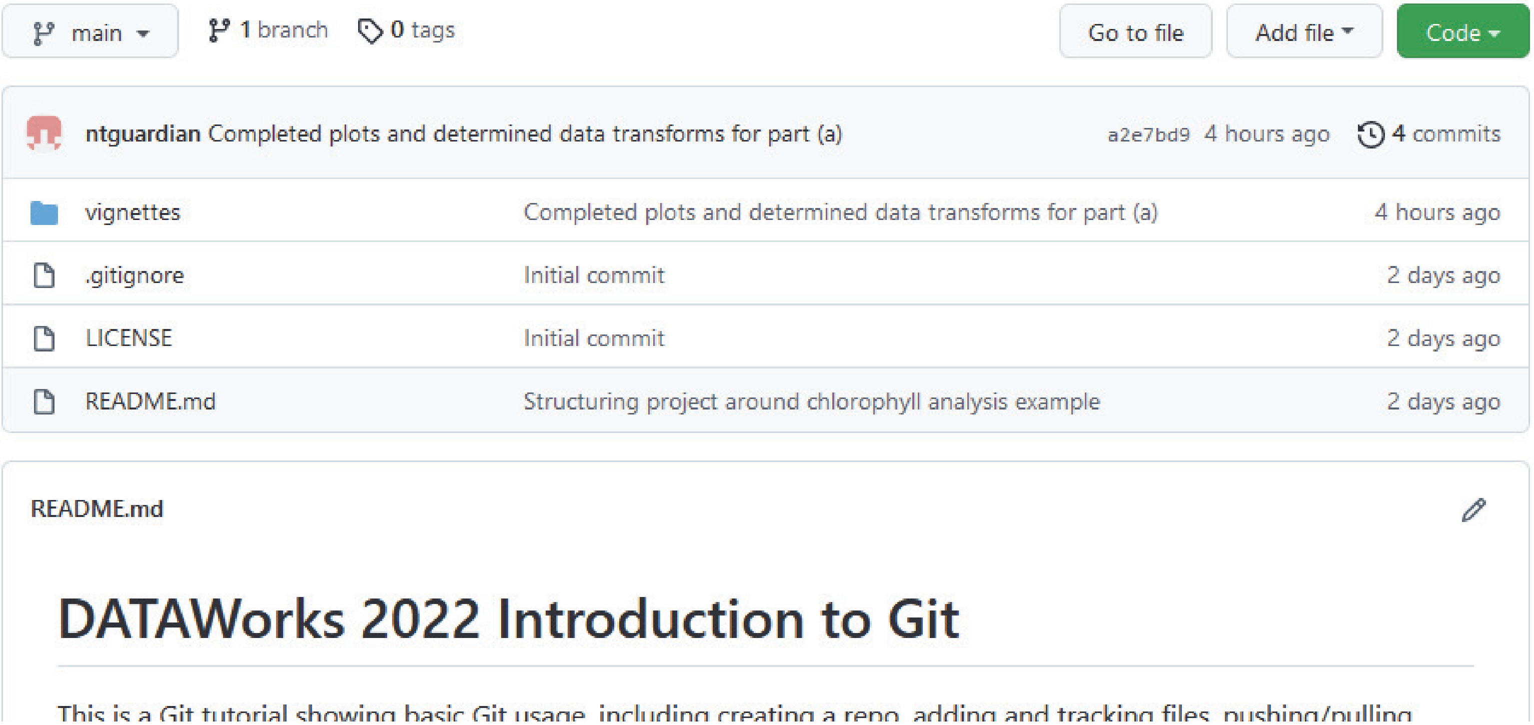


```
1  2  3  4  5  6  7  8  9  []=  st

 ~
cd Development

 ~/Development
git clone https://github.com/ntguardian/DATAWorks22-Git-Intro
Cloning into 'DATAWorks22-Git-I
remote: Enumerating objects: 12
remote: Counting objects: 100%
remote: Compressing objects: 10
remote: Total 12 (delta 3), reu
Receiving objects: 100% (12/12)
Resolving deltas: 100% (3/3), d

 ~/Development
cd DATAWorks22-Git-Intro

 ~/Development/DATAW
ls
LICENSE  README.md  vignettes

 ~/Development/DATAW
```

```
 ~/Development/DATAWorks22-Git-Intro   main !1
git add --all

 ~/Development/DATAWorks22-Git-Intro   main +1
git commit -m "Completed plots and determined data transforms for part (a)"
[main a2e7bd9] Completed plots and determined data transforms for part (a)
 1 file changed, 23 insertions(+), 1 deletion(-)

 ~/Development/DATAWorks22-Git-Intro   main ⇡1
git push -u
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
^C

 ~/Development/DATAWorks22-Git-Intro   main ⇡1
git push
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 719 bytes | 719.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   0f66f94..a2e7bd9  main -> main

 ~/Development/DATAWorks22-Git-Intro   main
```

By the way, I had a hard time pushing changes to GitHub because, as of August 14, 2021, GitHub won't accept passwords on the command line. I had to generate a token and use that as my password. This was a headache that took a long time to deal with. Be warned that this may be an issue when communicating with GitHub.

We can see these changes on GitHub and now need to get them locally



ntguardian Completed plots and determined data transforms for part (a) a2e7bd9 4 hours ago 4 commits

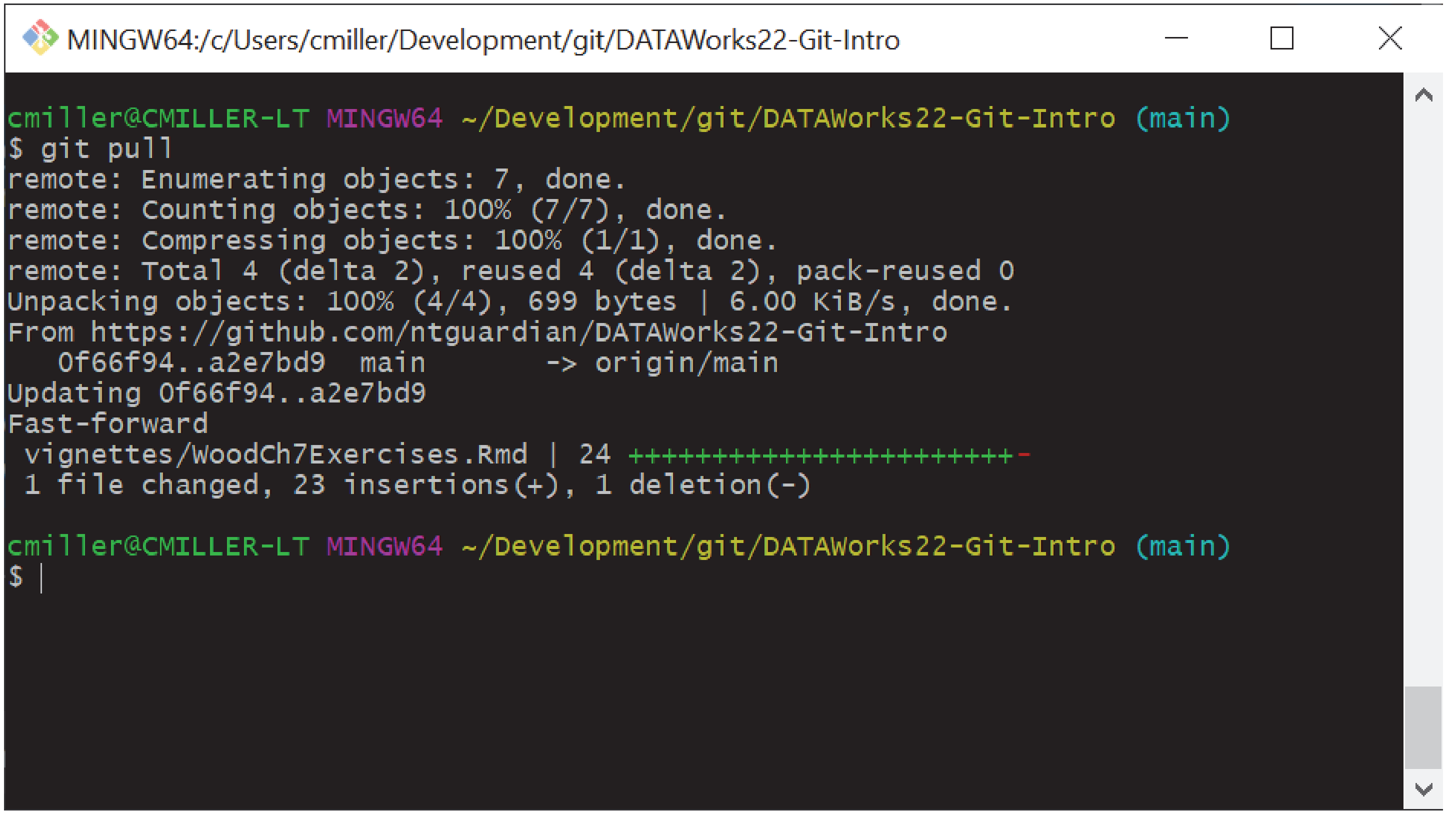
vignettes	Completed plots and determined data transforms for part (a)	4 hours ago
.gitignore	Initial commit	2 days ago
LICENSE	Initial commit	2 days ago
README.md	Structuring project around chlorophyll analysis example	2 days ago

README.md

DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling

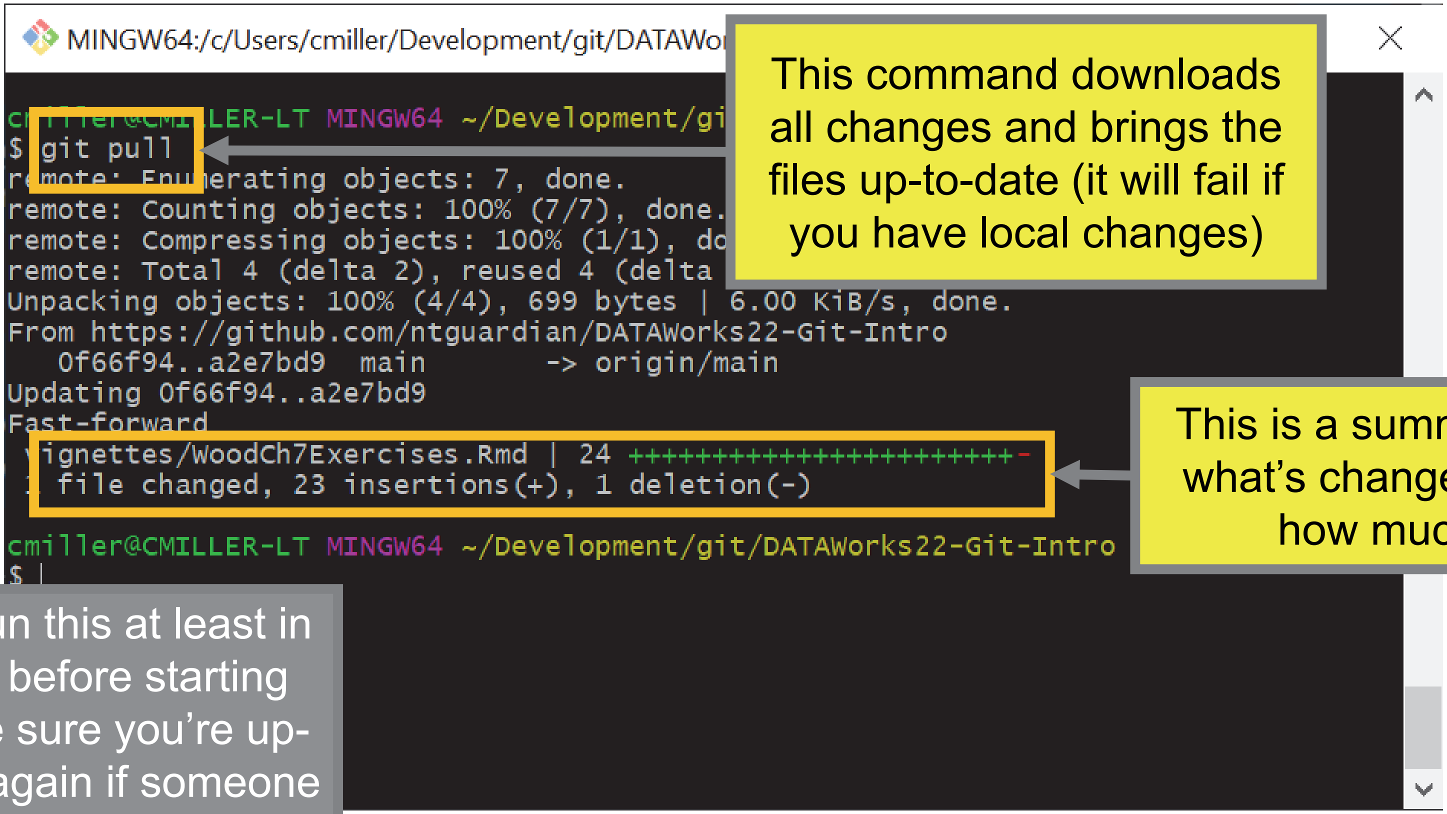
Use `git pull` to bring the local repo on your hard drive up-to-date



MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 2), reused 4 (delta 2), pack-reused 0
Unpacking objects: 100% (4/4), 699 bytes | 6.00 KiB/s, done.
From https://github.com/ntguardian/DATAWorks22-Git-Intro
   0f66f94..a2e7bd9  main       -> origin/main
Updating 0f66f94..a2e7bd9
Fast-forward
 vignettes/WoodCh7Exercises.Rmd | 24 +++++++++++++++++++++++-
 1 file changed, 23 insertions(+), 1 deletion(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ |
```

Use `git pull` to bring the local repo on your hard drive up-to-date



MINGW64:/c/Users/cmiller/Development/git/DATAWor

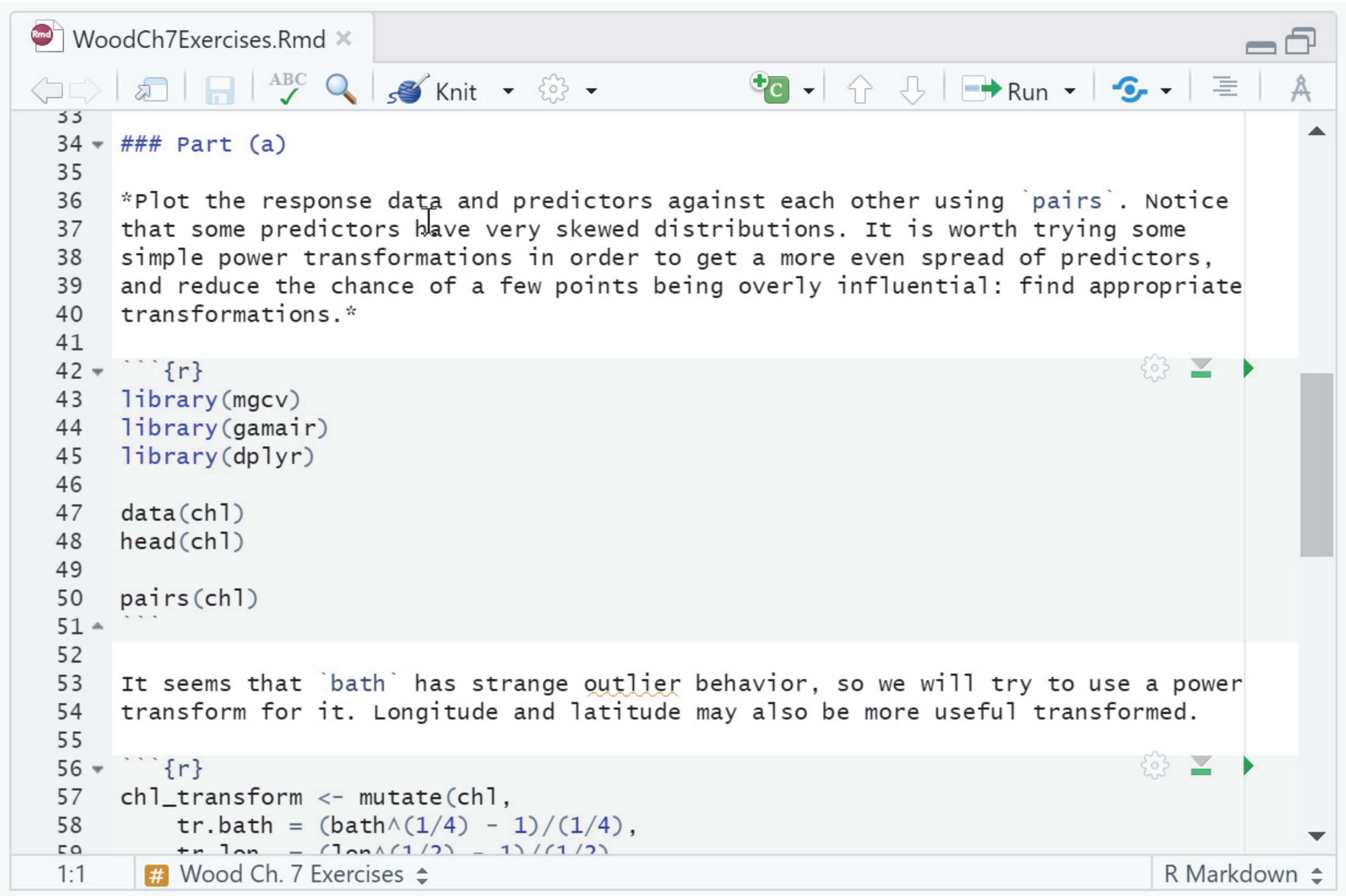
```
cmiller@CMILLER-LT MINGW64 ~/Development/gi
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), do
remote: Total 4 (delta 2), reused 4 (delta
Unpacking objects: 100% (4/4), 699 bytes | 6.00 KiB/s, done.
From https://github.com/ntguardian/DATAWorks22-Git-Intro
   0f66f94..a2e7bd9  main       -> origin/main
Updating 0f66f94..a2e7bd9
Fast-forward
 vignettes/WoodCh7Exercises.Rmd | 24 +++++++++++++++++++++++-
 1 file changed, 23 insertions(+), 1 deletion(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro
$
```

This command downloads all changes and brings the files up-to-date (it will fail if you have local changes)

This is a summary of what’s changed and how much

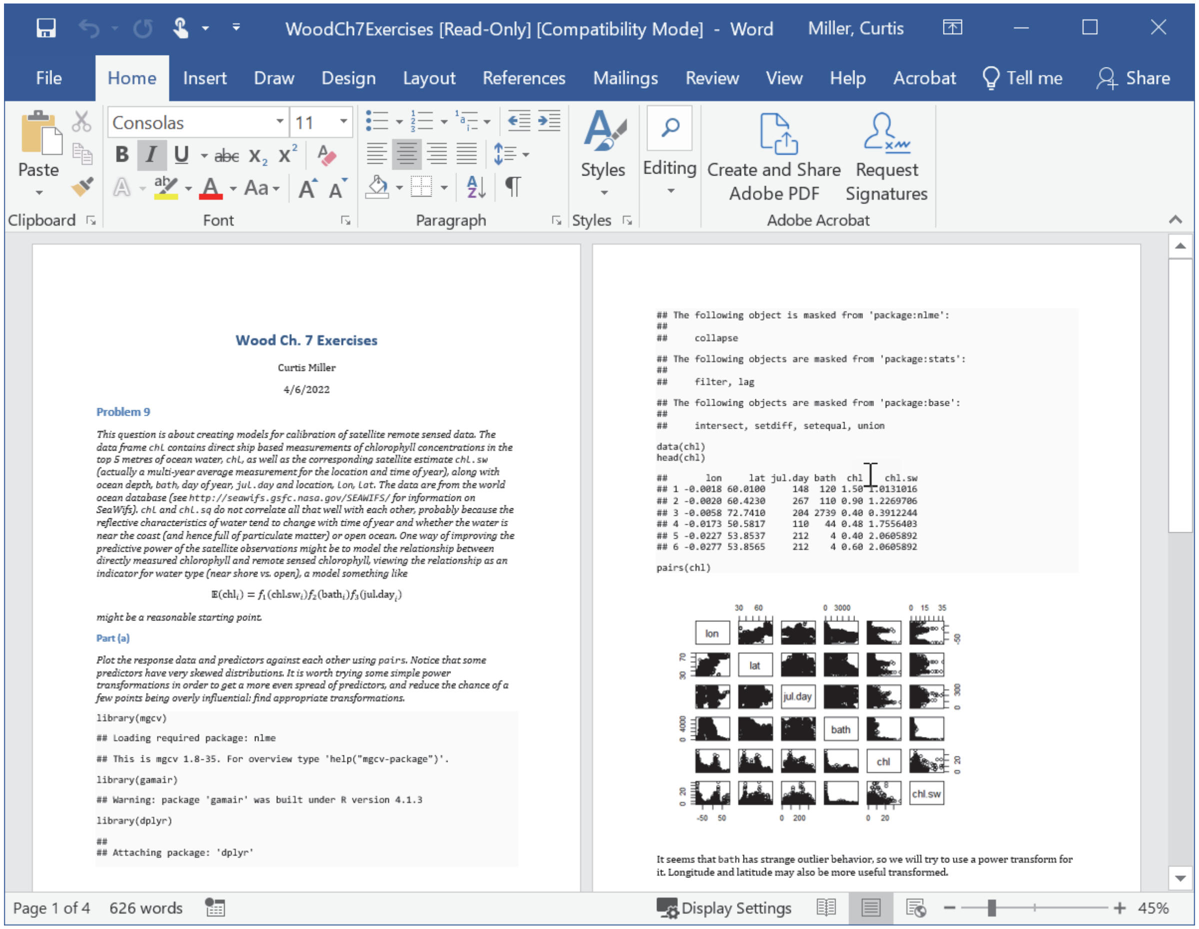
You should run this at least in the morning before starting work to make sure you’re up-to-date. Run again if someone just made changes needed immediately.

Now those changes are available to work with



~~~
### Part (a)

*Plot the response data and predictors against each other using `pairs`. Notice
that some predictors have very skewed distributions. It is worth trying some
simple power transformations in order to get a more even spread of predictors,
and reduce the chance of a few points being overly influential: find appropriate
transformations.*

```{r}
library(mgcv)
library(gamair)
library(dplyr)

data(chl)
head(chl)

pairs(chl)
```

It seems that `bath` has strange outlier behavior, so we will try to use a power
transform for it. Longitude and latitude may also be more useful transformed.

```{r}
chl_transform <- mutate(chl,
    tr.bath = (bath^(1/4) - 1)/(1/4),
~~~

We can convert that Rmd file into a Word file...


WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word
Miller, Curtis
File
Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Help
Acrobat
Tell me
Share
Paste
Clipboard
Consolas
11
Font
Paragraph
Styles
Editing
Create and Share Adobe PDF
Request Signatures
Adobe Acrobat
lon
lat
jul.day
bath
chl
chl.sw
30 60
0 3000
0 15 35
-50 50
0 200
0 20
Page 1 of 4
626 words
Display Settings
45%

Wood Ch. 7 Exercises

Curtis Miller

4/6/2022

Problem 9

This question is about creating models for calibration of satellite remote sensed data. The data frame `chl` *contains direct ship based measurements of chlorophyll concentrations in the top 5 metres of ocean water,* `chl`*, as well as the corresponding satellite estimate* `chl.sw` *(actually a multi-year average measurement for the location and time of year), along with ocean depth,* `bath`*, day of year,* `jul.day` *and location,* `lon`*,* `lat`*. The data are from the world ocean database (see* `http://seawifs.gsfc.nasa.gov/SEAWIFS/` *for information on SeaWifs).* `chl` *and* `chl.sq` *do not correlate all that well with each other, probably because the reflective characteristics of water tend to change with time of year and whether the water is near the coast (and hence full of particulate matter) or open ocean. One way of improving the predictive power of the satellite observations might be to model the relationship between directly measured chlorophyll and remote sensed chlorophyll, viewing the relationship as an indicator for water type (near shore vs. open), a model something like*

$$\mathbb{E}(\text{chl}_i) = f_1(\text{chl.sw}_i)f_2(\text{bath}_i)f_3(\text{jul.day}_i)$$

might be a reasonable starting point.

Part (a)

Plot the response data and predictors against each other using `pairs`*. Notice that some predictors have very skewed distributions. It is worth trying some simple power transformations in order to get a more even spread of predictors, and reduce the chance of a few points being overly influential: find appropriate transformations.*

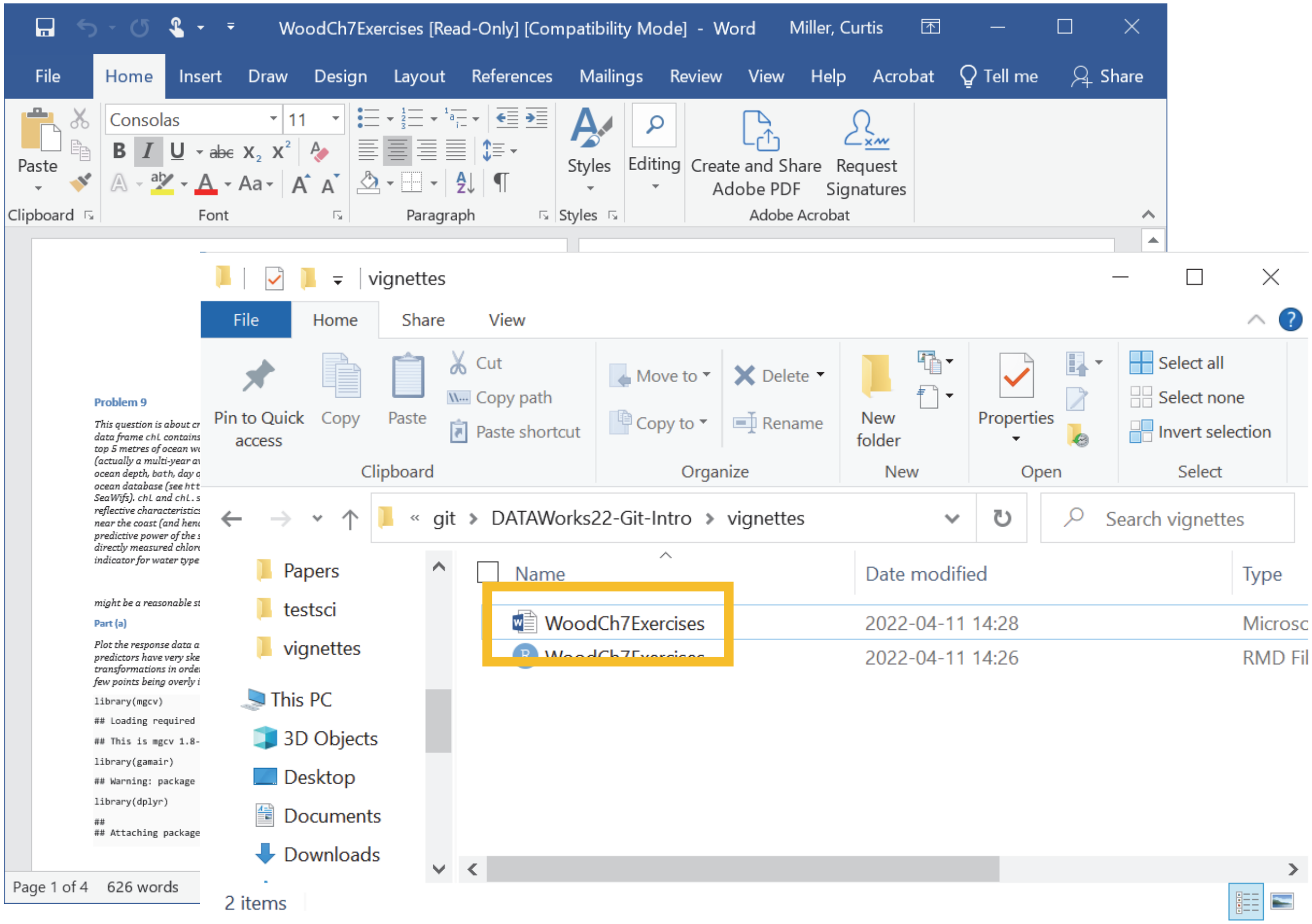
```
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-35. For overview type 'help("mgcv-package")'.

library(gamair)

## Warning: package 'gamair' was built under R version 4.1.3

library(dplyr)

##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
##
##     collapse

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

data(chl)
head(chl)

##       lon     lat jul.day bath  chl    chl.sw
## 1 -0.0018 60.0100     148  120 1.50 1.0131016
## 2 -0.0020 60.4230     267  110 0.90 1.2269706
## 3 -0.0058 72.7410     204 2739 0.40 0.3912244
## 4 -0.0173 50.5817     110   44 0.48 1.7556403
## 5 -0.0227 53.8537     212    4 0.40 2.0605892
## 6 -0.0277 53.8565     212    4 0.60 2.0605892

pairs(chl)
```

It seems that `bath` has strange outlier behavior, so we will try to use a power transform for it. Longitude and latitude may also be more useful transformed.

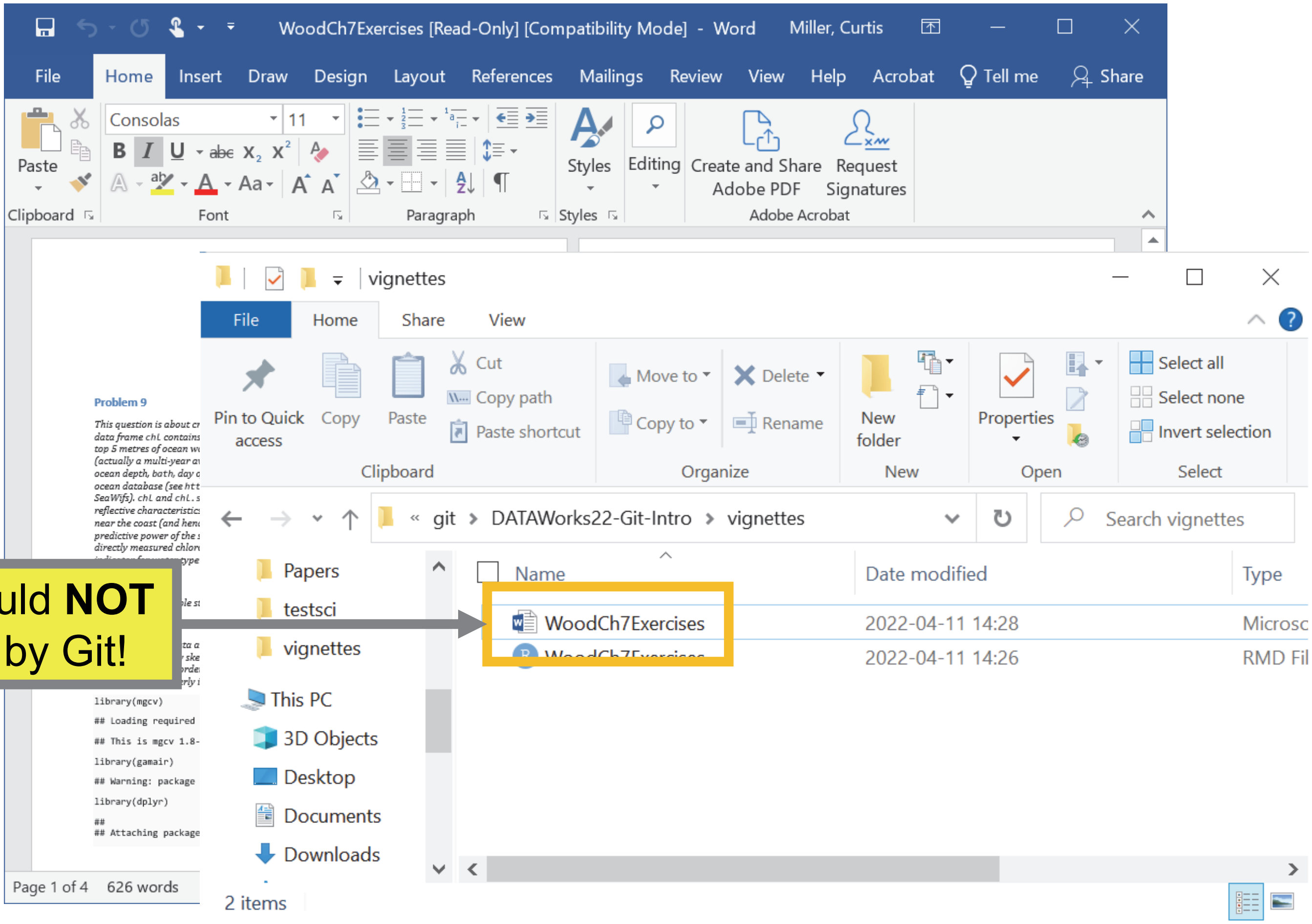
R Markdown is a format that can be translated into many other formats. Word is one of them, in addition to HTML and PDF.

… that then appears in the directory with the original file…


WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word
Miller, Curtis
File
Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Help
Acrobat
Tell me
Share
Consolas
11
Paste
Clipboard
Font
Paragraph
Styles
Editing
Create and Share Adobe PDF
Request Signatures
Adobe Acrobat
Page 1 of 4
626 words
vignettes
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Select all
Select none
Invert selection
Select
git › DATAWorks22-Git-Intro › vignettes
Search vignettes
Papers
testsci
vignettes
This PC
3D Objects
Desktop
Documents
Downloads
2 items

Name	Date modified	Type
WoodCh7Exercises	2022-04-11 14:28	Microsc
WoodCh7Exercises	2022-04-11 14:26	RMD Fil

… but we *don't* want to track this file


WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word
Miller, Curtis
File
Home
Insert
Draw
Design
Layout
References
Mailings
Review
View
Help
Acrobat
Tell me
Share
Consolas
11
Paste
Clipboard
Font
Paragraph
Styles
Editing
Create and Share Adobe PDF
Request Signatures
Adobe Acrobat
Problem 9
Page 1 of 4
626 words
vignettes
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New
Properties
Open
Select all
Select none
Invert selection
Select
« git › DATAWorks22-Git-Intro › vignettes
Search vignettes
Papers
testsci
vignettes
This PC
3D Objects
Desktop
Documents
Downloads
Name
Date modified
Type
2 items

Name	Date modified	Type
WoodCh7Exercises	2022-04-11 14:28	Microsc
WoodCh7Exercises	2022-04-11 14:26	RMD Fil

This file should **NOT** be tracked by Git!

Not every file in a repository should be tracked

TRACK

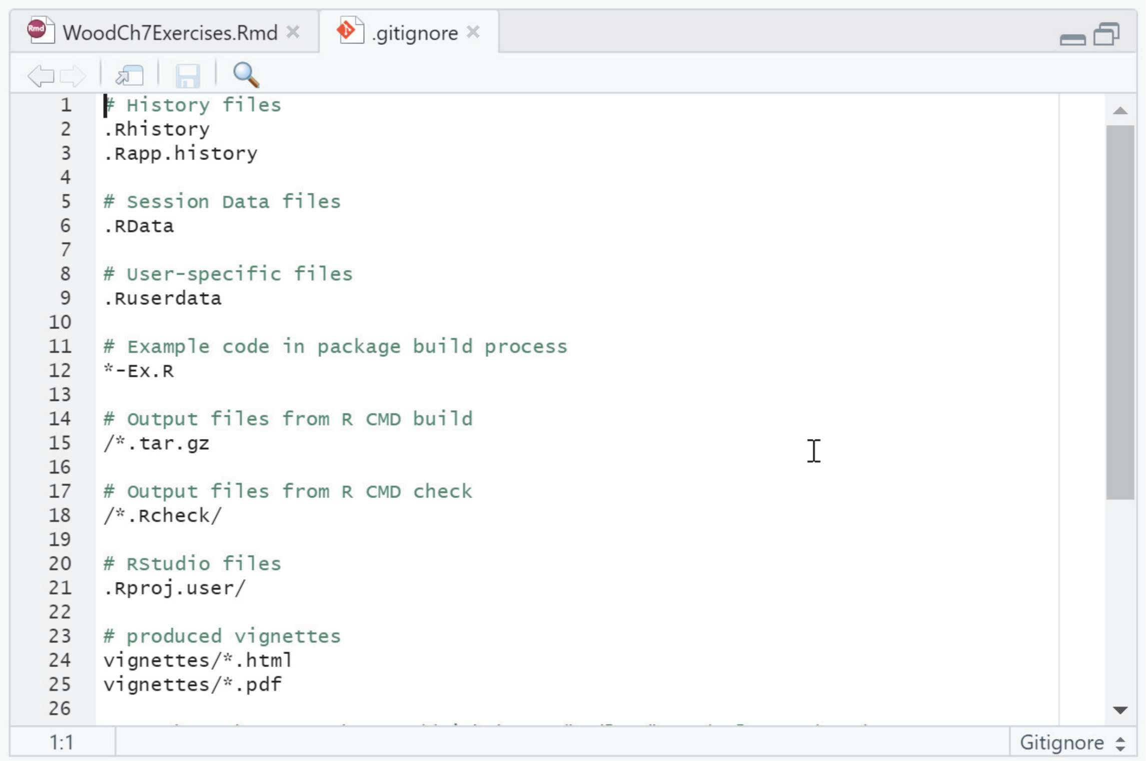
- Plain text files (txt, R, Rmd, py, …)
- Files directly edited
- Essential files that are hard to generate

AVOID

- Binary files (exe, docx, xlsx)
- Files generated by other files
- Irrelevant files

** These are guidelines and violating them may be reasonable*

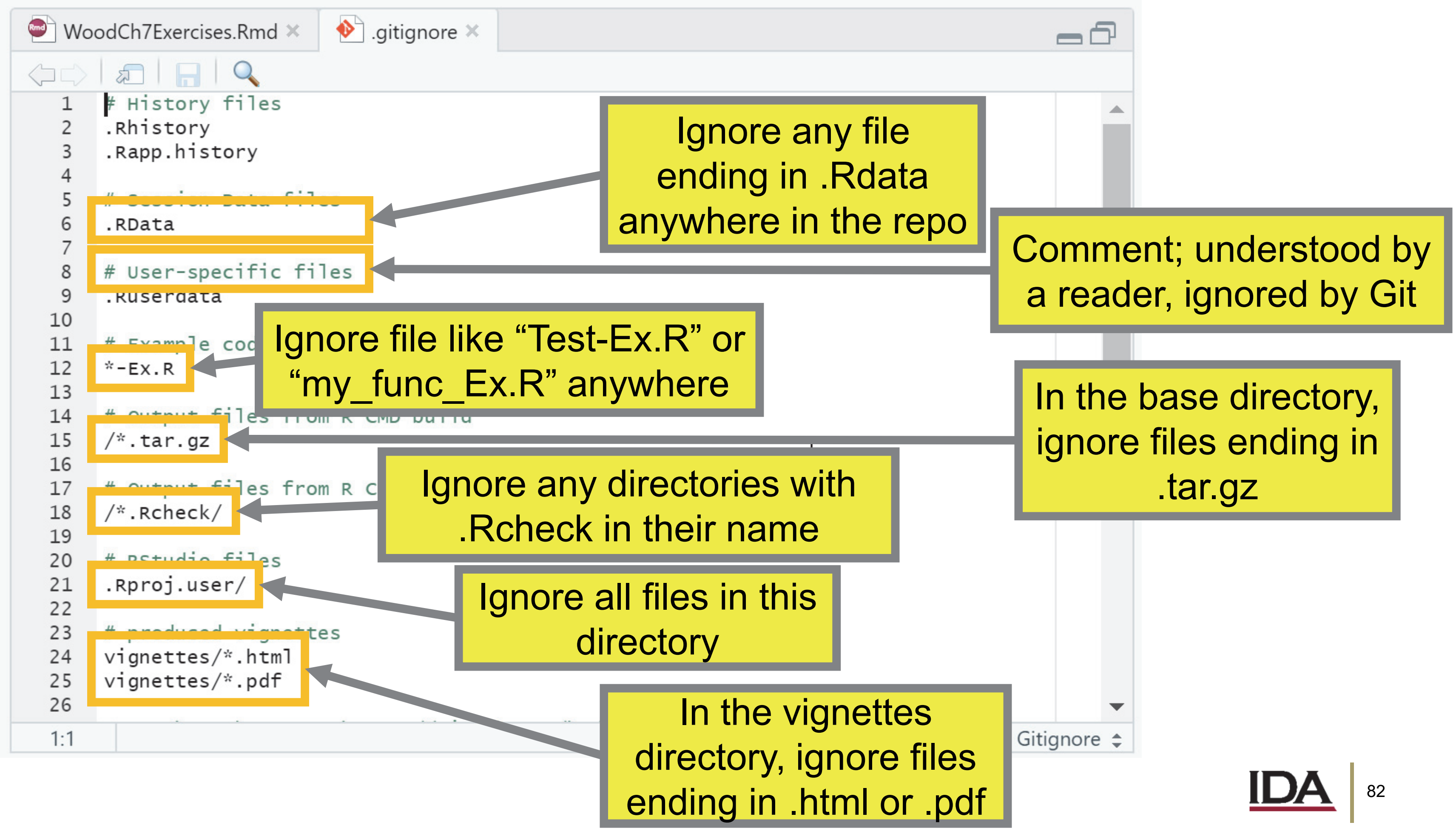
Use the .gitignore file to describe files Git should ignore when adding files to track changes



```
# History files
.Rhistory
.Rapp.history

# Session Data files
.RData

# User-specific files
.Ruserdata

# Example code in package build process
*-Ex.R

# Output files from R CMD build
/*.tar.gz

# Output files from R CMD check
/*.Rcheck/

# RStudio files
.Rproj.user/

# produced vignettes
vignettes/*.html
vignettes/*.pdf

```

- .gitignore describes files that, by default, should not be tracked in commits
- .gitignore is a plain text file containing descriptions of files that should be ignored
- File names and locations can be entered manually
- Pattern can also be used to describe classes of files to be ignored; if a file name matches the pattern the file will be ignored
- See https://git-scm.com/docs/gitignore for more information on writing a .gitignore file

GitHub already made a .gitignore file with defaults



```
# History files
.Rhistory
.Rapp.history

# Session Data files
.RData

# User-specific files
.Ruserdata

# Example coc
*-Ex.R

# Output files from R CMD build
/*.tar.gz

# Output files from R C
/*.Rcheck/

# RStudio files
.Rproj.user/

# produced vignettes
vignettes/*.html
vignettes/*.pdf

```

Ignore any file ending in .Rdata anywhere in the repo

Comment; understood by a reader, ignored by Git

Ignore file like “Test-Ex.R” or “my_func_Ex.R” anywhere

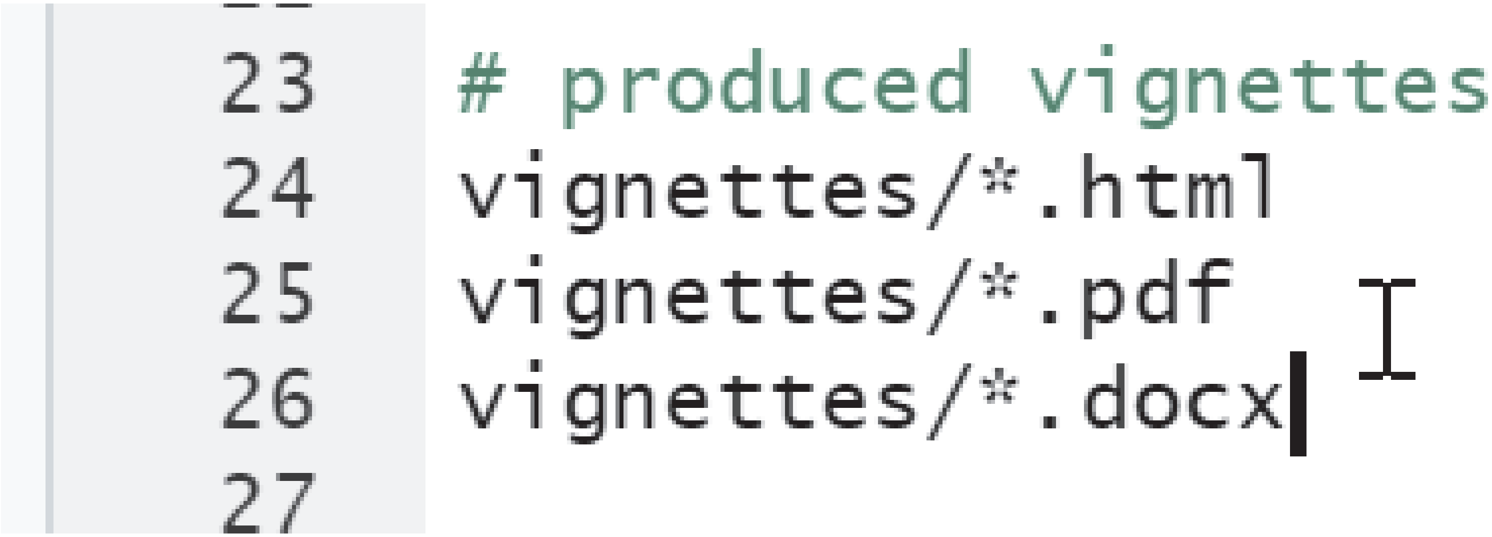
In the base directory, ignore files ending in .tar.gz

Ignore any directories with .Rcheck in their name

Ignore all files in this directory

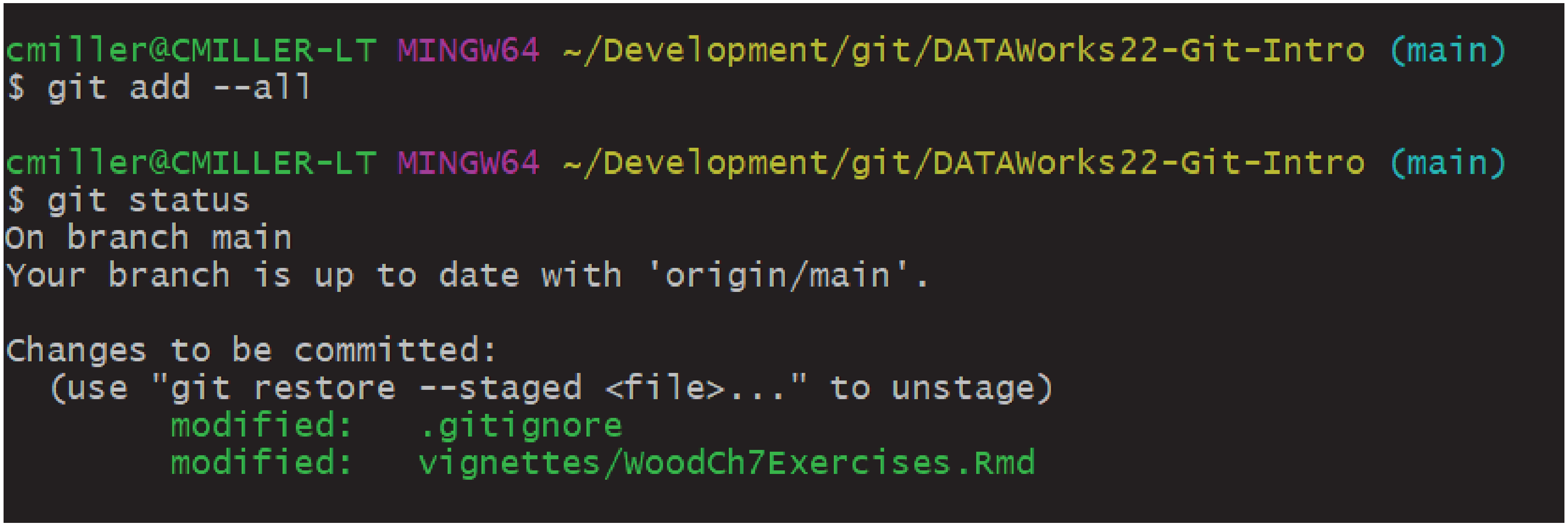
In the vignettes directory, ignore files ending in .html or .pdf

Now just add a line to ignore Word (.docx) files



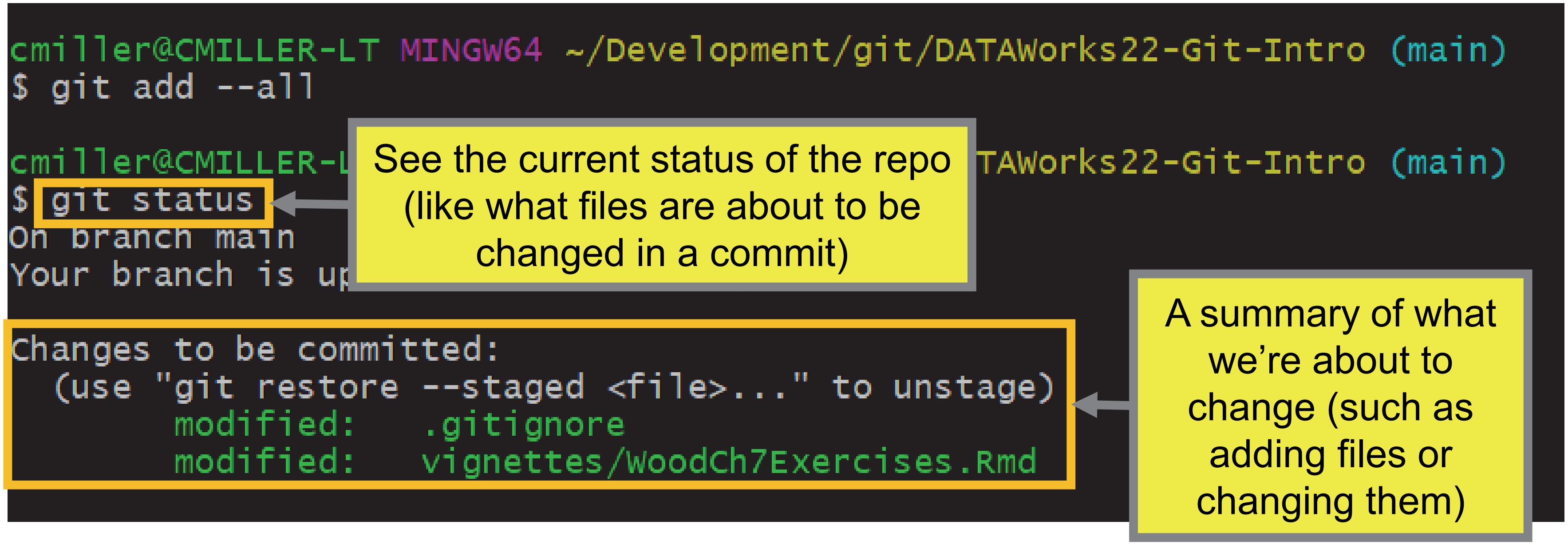
```
# produced vignettes
vignettes/*.html
vignettes/*.pdf
vignettes/*.docx

```

`git status` shows us what changes we are about to commit



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore
        modified:   vignettes/WoodCh7Exercises.Rmd
```

Notice that our Word file was not added with `git add --all`

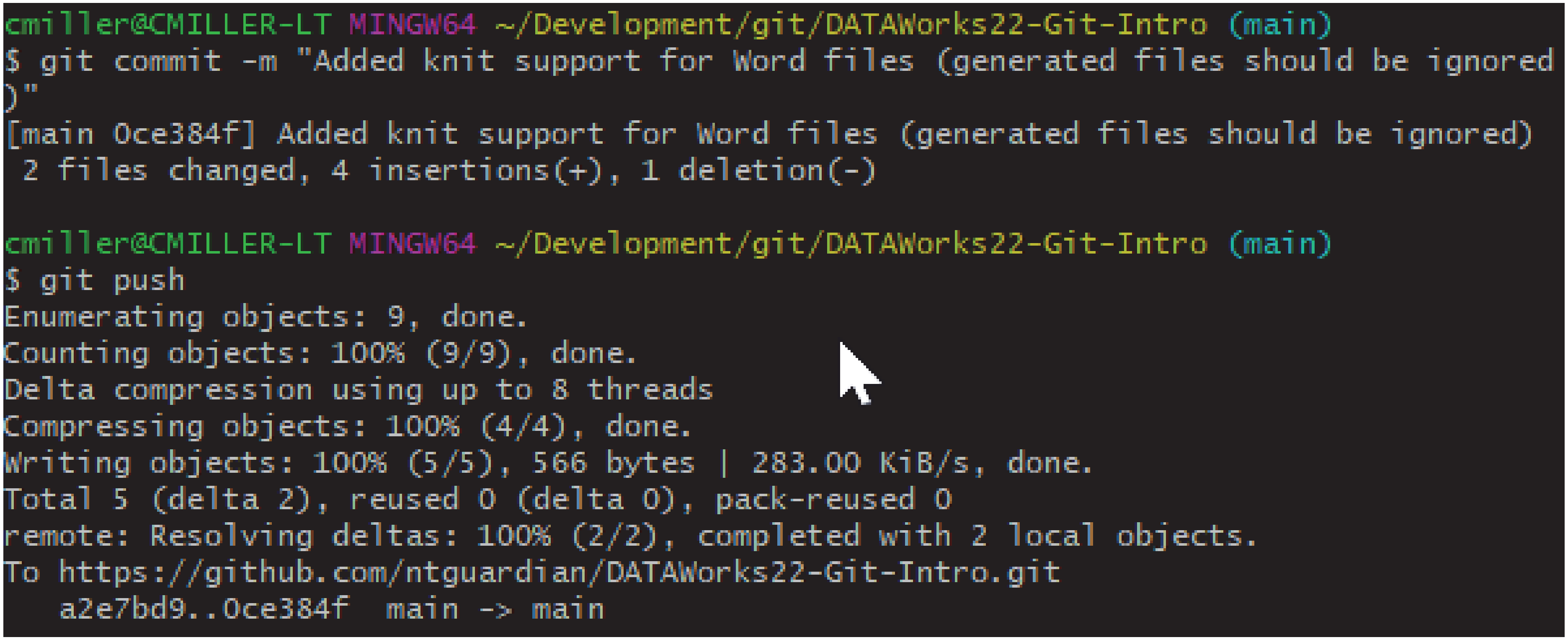


```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git add --all

cmiller@CMILLER-L                                  TAWorks22-Git-Intro (main)
$ git status
On branch main
Your branch is up

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore
        modified:   vignettes/WoodCh7Exercises.Rmd
```

See the current status of the repo (like what files are about to be changed in a commit)

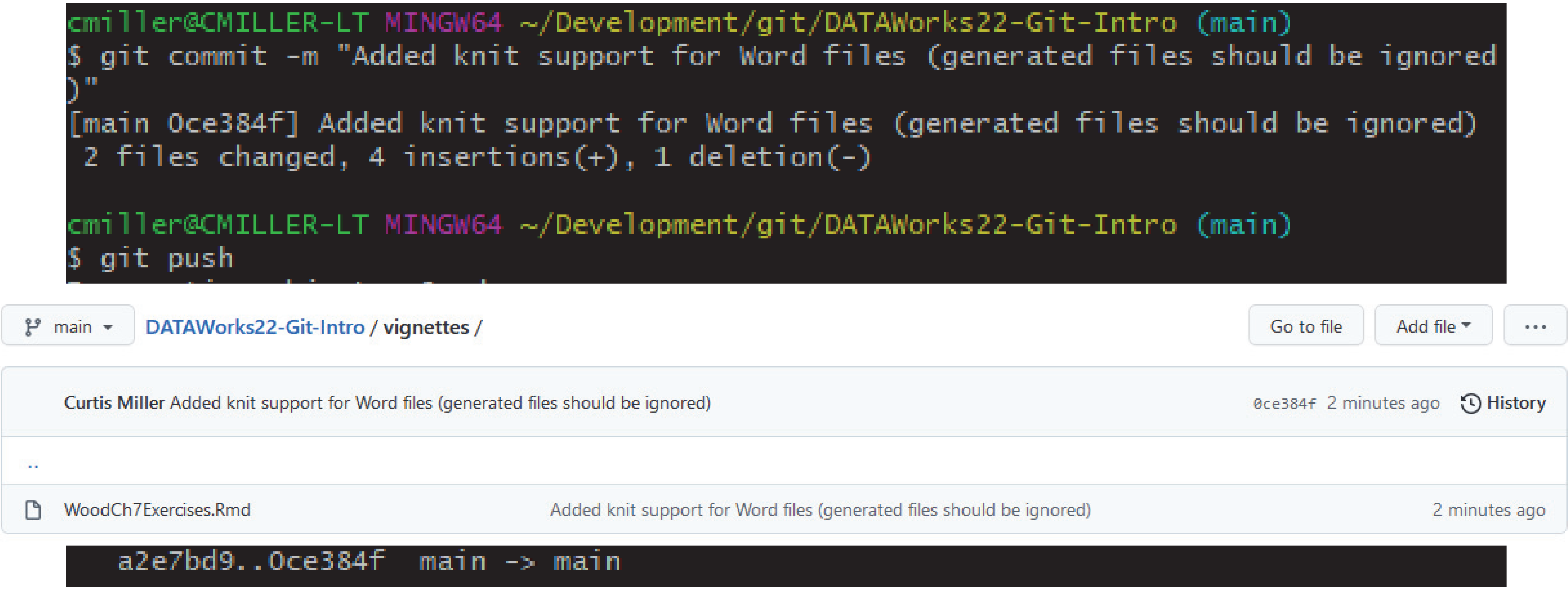
A summary of what we're about to change (such as adding files or changing them)

It did not add the Word file when we committed the changes…



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Added knit support for Word files (generated files should be ignored
)"
[main 0ce384f] Added knit support for Word files (generated files should be ignored)
 2 files changed, 4 insertions(+), 1 deletion(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 566 bytes | 283.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   a2e7bd9..0ce384f  main -> main
```

… Or pushed them to GitHub…



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git commit -m "Added knit support for Word files (generated files should be ignored
)"
[main 0ce384f] Added knit support for Word files (generated files should be ignored)
 2 files changed, 4 insertions(+), 1 deletion(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push
```

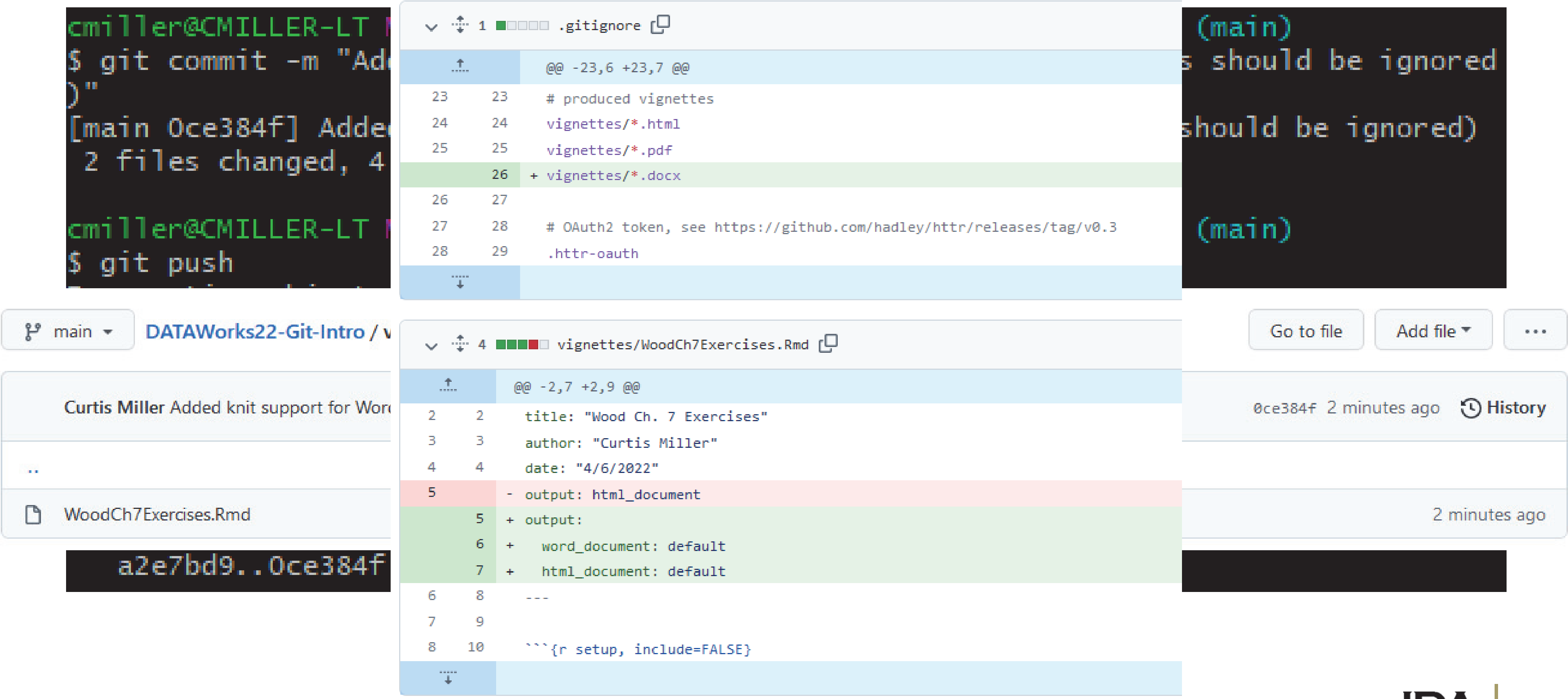
Curtis Miller Added knit support for Word files (generated files should be ignored) `0ce384f` 2 minutes ago

..

WoodCh7Exercises.Rmd Added knit support for Word files (generated files should be ignored) 2 minutes ago

```
   a2e7bd9..0ce384f  main -> main
```

… But it did track the other changes



```
cmiller@CMILLER-LT M                                        (main)
$ git commit -m "Ad                                         s should be ignored
)"
[main 0ce384f] Adde                                         should be ignored)
 2 files changed, 4

cmiller@CMILLER-LT M                                        (main)
$ git push
   a2e7bd9..0ce384f
```

main
DATAWorks22-Git-Intro / v
Go to file
Add file
History
..
WoodCh7Exercises.Rmd
2 minutes ago

Curtis Miller Added knit support for Word   0ce384f 2 minutes ago

```
1 .gitignore
@@ -23,6 +23,7 @@
  # produced vignettes
  vignettes/*.html
  vignettes/*.pdf
+ vignettes/*.docx

  # OAuth2 token, see https://github.com/hadley/httr/releases/tag/v0.3
  .httr-oauth
```

~~~
4 vignettes/WoodCh7Exercises.Rmd
@@ -2,7 +2,9 @@
  title: "Wood Ch. 7 Exercises"
  author: "Curtis Miller"
  date: "4/6/2022"
- output: html_document
+ output:
+   word_document: default
+   html_document: default
  ---

  ```{r setup, include=FALSE}
~~~

Basic change tracking commands

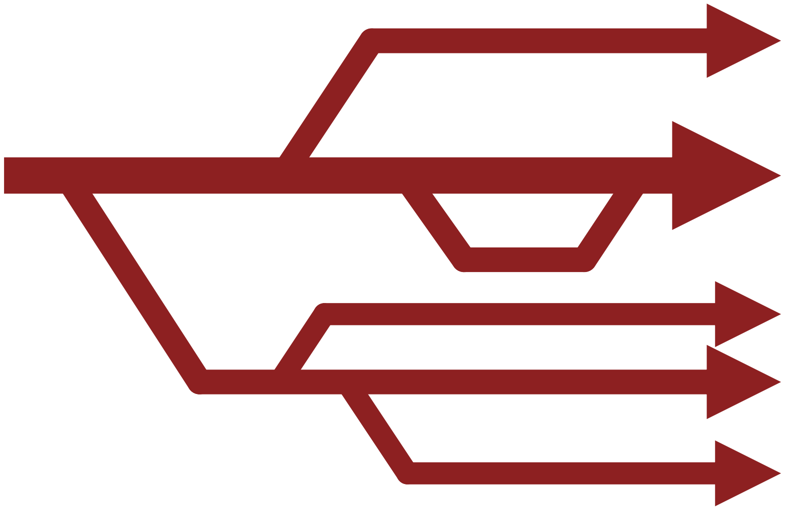
Command	Description	Examples
`git add`	Add changes to (or brand new) files to a staged commit	`git add myfile.txt` `git add --all`
`git commit`	Commit changes	`git commit –m “Message”`
`git pull`	For repos hosted remotely (such as GitHub), update repo with changes from host	`git pull`
`git push`	For repos hosted remotely (such as GitHub), send changes to host	`git push`
`git rm`	Remove a file from tracking	`git rm myfile.txt`
`git reset`	Git’s “undo” command	`git reset --hard    # CAUTION: Deletes changes`

`git diff`	Compare commits and branches	`git diff HEAD~ HEAD`
		`git diff HEAD~ HEAD myfile.txt`
		`git diff onebranch otherbranch`

`git status`	Show the status of a repo (such as what will be changed in a commit)	`git status`
`git log`	See commit history	`git log`
`git checkout`	Retrieve data from commits	`git checkout identifier`

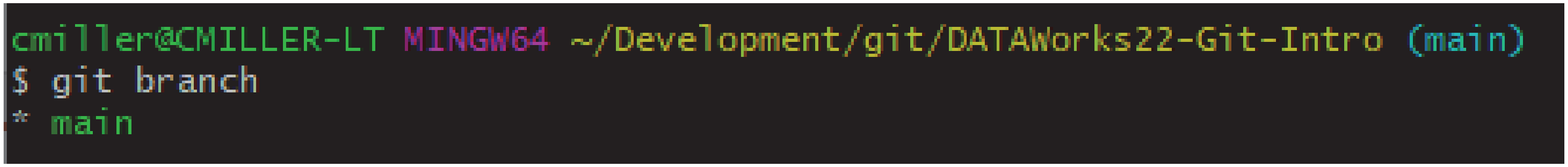
Branches

Branches allow multiple versions of files to exist and be tracked simultaneously



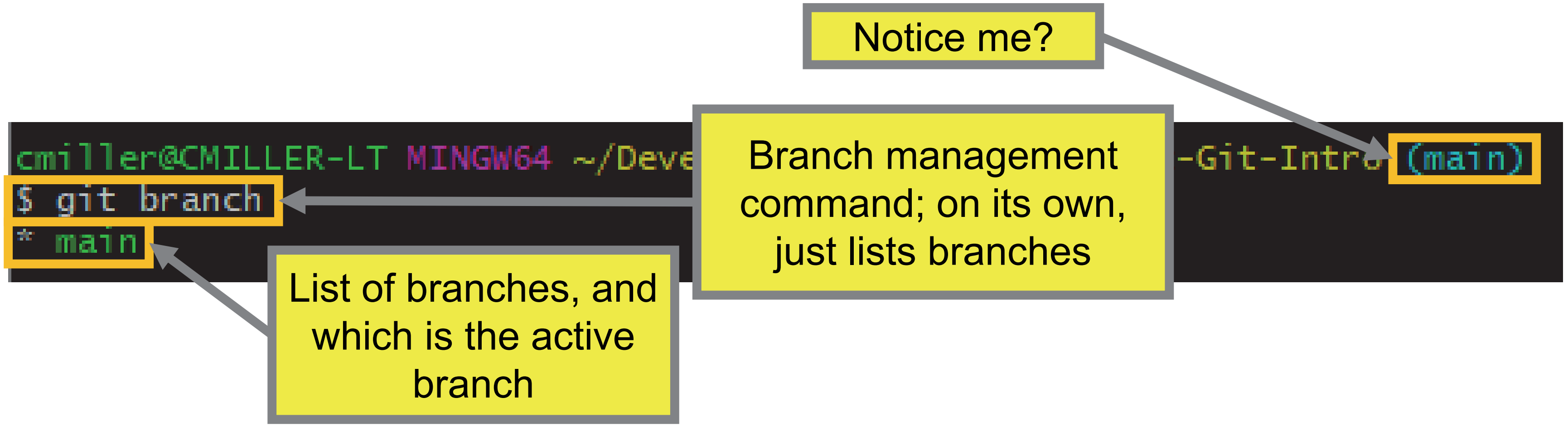
- Commits represent moving forward and backward in time while branches allow for differences between files to exist simultaneously
- Git can track the difference between branches and propagate a change that affects both branches while maintaining what makes them distinct
- Changes made in one branch can be merged into another branch
- Branching provides a means for collaboration; each collaborator creates their own private branch to do their work
- Branching allows collaborators to simultaneously maintain different versions of a product

What branches do we start out with?



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git branch
* main
```

The branch that best represents the project’s most stable state is the main branch



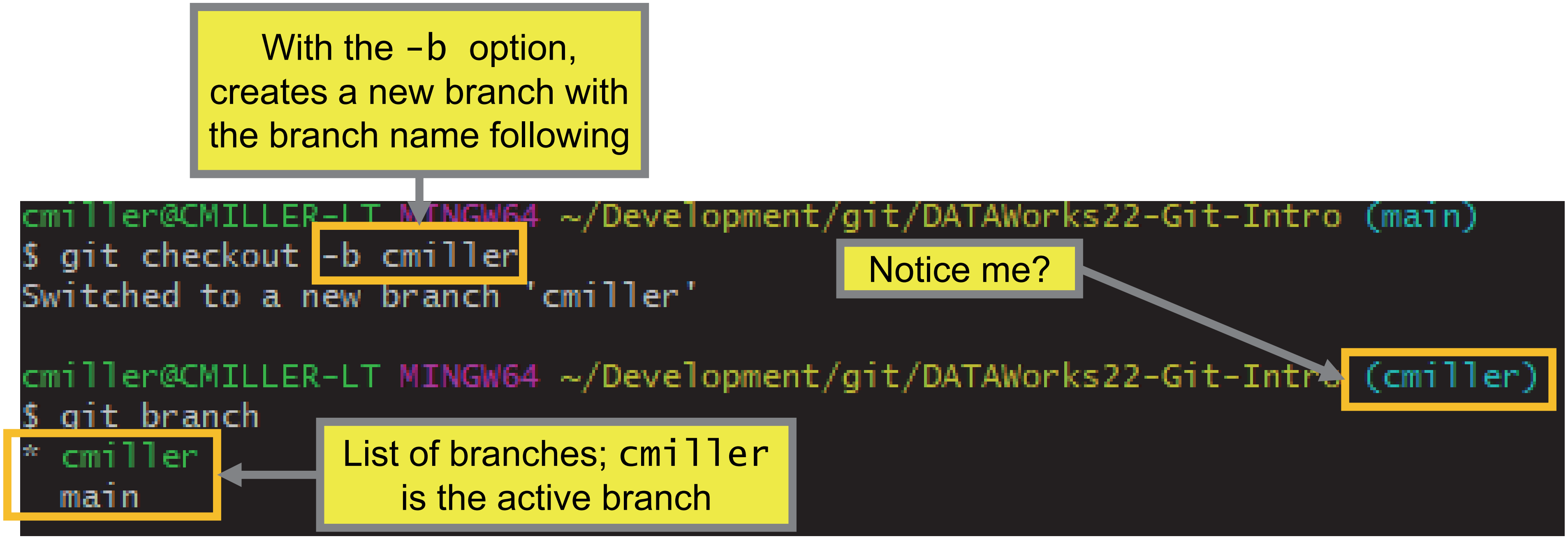
```
cmiller@CMILLER-LT MINGW64 ~/Deve                        -Git-Intro (main)
$ git branch
* main
```

Notice me?

Branch management command; on its own, just lists branches

List of branches, and which is the active branch

Let’s create a personal branch



With the `-b` option, creates a new branch with the branch name following

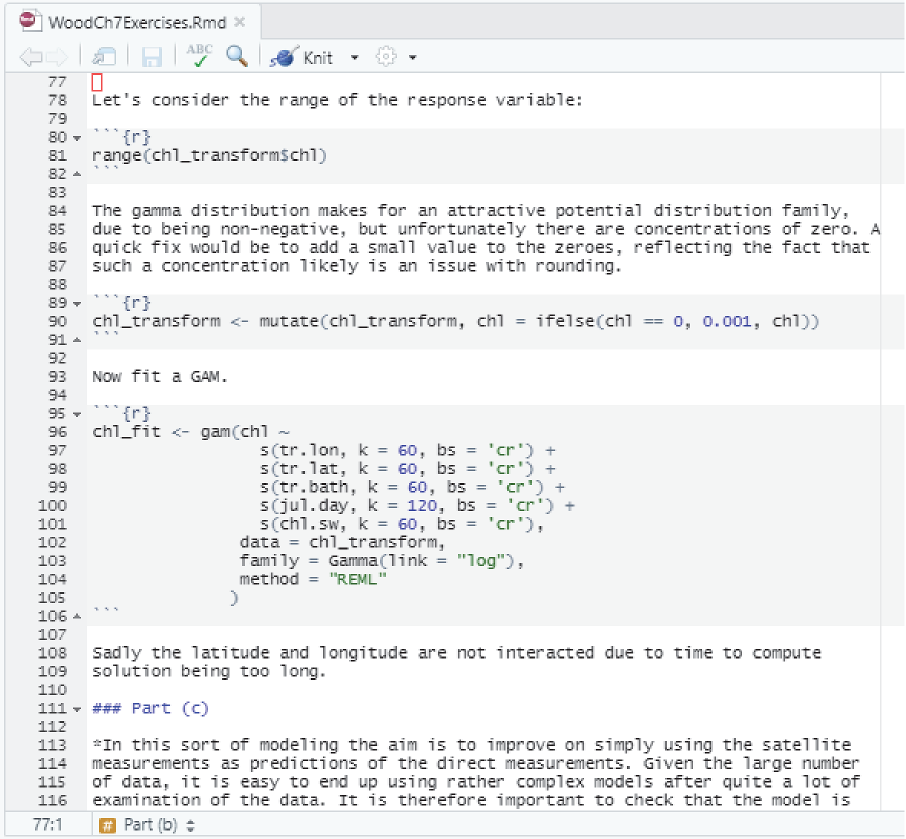
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git checkout -b cmiller
Switched to a new branch 'cmiller'

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git branch
* cmiller
  main
```

Notice me?

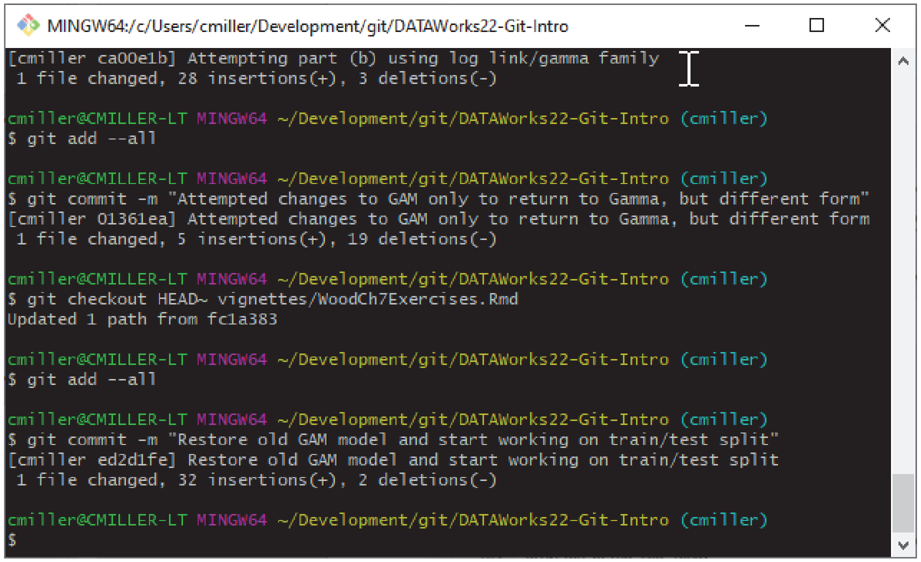
List of branches; `cmiller` is the active branch

When using Git with a team, avoid editing the main branch directly. Make edits on a personal branch freely. Changes to the main branch should be done via merging changes from other branches most of the time (we will discuss branch merging later). Only when the team is ready to “accept” edits should the main branch be changed.

Now work primarily with the personal branch

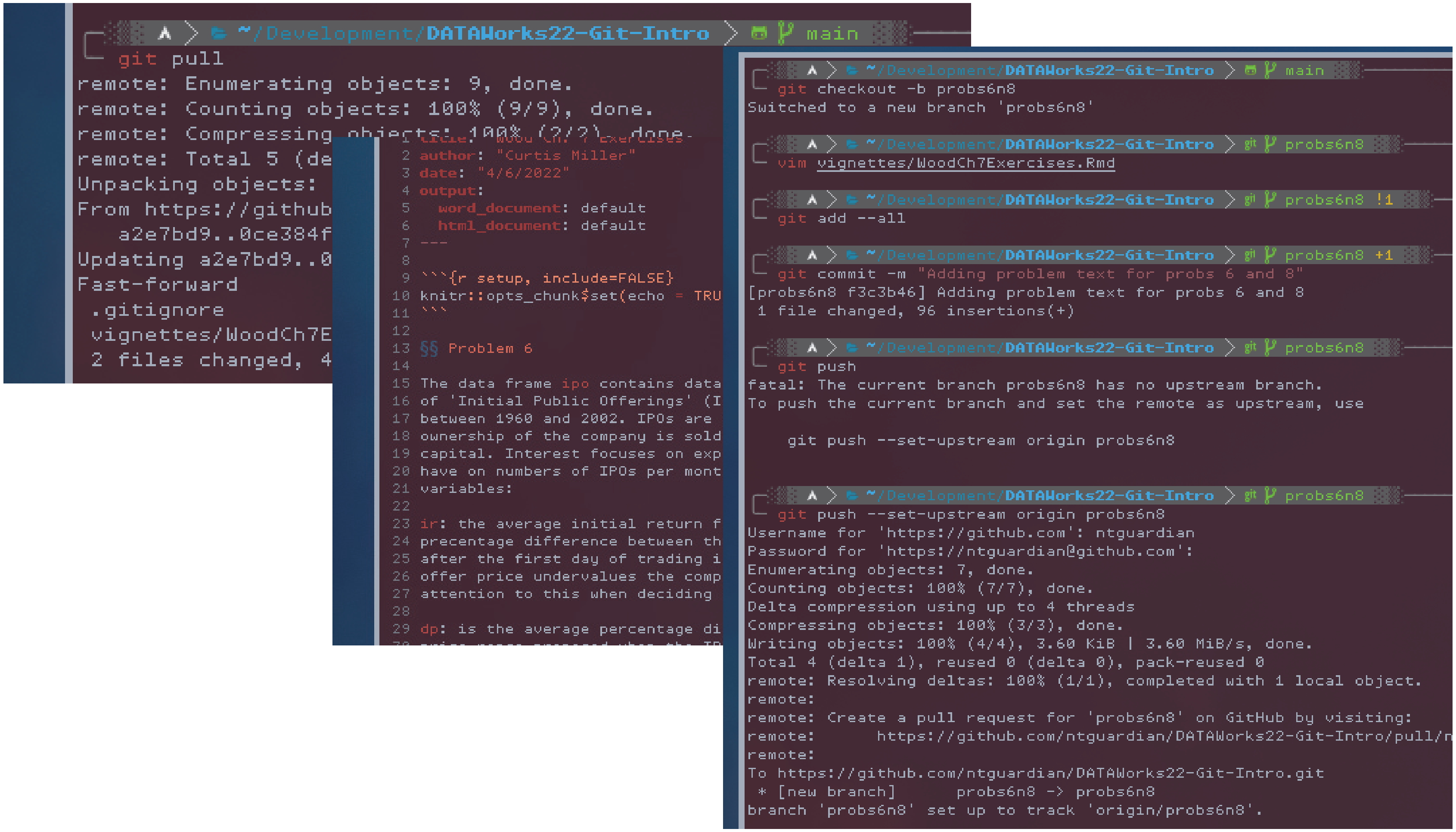


~~~
Let's consider the range of the response variable:

```{r}
range(chl_transform$chl)
```

The gamma distribution makes for an attractive potential distribution family,
due to being non-negative, but unfortunately there are concentrations of zero. A
quick fix would be to add a small value to the zeroes, reflecting the fact that
such a concentration likely is an issue with rounding.

```{r}
chl_transform <- mutate(chl_transform, chl = ifelse(chl == 0, 0.001, chl))
```

Now fit a GAM.

```{r}
chl_fit <- gam(chl ~
                 s(tr.lon, k = 60, bs = 'cr') +
                 s(tr.lat, k = 60, bs = 'cr') +
                 s(tr.bath, k = 60, bs = 'cr') +
                 s(jul.day, k = 120, bs = 'cr') +
                 s(chl.sw, k = 60, bs = 'cr'),
               data = chl_transform,
               family = Gamma(link = "log"),
               method = "REML"
              )
```

Sadly the latitude and longitude are not interacted due to time to compute
solution being too long.

### Part (c)

*In this sort of modeling the aim is to improve on simply using the satellite
measurements as predictions of the direct measurements. Given the large number
of data, it is easy to end up using rather complex models after quite a lot of
examination of the data. It is therefore important to check that the model is
~~~


MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro

```
[cmiller ca00e1b] Attempting part (b) using log link/gamma family
 1 file changed, 28 insertions(+), 3 deletions(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git commit -m "Attempted changes to GAM only to return to Gamma, but different form"
[cmiller 01361ea] Attempted changes to GAM only to return to Gamma, but different form
 1 file changed, 5 insertions(+), 19 deletions(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git checkout HEAD~ vignettes/WoodCh7Exercises.Rmd
Updated 1 path from fc1a383

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git commit -m "Restore old GAM model and start working on train/test split"
[cmiller ed2d1fe] Restore old GAM model and start working on train/test split
 1 file changed, 32 insertions(+), 2 deletions(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$
```

As we were doing work, a “collaborator” did some work too

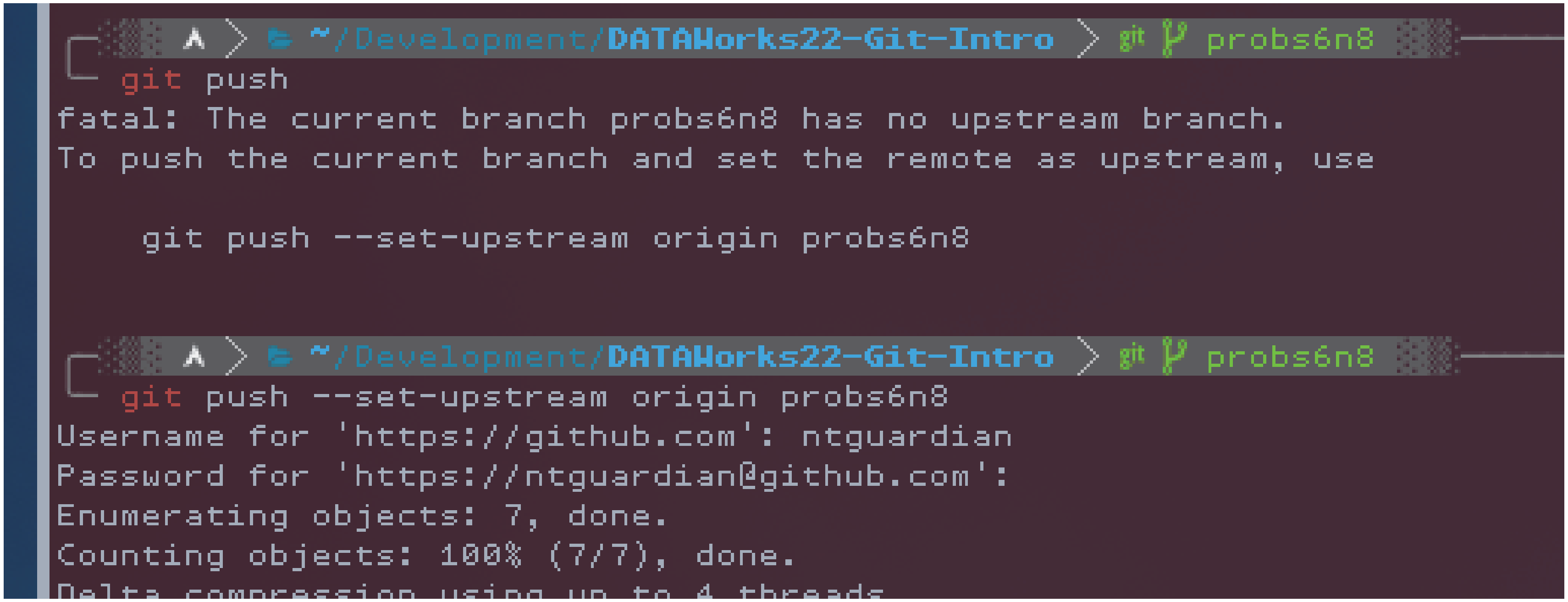


```
~/Development/DATAWorks22-Git-Intro  main
git pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 5 (de
Unpacking objects:
From https://github
   a2e7bd9..0ce384f
Updating a2e7bd9..0
Fast-forward
 .gitignore
 vignettes/WoodCh7E
 2 files changed, 4
```

~~~
title: "Wood Ch. 7 Exercises"
author: "Curtis Miller"
date: "4/6/2022"
output:
  word_document: default
  html_document: default
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRU
```

§§ Problem 6

The data frame ipo contains data
of 'Initial Public Offerings' (I
between 1960 and 2002. IPOs are
ownership of the company is sold
capital. Interest focuses on exp
have on numbers of IPOs per mont
variables:

ir: the average initial return f
precentage difference between th
after the first day of trading i
offer price undervalues the comp
attention to this when deciding

dp: is the average percentage di
~~~

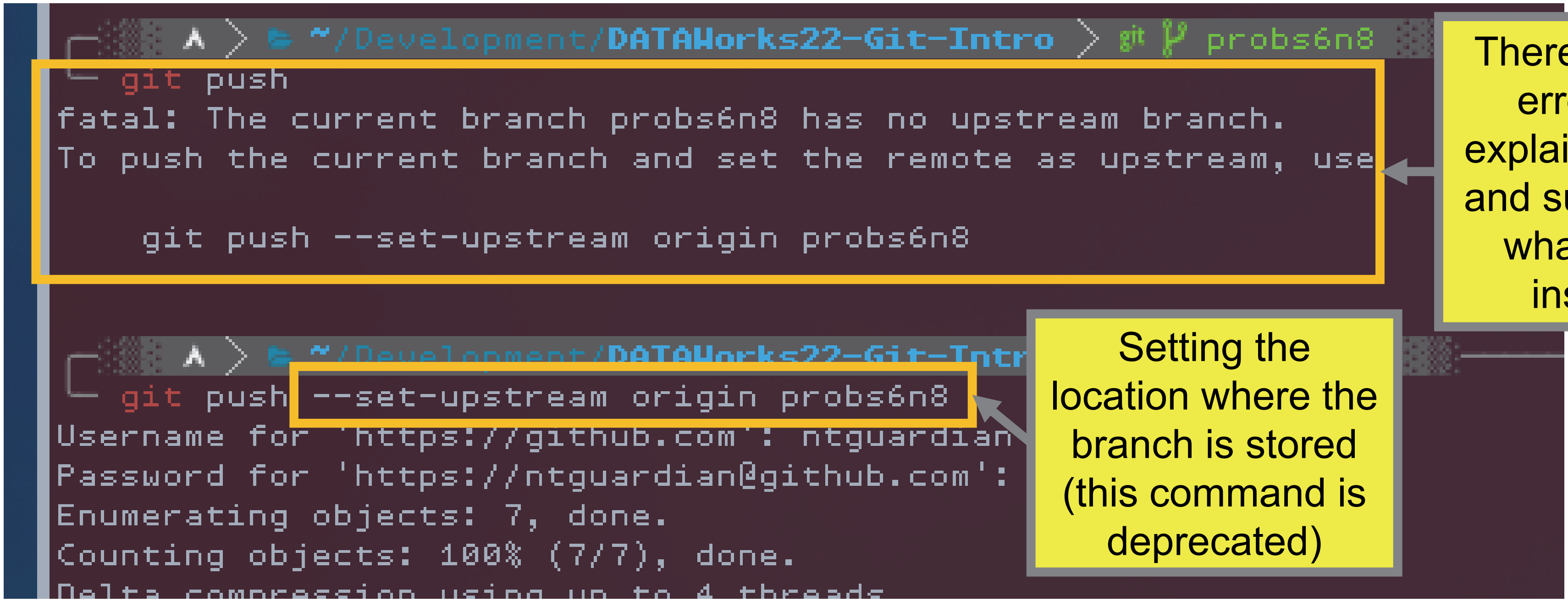
```
~/Development/DATAWorks22-Git-Intro  main
git checkout -b probs6n8
Switched to a new branch 'probs6n8'

~/Development/DATAWorks22-Git-Intro  git  probs6n8
vim vignettes/WoodCh7Exercises.Rmd

~/Development/DATAWorks22-Git-Intro  git  probs6n8 !1
git add --all

~/Development/DATAWorks22-Git-Intro  git  probs6n8 +1
git commit -m "Adding problem text for probs 6 and 8"
[probs6n8 f3c3b46] Adding problem text for probs 6 and 8
 1 file changed, 96 insertions(+)

~/Development/DATAWorks22-Git-Intro  git  probs6n8
git push
fatal: The current branch probs6n8 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin probs6n8


~/Development/DATAWorks22-Git-Intro  git  probs6n8
git push --set-upstream origin probs6n8
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 3.60 KiB | 3.60 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'probs6n8' on GitHub by visiting:
remote:      https://github.com/ntguardian/DATAWorks22-Git-Intro/pull/n
remote:
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
 * [new branch]      probs6n8 -> probs6n8
branch 'probs6n8' set up to track 'origin/probs6n8'.
```

Note how to add a branch to the external repo



```
~/Development/DATAWorks22-Git-Intro  git  probs6n8
 git push
fatal: The current branch probs6n8 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin probs6n8


~/Development/DATAWorks22-Git-Intro  git  probs6n8
 git push --set-upstream origin probs6n8
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
```

Note how to add a branch to the external repo

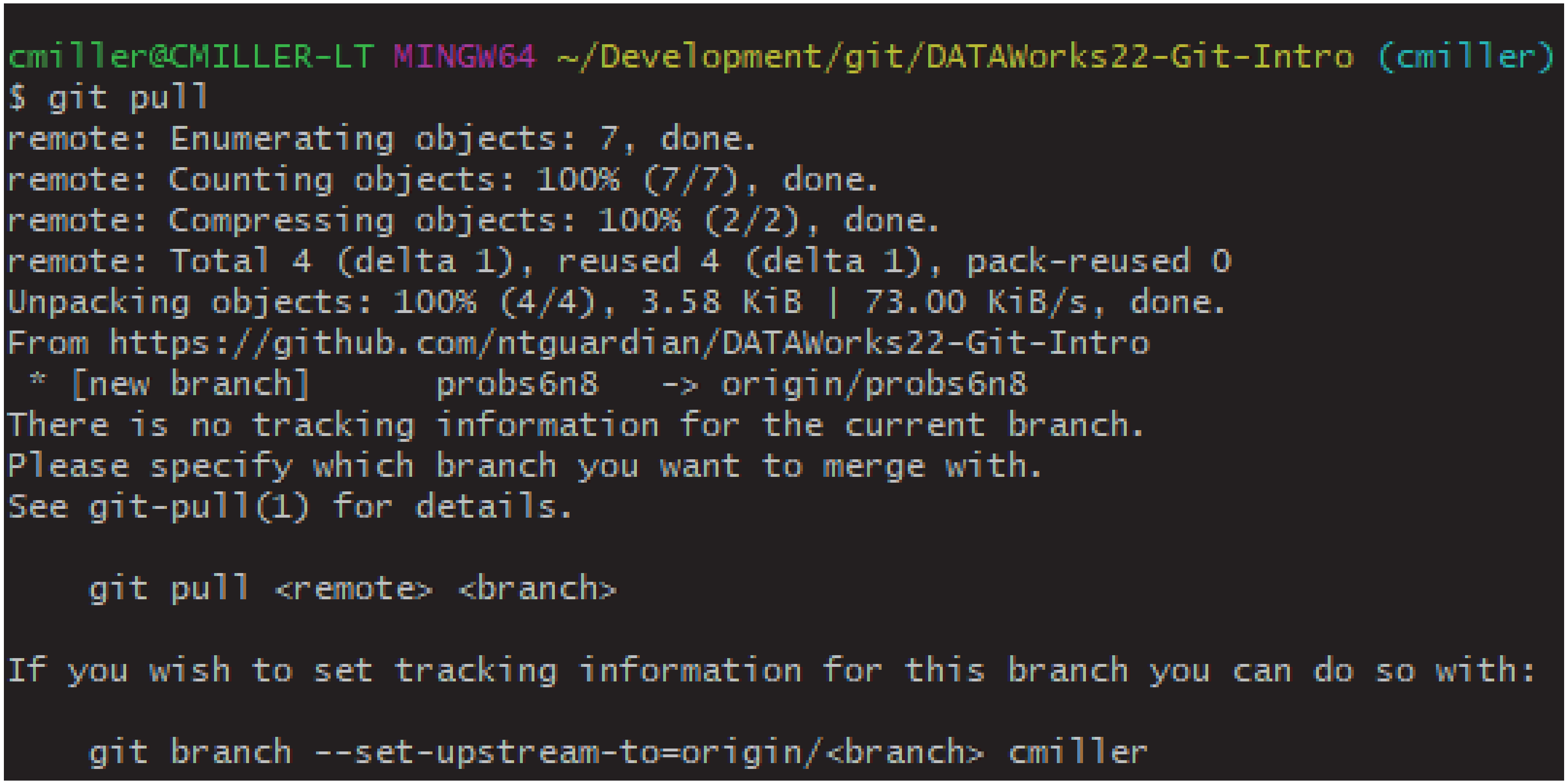


```
~/Development/DATAWorks22-Git-Intro  git  probs6n8
 git push
fatal: The current branch probs6n8 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin probs6n8

~/Development/DATAWorks22-Git-Intr
 git push --set-upstream origin probs6n8
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
```

There was an error; Git explained why, and suggested what to do instead

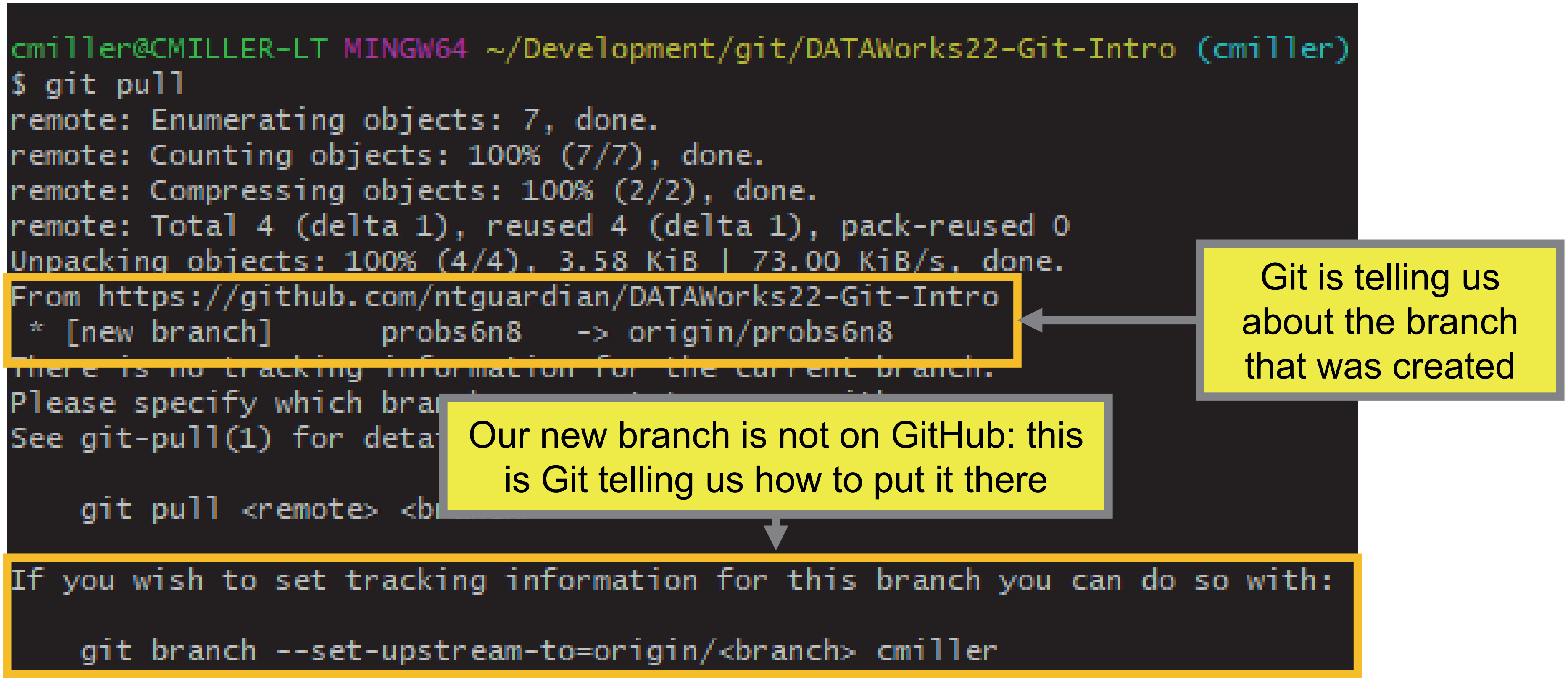
Setting the location where the branch is stored (this command is deprecated)

We should sync our local copy of the repo now



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), 3.58 KiB | 73.00 KiB/s, done.
From https://github.com/ntguardian/DATAWorks22-Git-Intro
 * [new branch]      probs6n8   -> origin/probs6n8
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> cmiller
```

Read the resulting messages

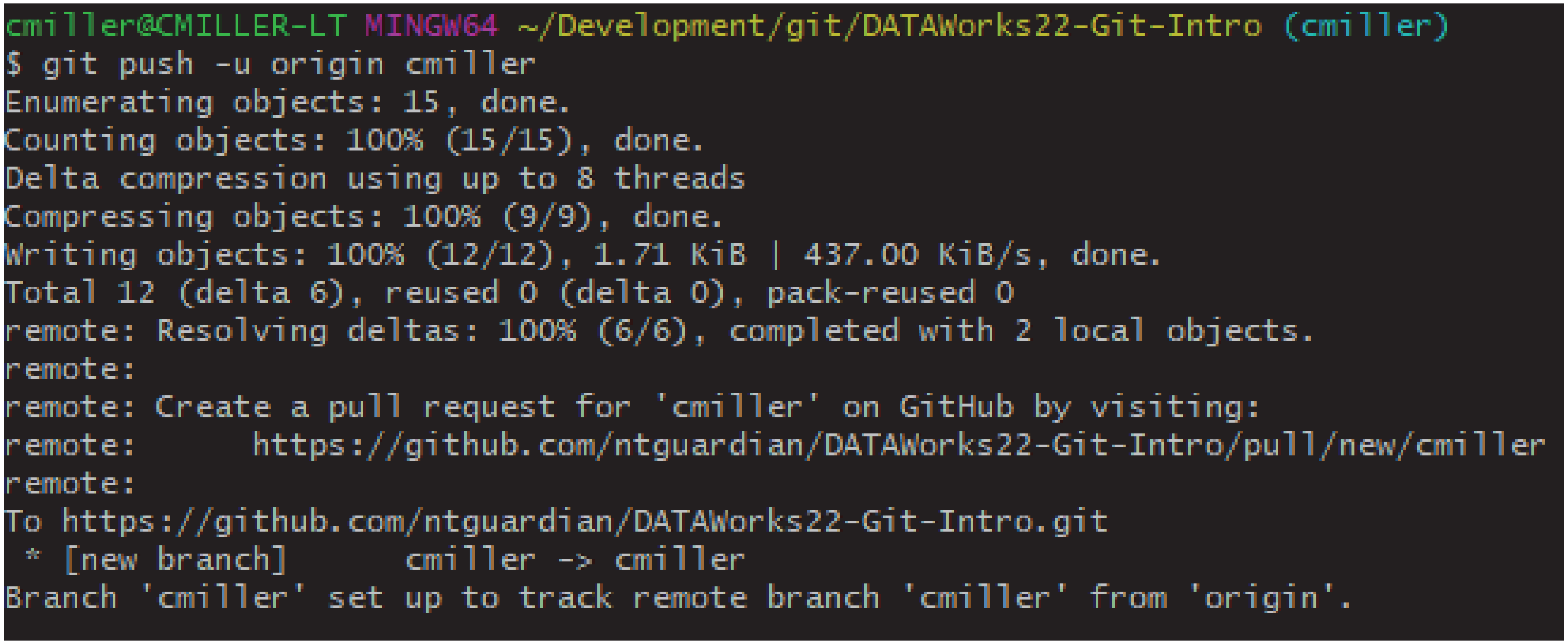


```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (4/4), 3.58 KiB | 73.00 KiB/s, done.
From https://github.com/ntguardian/DATAWorks22-Git-Intro
 * [new branch]      probs6n8    -> origin/probs6n8
There is no tracking information for the current branch.
Please specify which bra
See git-pull(1) for deta

    git pull <remote> <b

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> cmiller
```

Git is telling us about the branch that was created

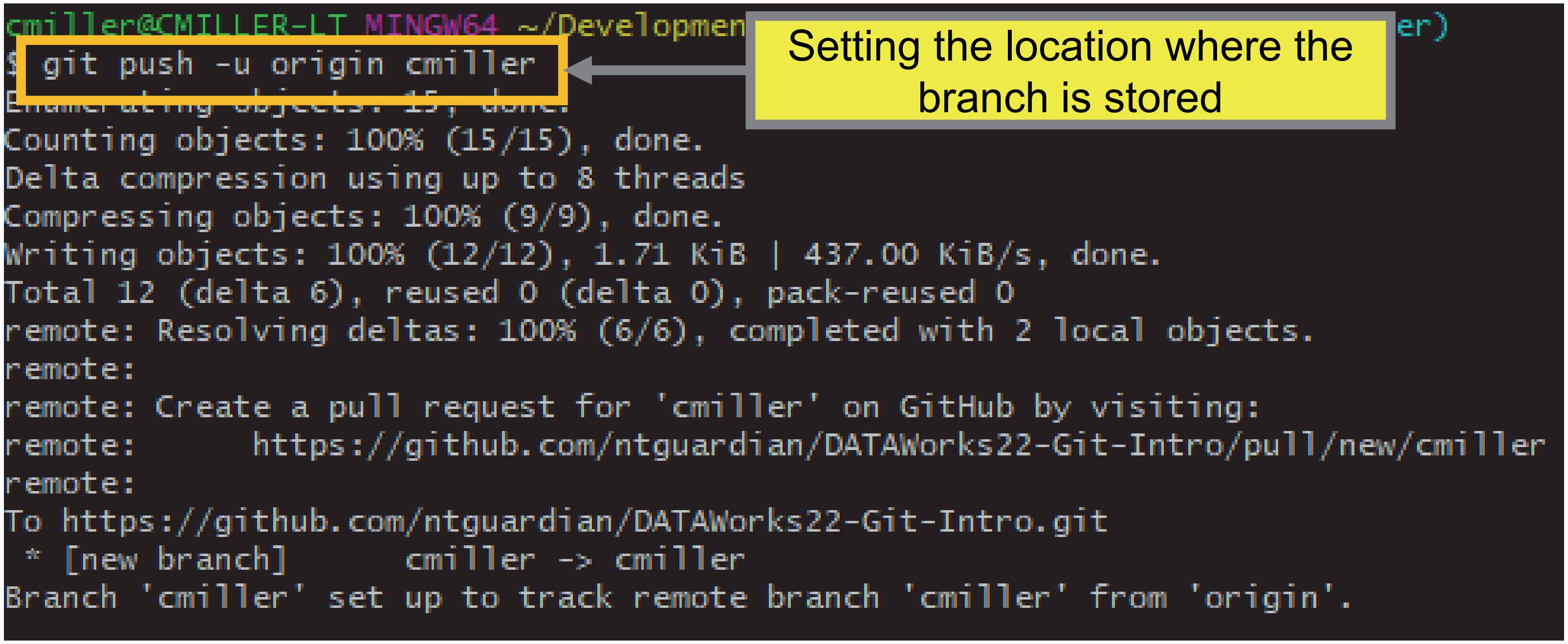
Our new branch is not on GitHub: this is Git telling us how to put it there

We can use the following command to track our branch remotely



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git push -u origin cmiller
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.71 KiB | 437.00 KiB/s, done.
Total 12 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
remote:
remote: Create a pull request for 'cmiller' on GitHub by visiting:
remote:      https://github.com/ntguardian/DATAWorks22-Git-Intro/pull/new/cmiller
remote:
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
 * [new branch]      cmiller -> cmiller
Branch 'cmiller' set up to track remote branch 'cmiller' from 'origin'.
```

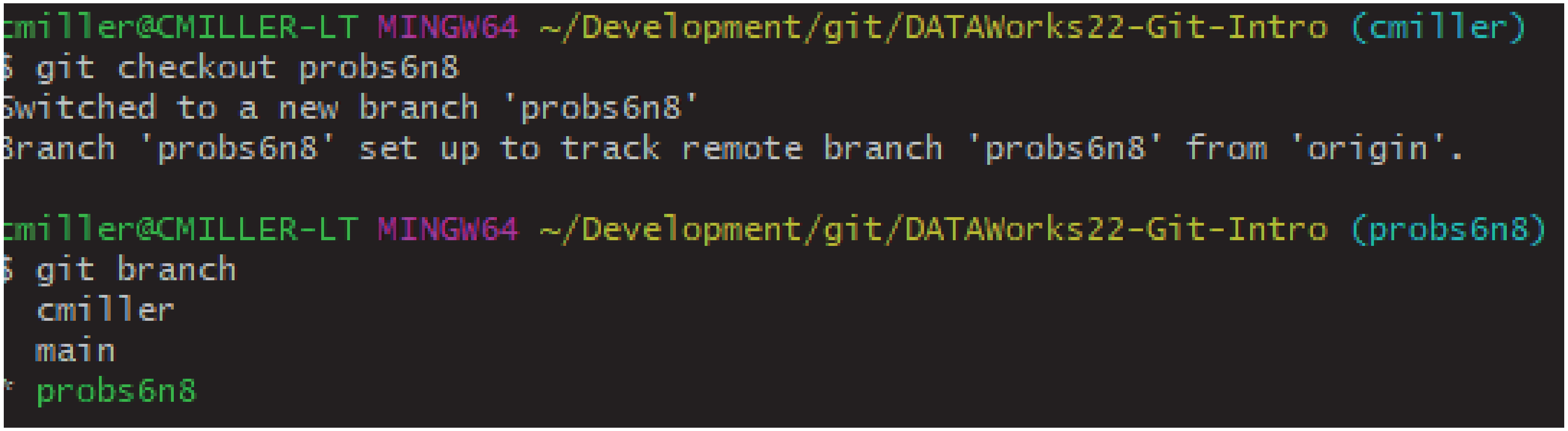
We can use the following command to track our branch remotely



```
cmiller@CMILLER-LT MINGW64 ~/Developmen                                    er)
$ git push -u origin cmiller
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.71 KiB | 437.00 KiB/s, done.
Total 12 (delta 6), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (6/6), completed with 2 local objects.
remote:
remote: Create a pull request for 'cmiller' on GitHub by visiting:
remote:      https://github.com/ntguardian/DATAWorks22-Git-Intro/pull/new/cmiller
remote:
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
 * [new branch]      cmiller -> cmiller
Branch 'cmiller' set up to track remote branch 'cmiller' from 'origin'.
```

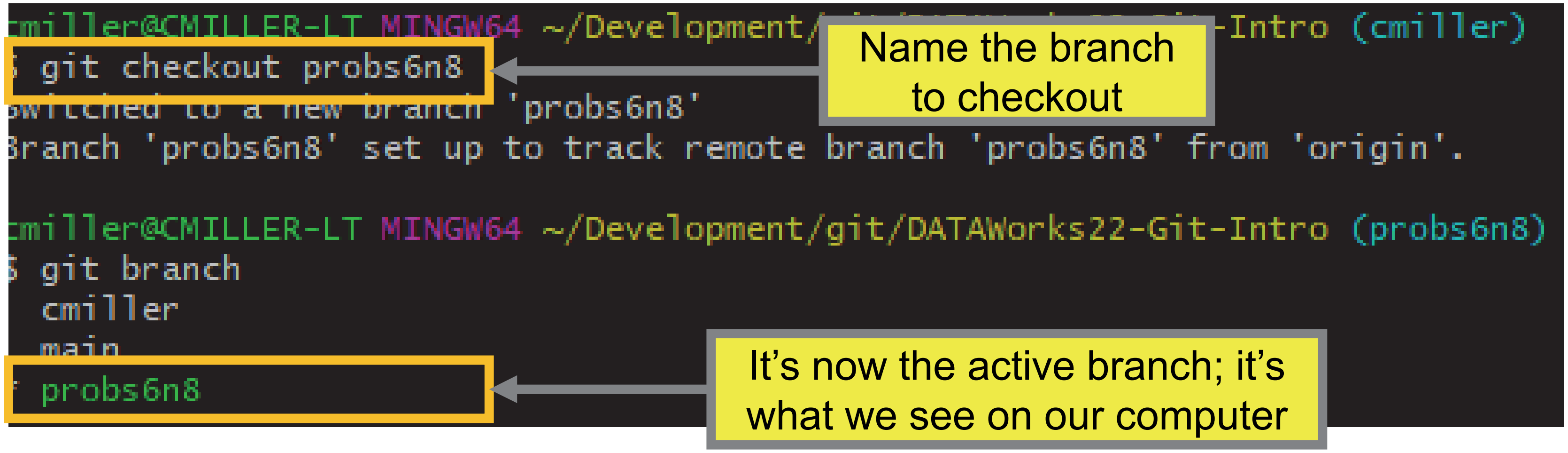
Setting the location where the branch is stored

To see the new branch, use `git checkout`



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git checkout probs6n8
Switched to a new branch 'probs6n8'
Branch 'probs6n8' set up to track remote branch 'probs6n8' from 'origin'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git branch
  cmiller
  main
* probs6n8
```

To see the new branch, use `git checkout`

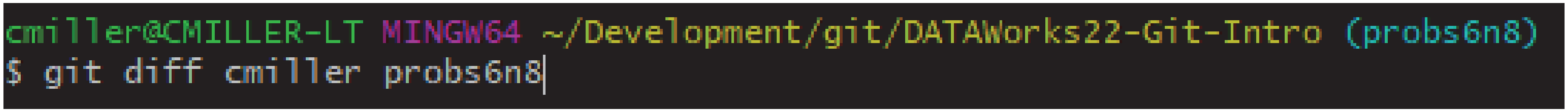


```
cmiller@CMILLER-LT MINGW64 ~/Development/                   -Intro (cmiller)
$ git checkout probs6n8
Switched to a new branch 'probs6n8'
Branch 'probs6n8' set up to track remote branch 'probs6n8' from 'origin'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git branch
  cmiller
  main
* probs6n8
```

Name the branch to checkout

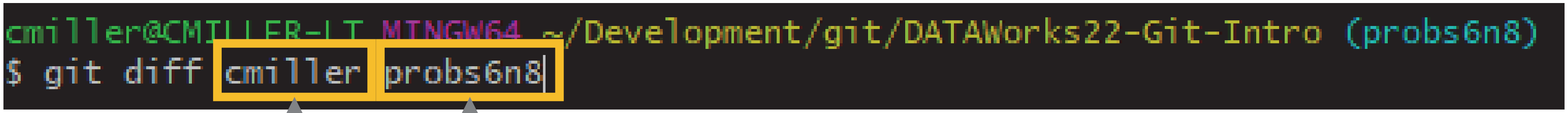
It's now the active branch; it's what we see on our computer

To compare branches, use `git diff`



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git diff cmiller probs6n8
```

To compare branches, use `git diff`



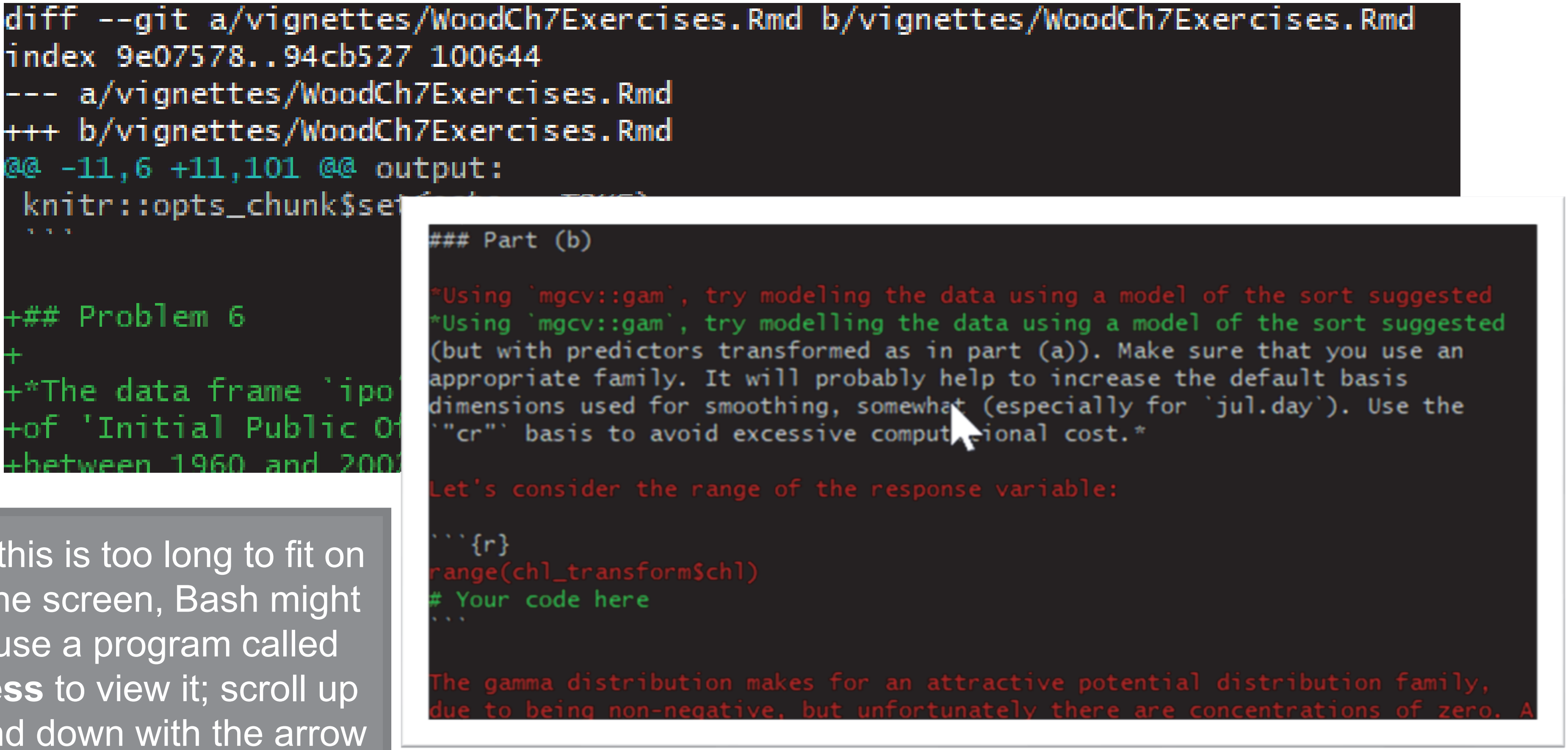
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git diff cmiller probs6n8
```

Other branch to be compared

One branch to be compared

Stuff in the branch on the right but not the left will be shown in green (an addition), and stuff in the branch on the left and not the right shown in red (a deletion)

We can now compare what’s in the branches



```
diff --git a/vignettes/WoodCh7Exercises.Rmd b/vignettes/WoodCh7Exercises.Rmd
index 9e07578..94cb527 100644
--- a/vignettes/WoodCh7Exercises.Rmd
+++ b/vignettes/WoodCh7Exercises.Rmd
@@ -11,6 +11,101 @@ output:
 knitr::opts_chunk$set
 ```

+## Problem 6
+
+*The data frame `ipo
+of 'Initial Public Of
+between 1960 and 200
```

```
Part (b)

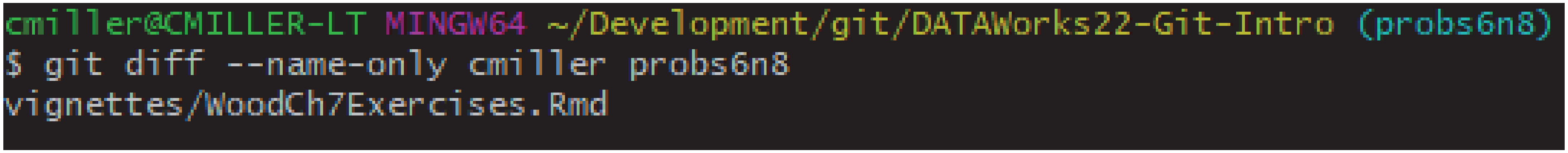
*Using `mgcv::gam`, try modeling the data using a model of the sort suggested
*Using `mgcv::gam`, try modelling the data using a model of the sort suggested
(but with predictors transformed as in part (a)). Make sure that you use an
appropriate family. It will probably help to increase the default basis
dimensions used for smoothing, somewhat (especially for `jul.day`). Use the
`"cr"` basis to avoid excessive computational cost.*

Let's consider the range of the response variable:

```{r}
range(chl_transform$chl)
# Your code here
```

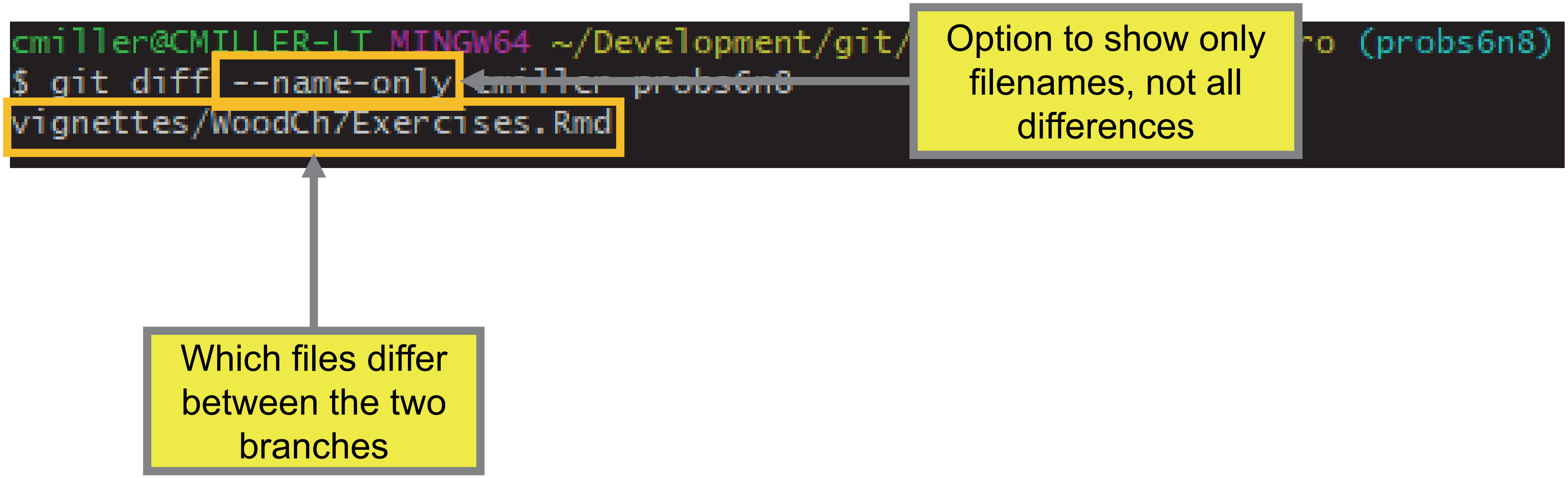
The gamma distribution makes for an attractive potential distribution family,
due to being non-negative, but unfortunately there are concentrations of zero. A
```

If this is too long to fit on one screen, Bash might use a program called **less** to view it; scroll up and down with the arrow keys, then press q to quit
```

“I only wanted to see which files are different, not the differences themselves.”



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git diff --name-only cmiller probs6n8
vignettes/WoodCh7Exercises.Rmd
```

“I only wanted to see which files are different, not the differences themselves.”

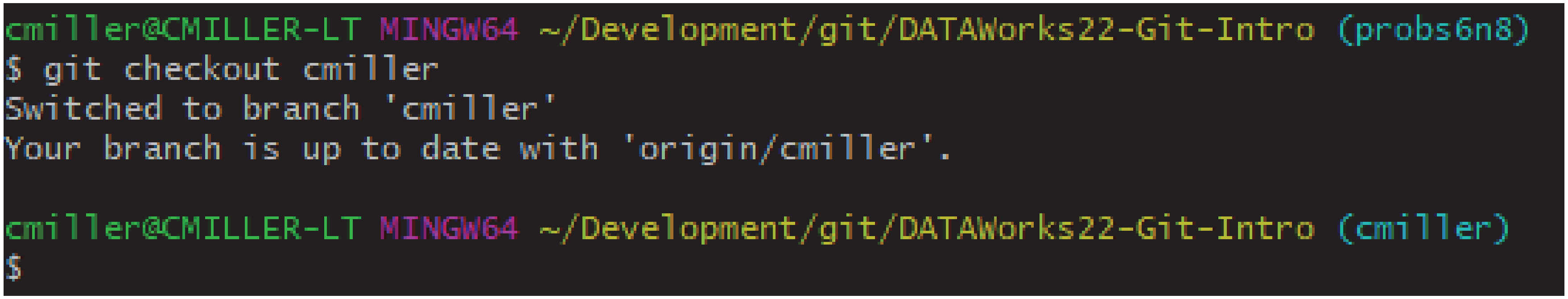


```
cmiller@CMILLER-LT MINGW64 ~/Development/git/                    ro (probs6n8)
$ git diff --name-only cmiller probs6n8
vignettes/WoodCh7Exercises.Rmd
```

Option to show only filenames, not all differences

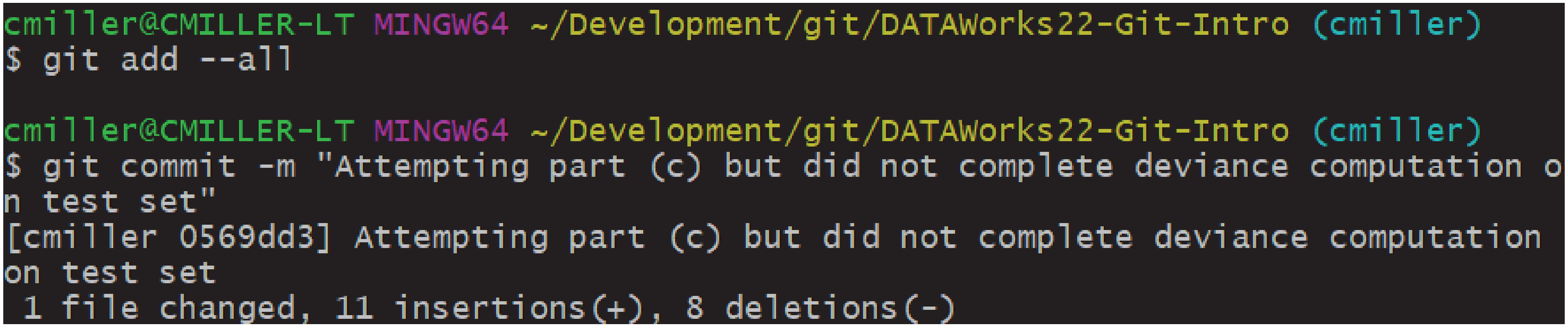
Which files differ between the two branches

Now go back to our personal branch



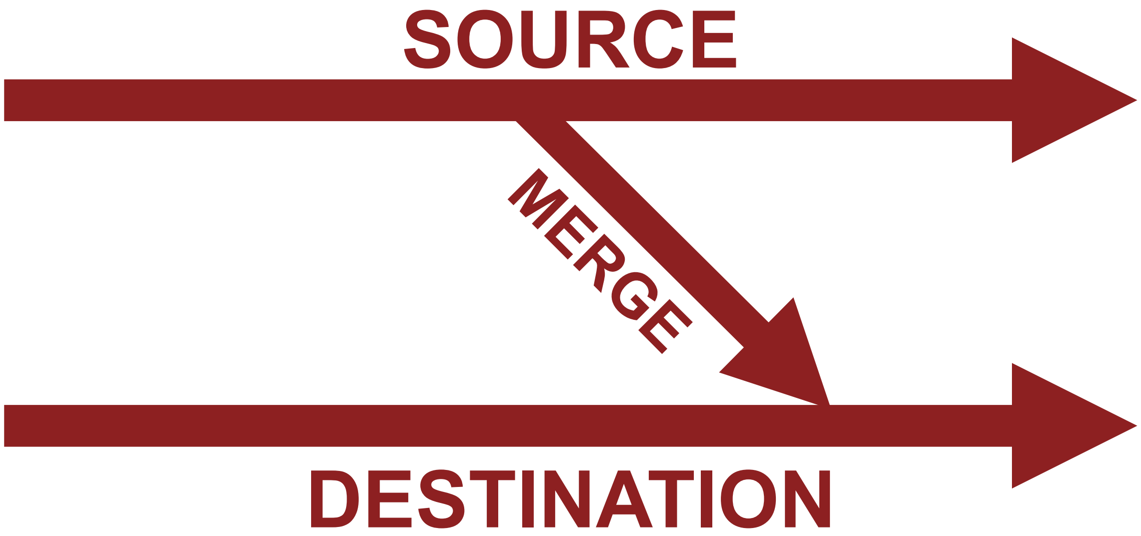
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git checkout cmiller
Switched to branch 'cmiller'
Your branch is up to date with 'origin/cmiller'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$
```

A few more edits…



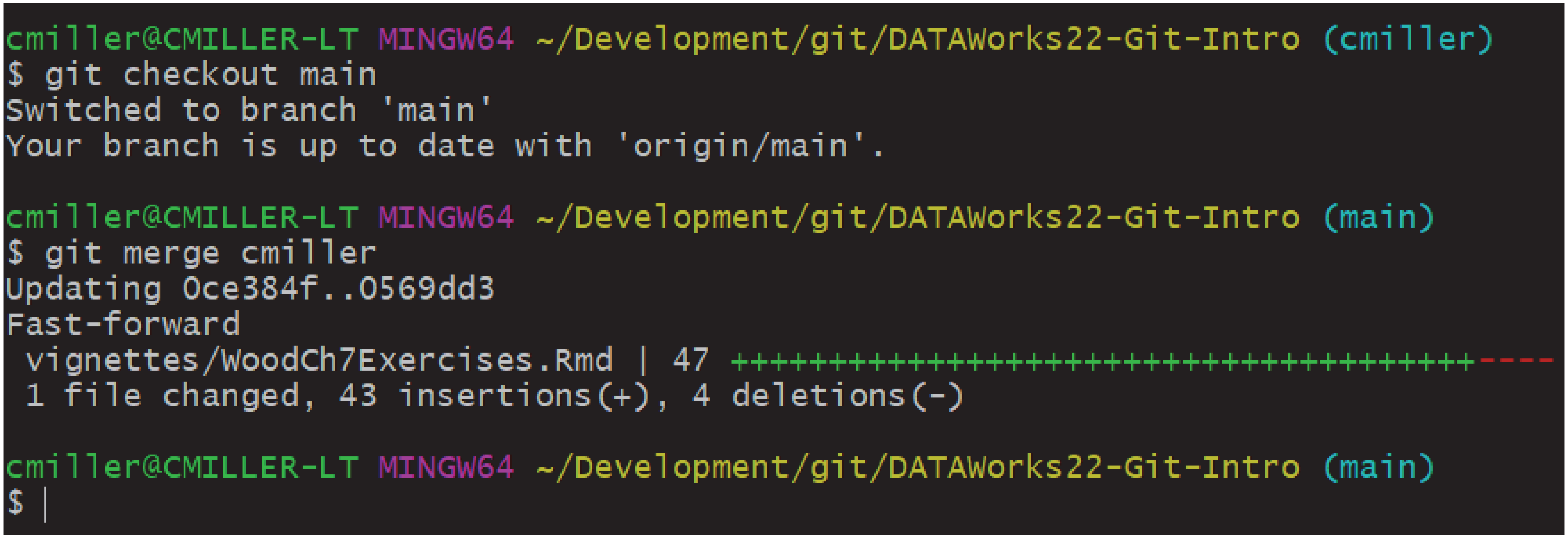
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git commit -m "Attempting part (c) but did not complete deviance computation o
n test set"
[cmiller 0569dd3] Attempting part (c) but did not complete deviance computation
on test set
 1 file changed, 11 insertions(+), 8 deletions(-)
```

Use git merge to propagate changes made in one branch to another branch



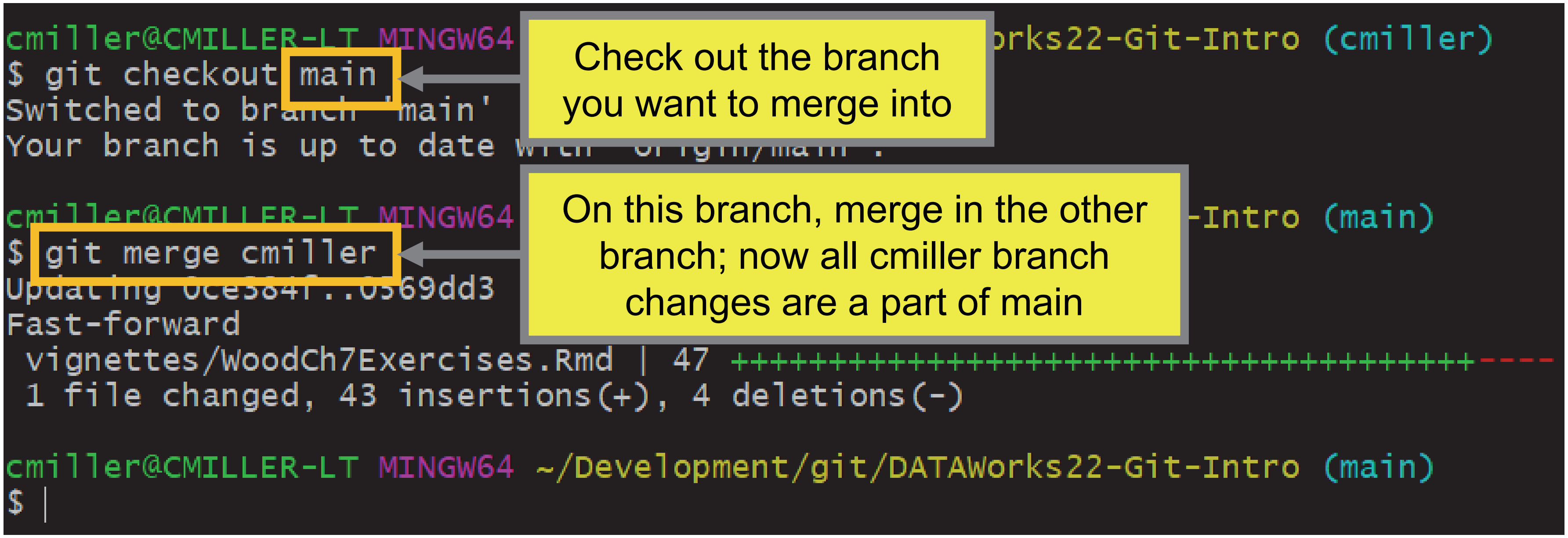
- Eventually one may want to merge the changes made in one branch to another branch
- `git merge` will determine how the two branches should be merged together
- When using Git merge, *check out the destination branch for the merge first*, then call `git merge sourcebranch`
- The changes from the source branch will then be propagated to the destination branch
- Git may ask for a message to be written to explain the merge

A merge may be simple...



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git merge cmiller
Updating 0ce384f..0569dd3
Fast-forward
 vignettes/WoodCh7Exercises.Rmd | 47 ++++++++++++++++++++++++++++++++++++++----
 1 file changed, 43 insertions(+), 4 deletions(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$
```

In this case, it was not too hard

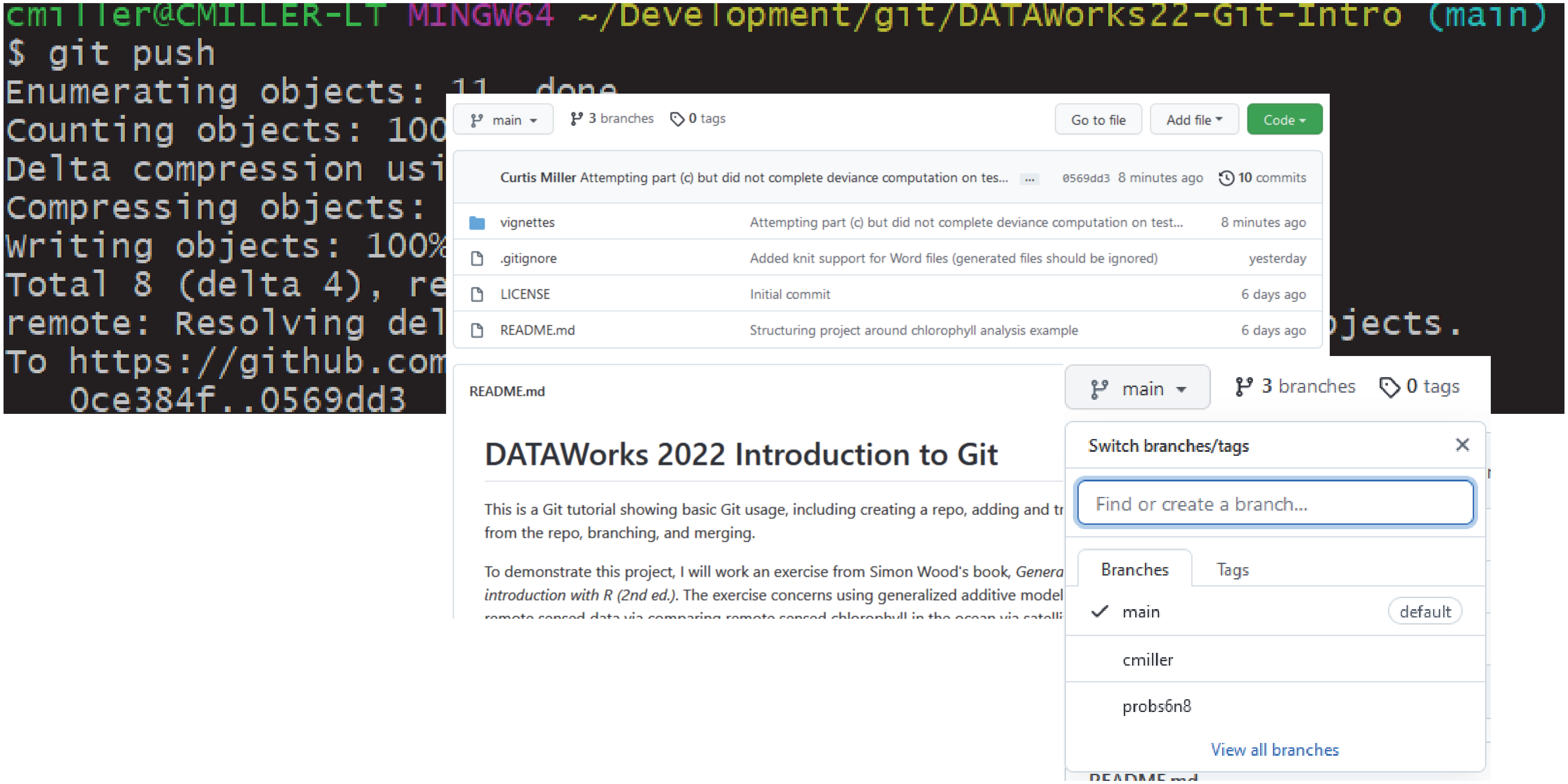


```
cmiller@CMILLER-LT MINGW64                         orks22-Git-Intro (cmiller)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

cmiller@CMILLER-LT MINGW64                                  -Intro (main)
$ git merge cmiller
Updating 0ce384f..0569dd3
Fast-forward
 vignettes/WoodCh7Exercises.Rmd | 47 ++++++++++++++++++++++++++++++++++++++----
 1 file changed, 43 insertions(+), 4 deletions(-)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ |
```

Check out the branch you want to merge into

On this branch, merge in the other branch; now all cmiller branch changes are a part of main

A merger is treated like a new commit


main
3 branches
0 tags
Go to file
Add file
Code
README.md
main
3 branches
0 tags
Switch branches/tags
Find or create a branch...
Branches
Tags
View all branches

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push
Enumerating objects: 11, done.
Counting objects: 100
Delta compression usi
Compressing objects:
Writing objects: 100%
Total 8 (delta 4), re
remote: Resolving del                                   ojects.
To https://github.com
   0ce384f..0569dd3
```

Curtis Miller Attempting part (c) but did not complete deviance computation on tes... 0569dd3 8 minutes ago 10 commits

vignettes	Attempting part (c) but did not complete deviance computation on test...	8 minutes ago
.gitignore	Added knit support for Word files (generated files should be ignored)	yesterday
LICENSE	Initial commit	6 days ago
README.md	Structuring project around chlorophyll analysis example	6 days ago

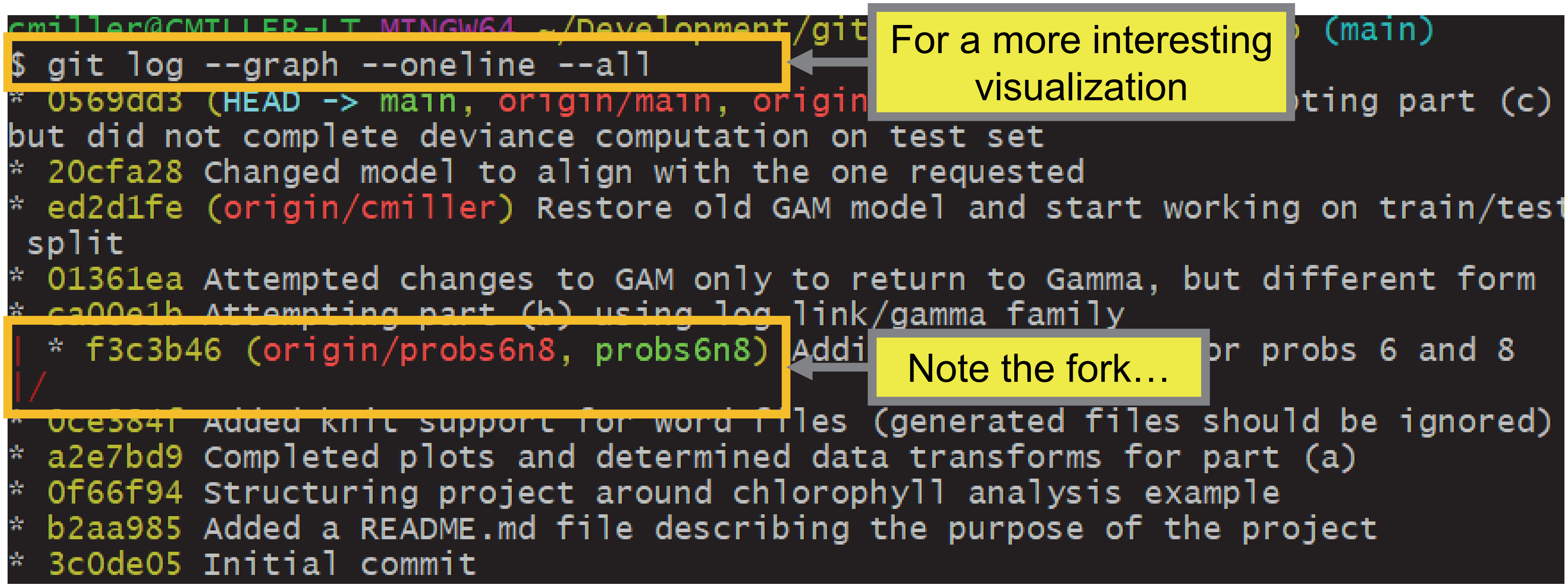
DATAWorks 2022 Introduction to Git

This is a Git tutorial showing basic Git usage, including creating a repo, adding and tr from the repo, branching, and merging.

To demonstrate this project, I will work an exercise from Simon Wood's book, *Genera introduction with R (2nd ed.)*. The exercise concerns using generalized additive model

- ✓ main default
- cmiller
- probs6n8

Our log shows a more interesting structure now

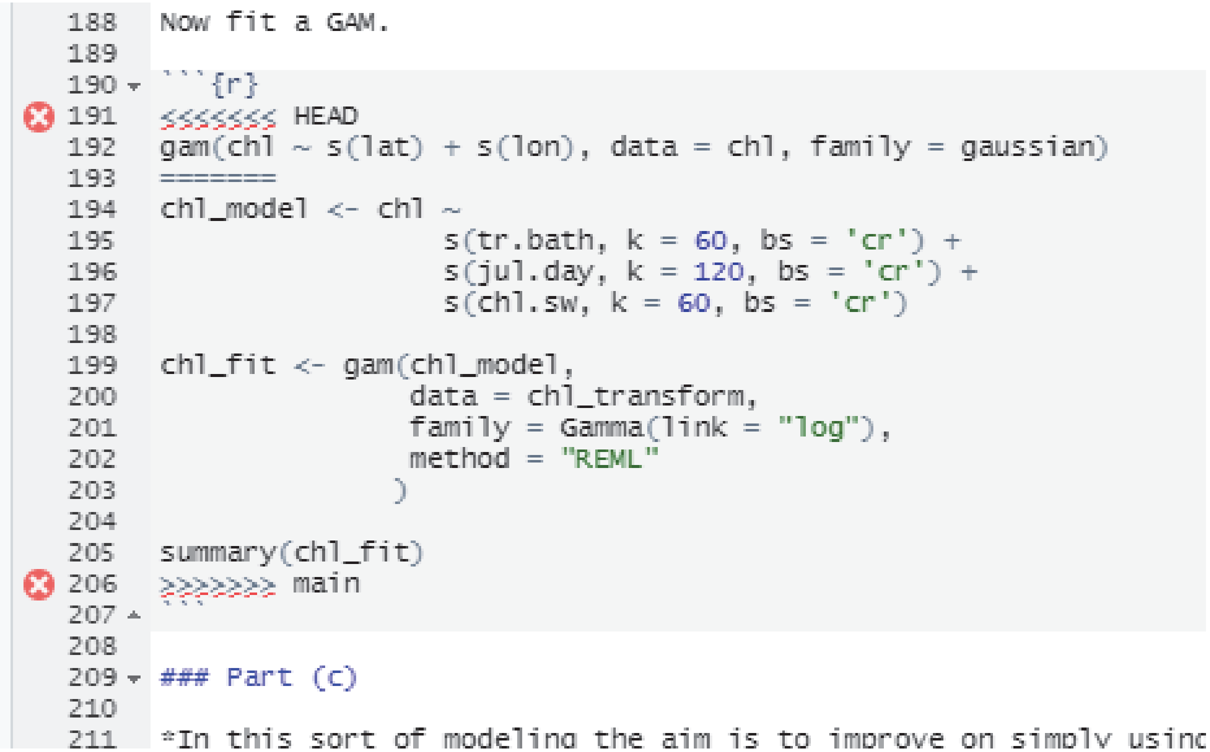


```
cmiller@CMILLER-LT MINGW64 ~/Development/git                          (main)
$ git log --graph --oneline --all
* 0569dd3 (HEAD -> main, origin/main, origin                      ting part (c)
but did not complete deviance computation on test set
* 20cfa28 Changed model to align with the one requested
* ed2d1fe (origin/cmiller) Restore old GAM model and start working on train/test
 split
* 01361ea Attempted changes to GAM only to return to Gamma, but different form
* ca00e1b Attempting part (b) using log link/gamma family
| * f3c3b46 (origin/probs6n8, probs6n8) Addi                    or probs 6 and 8
|/
* 0ce384f Added knit support for word files (generated files should be ignored)
* a2e7bd9 Completed plots and determined data transforms for part (a)
* 0f66f94 Structuring project around chlorophyll analysis example
* b2aa985 Added a README.md file describing the purpose of the project
* 3c0de05 Initial commit
```

For a more interesting visualization

Note the fork…

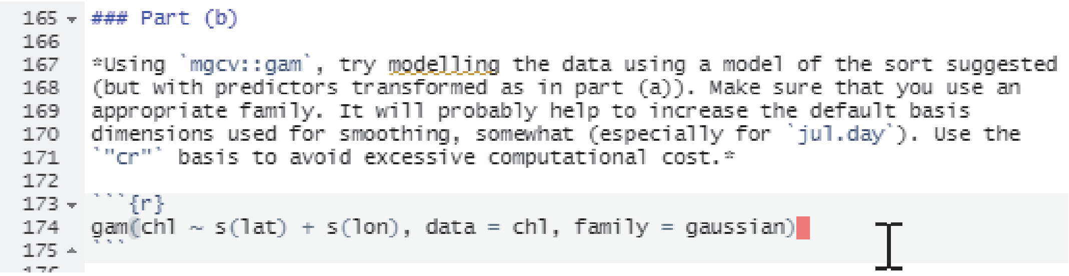
Merge conflicts happen when Git does not know how to properly merge branches, and needs manual intervention



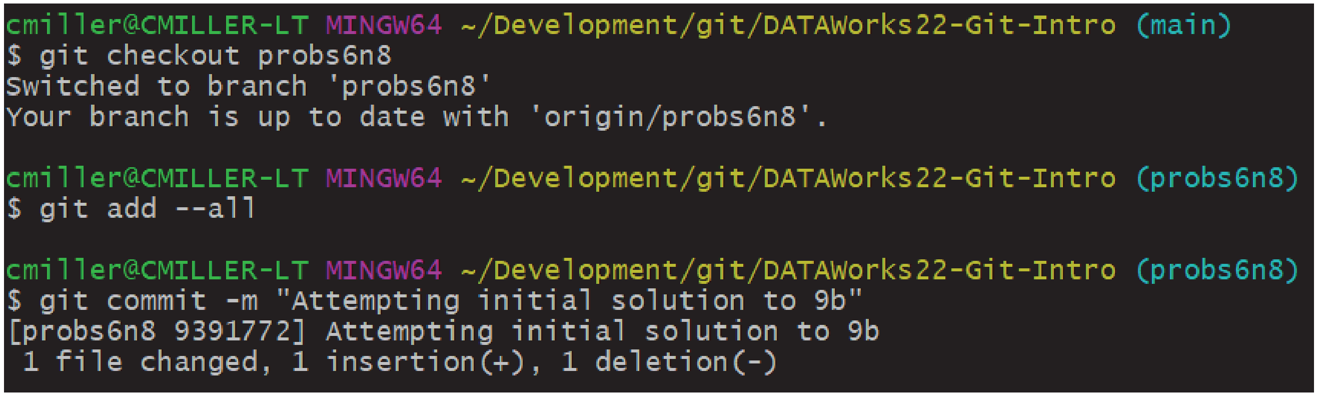
~~~
Now fit a GAM.

```{r}
<<<<<<< HEAD
gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
=======
chl_model <- chl ~
                 s(tr.bath, k = 60, bs = 'cr') +
                 s(jul.day, k = 120, bs = 'cr') +
                 s(chl.sw, k = 60, bs = 'cr')

chl_fit <- gam(chl_model,
               data = chl_transform,
               family = Gamma(link = "log"),
               method = "REML"
              )

summary(chl_fit)
>>>>>>> main
```

### Part (c)

*In this sort of modeling the aim is to improve on simply using
~~~

- Two users might edit the same line in two branches in different ways
- When those branches are merged, Git will throw an error and insert lines into files with unresolved differences
- After manually handling the conflicts with a text editor, commit the changes, then carry on; Git will view the conflicts as resolved

Let’s introduce a potential merge conflict when we merge into our collaborator’s branch…

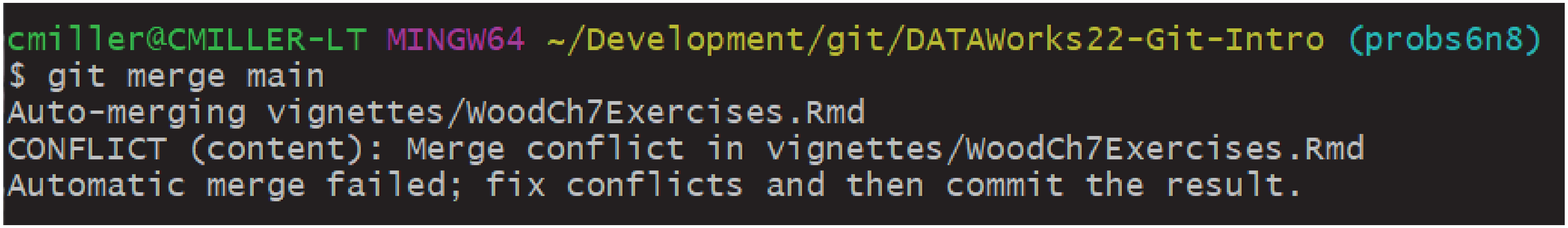


~~~
### Part (b)

*Using `mgcv::gam`, try modelling the data using a model of the sort suggested
(but with predictors transformed as in part (a)). Make sure that you use an
appropriate family. It will probably help to increase the default basis
dimensions used for smoothing, somewhat (especially for `jul.day`). Use the
`"cr"` basis to avoid excessive computational cost.*

```{r}
gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
```
~~~



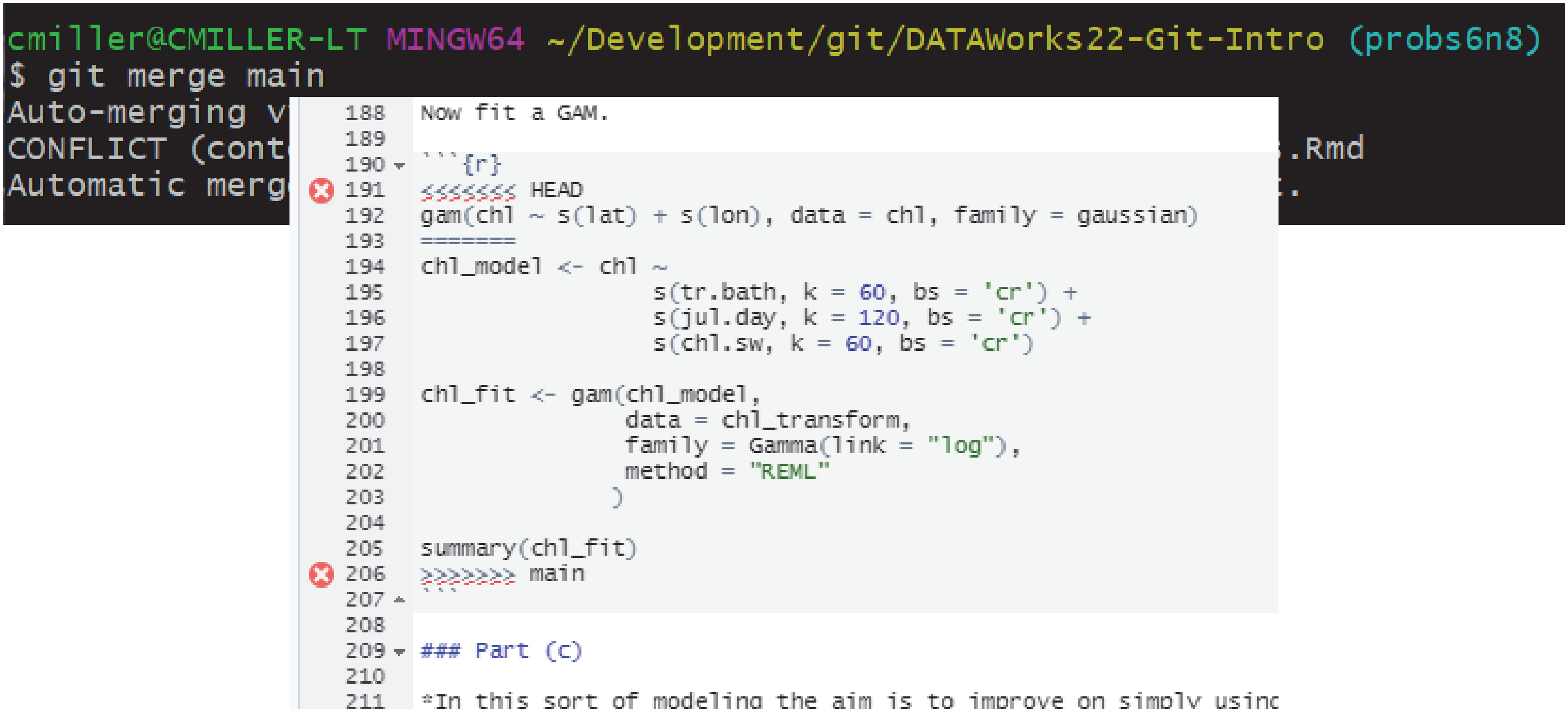
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git checkout probs6n8
Switched to branch 'probs6n8'
Your branch is up to date with 'origin/probs6n8'.

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git commit -m "Attempting initial solution to 9b"
[probs6n8 9391772] Attempting initial solution to 9b
 1 file changed, 1 insertion(+), 1 deletion(-)
```

The merger fails…



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git merge main
Auto-merging vignettes/WoodCh7Exercises.Rmd
CONFLICT (content): Merge conflict in vignettes/WoodCh7Exercises.Rmd
Automatic merge failed; fix conflicts and then commit the result.
```

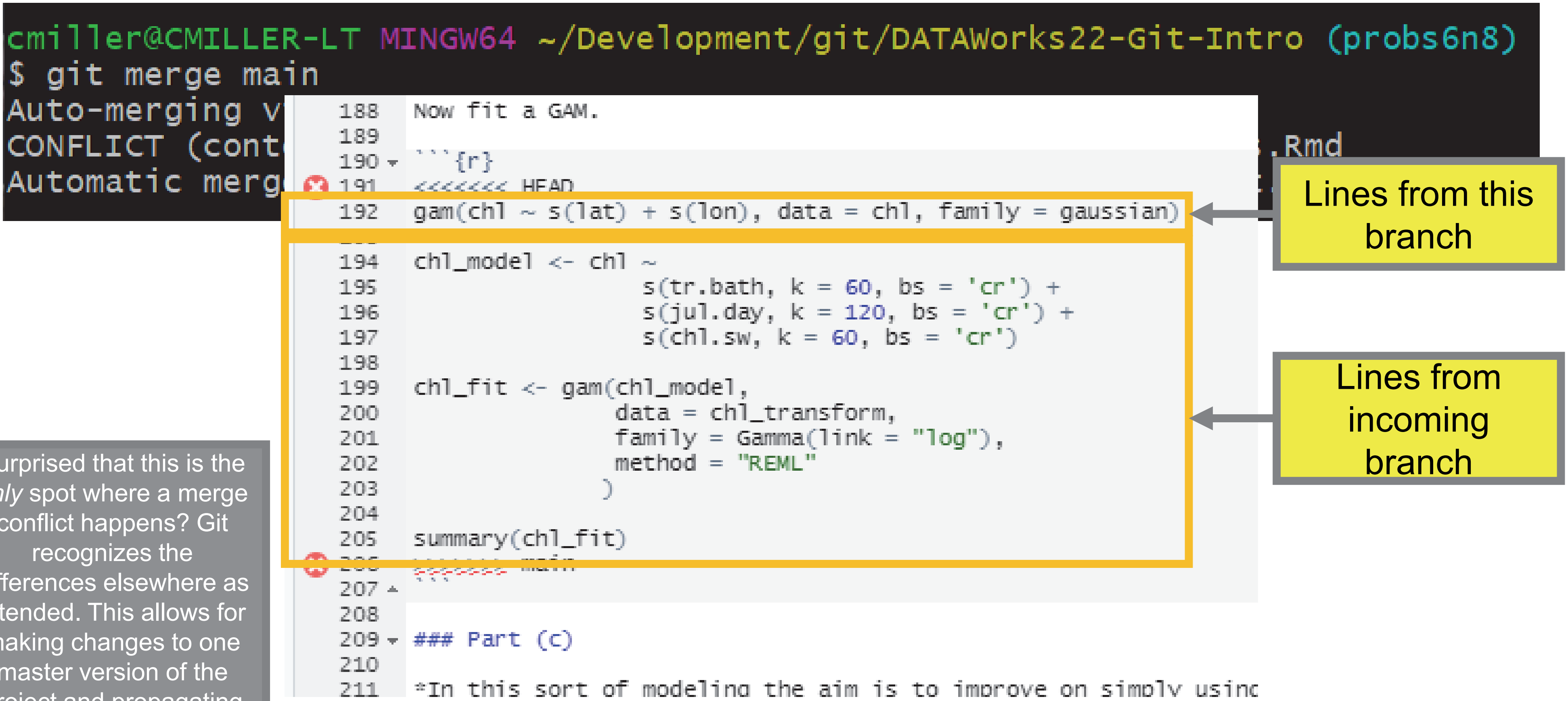
… and we can see where



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git merge main
Auto-merging v
CONFLICT (cont                                                  .Rmd
Automatic merg                                                  t.
```

~~~
Now fit a GAM.

```{r}
<<<<<<< HEAD
gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
=======
chl_model <- chl ~
                   s(tr.bath, k = 60, bs = 'cr') +
                   s(jul.day, k = 120, bs = 'cr') +
                   s(chl.sw, k = 60, bs = 'cr')

chl_fit <- gam(chl_model,
                 data = chl_transform,
                 family = Gamma(link = "log"),
                 method = "REML"
                )

summary(chl_fit)
>>>>>>> main
```


### Part (c)

*In this sort of modeling the aim is to improve on simply using
~~~

... and we can see where



```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git merge main
Auto-merging v
CONFLICT (cont                                                                        .Rmd
Automatic merg
```

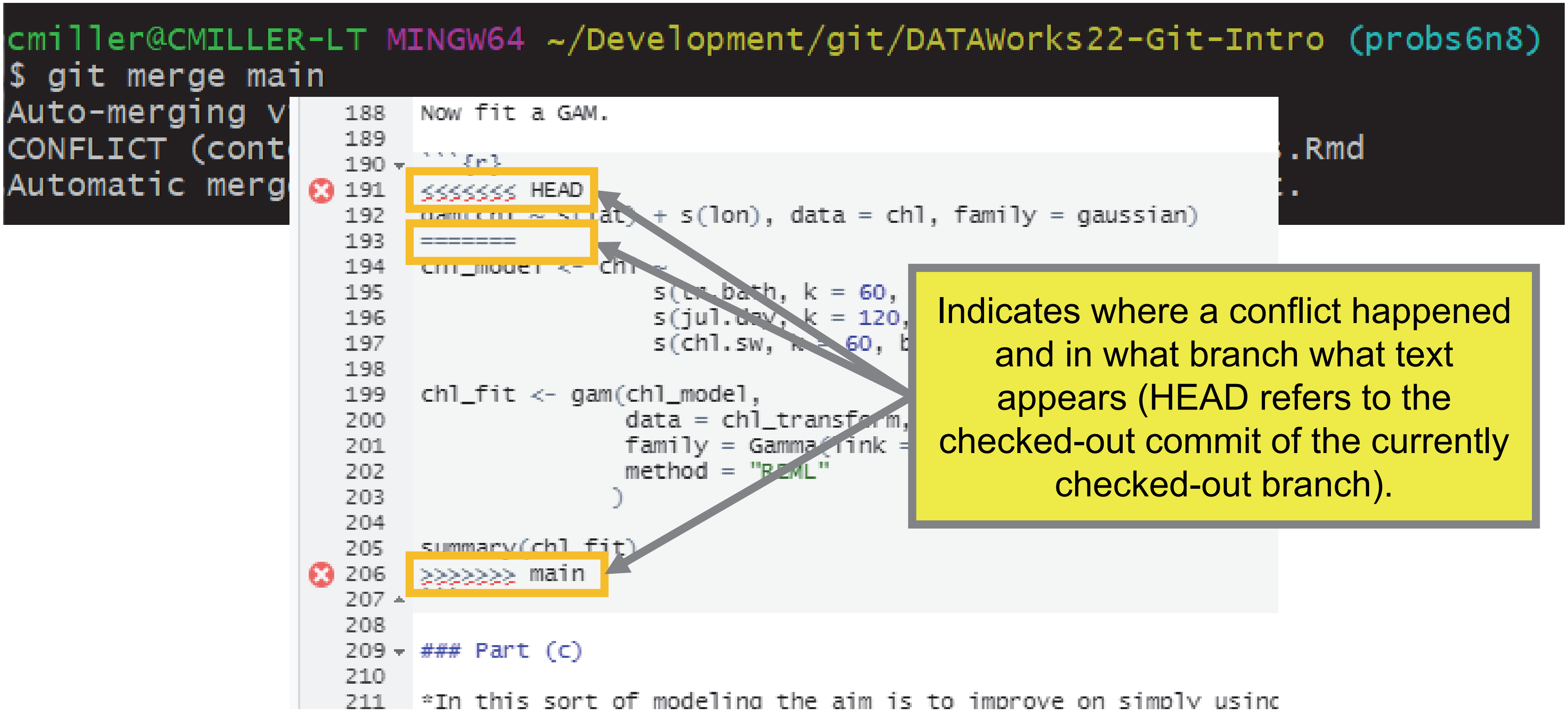
~~~
Now fit a GAM.

```{r}
<<<<<<< HEAD
gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)

chl_model <- chl ~
                 s(tr.bath, k = 60, bs = 'cr') +
                 s(jul.day, k = 120, bs = 'cr') +
                 s(chl.sw, k = 60, bs = 'cr')

chl_fit <- gam(chl_model,
               data = chl_transform,
               family = Gamma(link = "log"),
               method = "REML"
              )

summary(chl_fit)
>>>>>>> main
```

### Part (c)

*In this sort of modeling the aim is to improve on simply using
~~~

Lines from this branch

Lines from incoming branch

Surprised that this is the *only* spot where a merge conflict happens? Git recognizes the differences elsewhere as intended. This allows for making changes to one master version of the project and propagating them to other deviations of the project.

... and we can see where

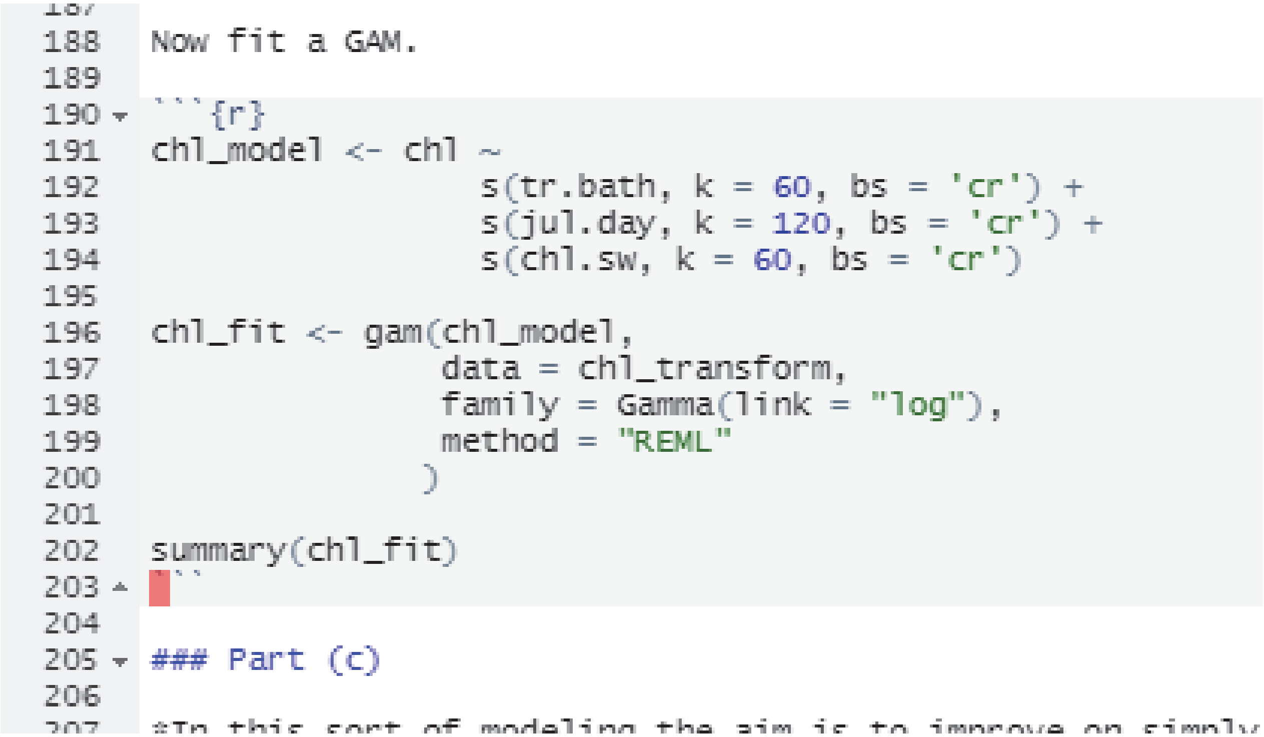


```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git merge main
Auto-merging v
CONFLICT (cont                                                            .Rmd
Automatic merg                                                            .
```

~~~
Now fit a GAM.

```{r}
<<<<<<< HEAD
gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
=======
chl_model <- chl ~
                    s(tr.bath, k = 60,
                    s(jul.day, k = 120,
                    s(chl.sw, k = 60, b

chl_fit <- gam(chl_model,
                  data = chl_transform,
                  family = Gamma(link =
                  method = "REML"
                )

summary(chl_fit)
>>>>>>> main


### Part (c)

*In this sort of modeling the aim is to improve on simply usinc
~~~

Indicates where a conflict happened and in what branch what text appears (HEAD refers to the checked-out commit of the currently checked-out branch).

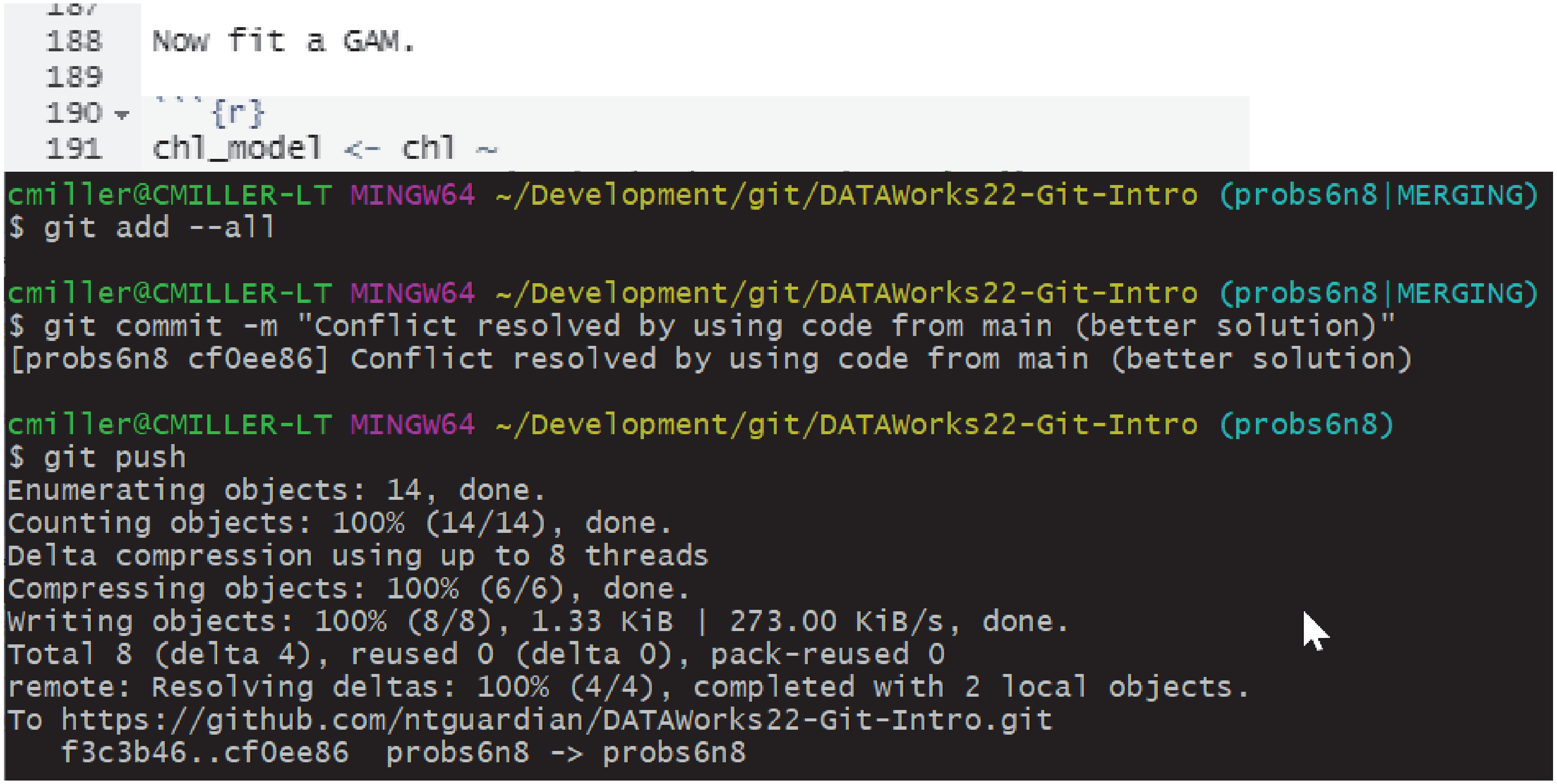
To fix, edit the file to resolve the conflict…



~~~
Now fit a GAM.

```{r}
chl_model <- chl ~
                 s(tr.bath, k = 60, bs = 'cr') +
                 s(jul.day, k = 120, bs = 'cr') +
                 s(chl.sw, k = 60, bs = 'cr')

chl_fit <- gam(chl_model,
               data = chl_transform,
               family = Gamma(link = "log"),
               method = "REML"
              )

summary(chl_fit)
```

### Part (c)

#In this sort of modeling the aim is to improve on simply
~~~

In this case we deleted lines from one commit, but we could have done anything we wanted to resolve the conflict, even adding brand new lines. Git doesn't care what you do to resolve the conflict.

… Then add and commit your changes (just like any other commit)



~~~
Now fit a GAM.

```{r}
chl_model <- chl ~
~~~

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8|MERGING)
$ git add --all

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8|MERGING)
$ git commit -m "Conflict resolved by using code from main (better solution)"
[probs6n8 cf0ee86] Conflict resolved by using code from main (better solution)

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (probs6n8)
$ git push
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.33 KiB | 273.00 KiB/s, done.
Total 8 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
   f3c3b46..cf0ee86  probs6n8 -> probs6n8
```

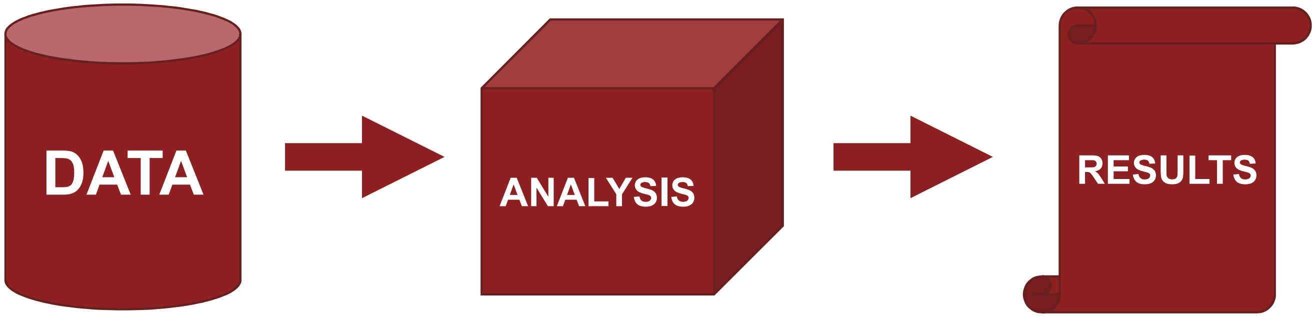
Basic branch management commands

Command	Description	Examples
`git branch`	Branch management command	`git branch -d somebranch    # Delete branch` `git branch -m old new        # Rename branch`
`git checkout`	Retrieve data from commits; discussed more later	`git checkout identifier`
`git merge`	Merge one branch into another branch	`git merge branchname`

Conclusion

Git can be an important part of making research reproducible

- As a research project carries on, the data and analysis techniques used are prone to change over time
- Results ultimately depend on the version of data and analysis used at any given time
- With Git, the evolution of the data and the analysis methods is tracked over time
- This allows for someone to see how the results of the project depend on the version of data and analysis used, and track down the specific version of these that made a particular analysis and what changed to produce a more current analysis



This is only the start of learning Git

- Git is powerful and frankly intimidating software
- It is not the most simple version control system out there, but it’s popular for good reasons
- I often default to using Git for almost any project, regardless of whether I plan to share or not, or how big the project might be
- The more you use Git, the more familiar it will become
- Remember: `--help`, Google, and StackExchange are your friends

Commands I would love to talk about if I had time

Command	Description
`git log`	See commit history; setting options can allow for making useful displays
`git diff`	See what files changed between commits and in what ways
`git bisect`	Find the first “bad” commit (say where a bug was introduced)
`git stash`	“Stash” changes, then reset to the most recent commit
`git grep`	Search an entire repo using regular expressions

Cheat Sheet for Git

Command	Description	Examples
`git config`	Configure Git	`git config --global user.name “Jane Doe”` `git config --global user.email jdoe@ida.org`
`git init`	Create a new repo	`git init`
`git clone`	Clone an existing repository	`git clone “https:...”`
`git add`	Add changes to (or brand new) files to a staged commit	`git add myfile.txt` `git add --all`
`git commit`	Commit changes	`git commit –m “Message”`
`git pull`	For repos hosted remotely (such as GitHub), update repo with changes from host	`git pull`
`git push`	For repos hosted remotely (such as GitHub), send changes to host	`git push`
`git rm`	Remove a file from tracking	`git rm myfile.txt`
`git reset`	Git’s “undo” command	`git reset --hard    # CAUTION: Deletes changes`
`git branch`	Branch management command	`git branch –d somebranch    # Delete branch` `git branch –m old new        # Rename branch`
`git checkout`	Retrieve data from commits	`git checkout identifier`
`git merge`	Merge one branch into another branch	`git merge branchname`

BACKUP

Navigating your computer using Bash

Command	Description	Examples

`ls`	List information about files or directories	`ls         # List files`
		`ls -lh     # List files with description`
		`ls -lha    # List all files with description`

`cd`	Change the current working directory	`cd “My Documents”    # Go to My Documents folder`
		`cd ..                # Go to parent directory`
		`cd ~/                # Go to home directory`
		`cd -                 # Go to last visited`

`pwd`	Print the current working directory	`pwd`
`head/tail`	Show the first few lines/last few lines of a text file (resp.)	`head “My file.txt”` `tail my_file.txt`
`cat`	Can be used to show file contents (but has different purpose)	`cat “My file.txt”`
`mv`	Move a file; also used to rename files	`mv file.txt dir/file.txt` `mv oldname.txt newname.txt`
`cp`	Copy a file	`cp original.txt new.txt`
`mkdir`	Create a new directory	`mkdir my_new_repo`
`rm`	Remove a file	`rm filename.txt` `rm -r my_new_repo/*    # Delete subcontents`
`rmdir`	Remove a directory	`rmdir my_new_repo`
`exit`	Exit the shell	`exit`

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION

1. REPORT DATE 04-2022	**2. REPORT TYPE** Final	**3. DATES COVERED** **START DATE**	**END DATE** Apr 2022

4. TITLE AND SUBTITLE
DATAWorks 2022 - Introduction to git

5a. CONTRACT NUMBER HQ0034-19-D-0001	**5b. GRANT NUMBER**	**5c. PROGRAM ELEMENT NUMBER**
5d. PROJECT NUMBER BD-09-2299	**5e. TASK NUMBER** 229990	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Miller, Curtis, G.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses
730 East Glebe Road
Alexandria, Virginia 22305

8. PERFORMING ORGANIZATION REPORT NUMBER
NS D-33021
H 2022-000108

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Director, Operational Test and Evaluation
1700 Defense Pentagon
Washington, DC 20301

10. SPONSOR/MONITOR'S ACRONYM(S)
DOT&E

11. SPONSOR/MONITOR'S REPORT NUMBER

12. DISTRIBUTION/AVAILABILITY STATEMENT
Public release approved. Distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Version control software manages, archives, and (optionally) different versions of files. Git is the most popular program for version control and serves as the backbone for websites such as Github, Bitbucket, and others. In this mini-tutorial we will introduce basics of version control in general, Git in particular. We explain what role Git plays in a reproducible research context. The goal of the course is to get participants started using Git. We will create and clone repositories, add and track files in a repository, and manage git branches. We also discuss a few Git best practices.

This tutorial was first given at DATAWorks Conference 2022 in Alexandria, Virginia.

15. SUBJECT TERMS
Git; Reproducible Research; reproducible analyses; Data Management; tutorial

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	143

19a. NAME OF RESPONSIBLE PERSON John T. Haman	**19b. PHONE NUMBER** 703-845-2132