



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
villard@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

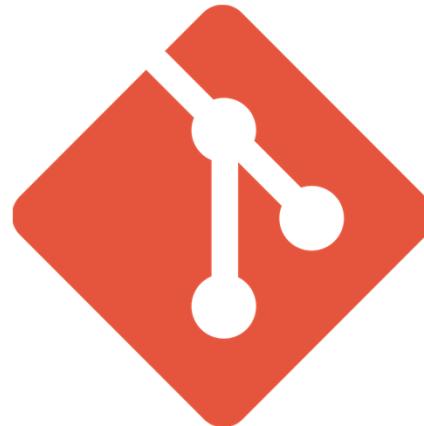
Introduction

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



git

Getting Started

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

The screenshot shows the official Git website at <http://git-scm.com/>. The page features a large header with the Git logo and the tagline "distributed-is-the-new-centralized". Below the header is a search bar. A central graphic illustrates a distributed version control system with multiple repositories connected by bidirectional arrows. To the left, there's a section about Git's advantages over other systems, followed by sections for documentation, downloads, and community. A prominent "Download for Windows" button is displayed on a computer monitor icon. At the bottom, there's a section titled "Companies & Projects Using Git" featuring logos for Google, Facebook, Microsoft, Twitter, LinkedIn, Netflix, Camel, and PostgreSQL.

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](#).

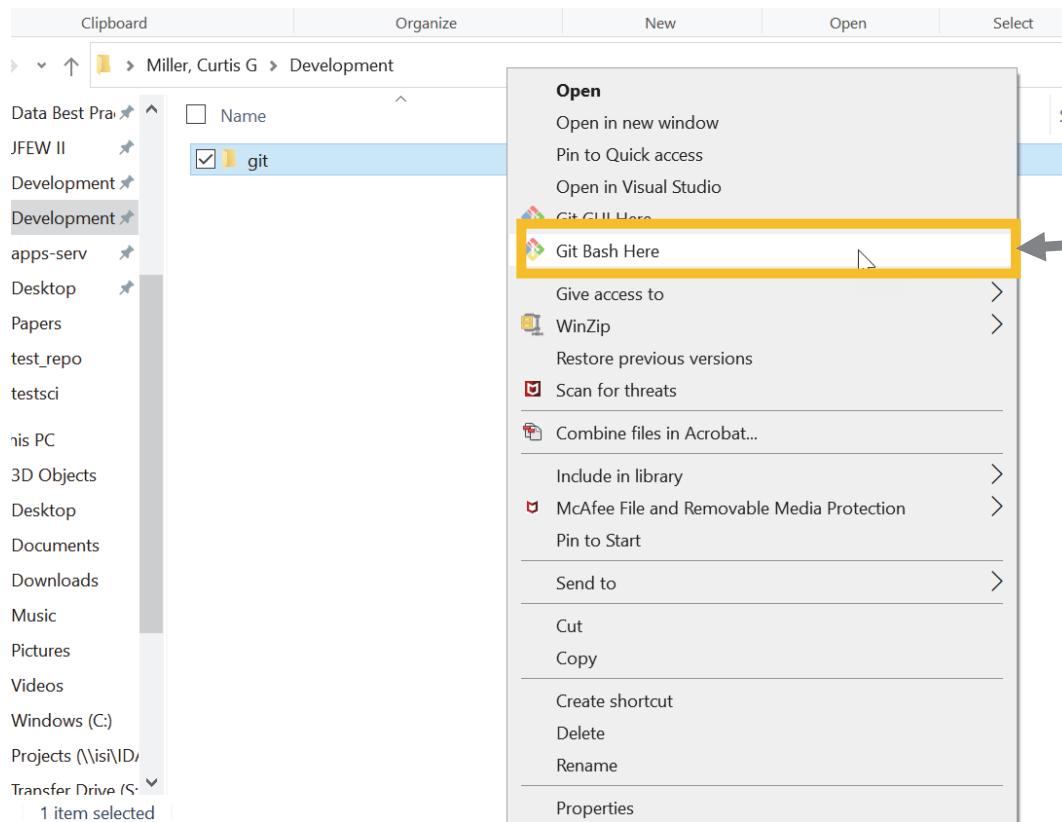
Companies & Projects Using Git

Google FACEBOOK Microsoft LinkedIn PostgreSQL

- 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

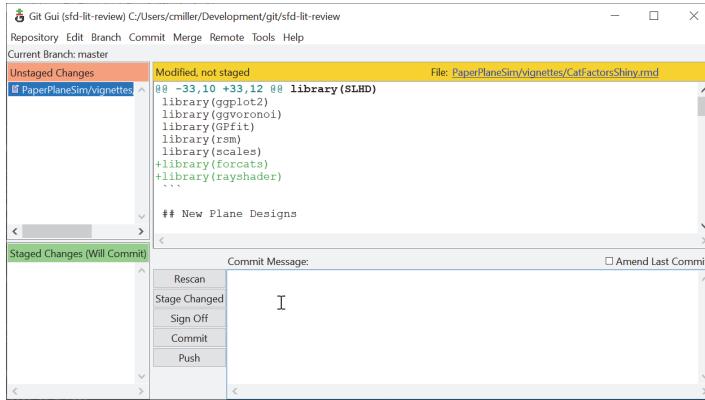
I'm going to assume you have Git installed now.

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



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



```
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

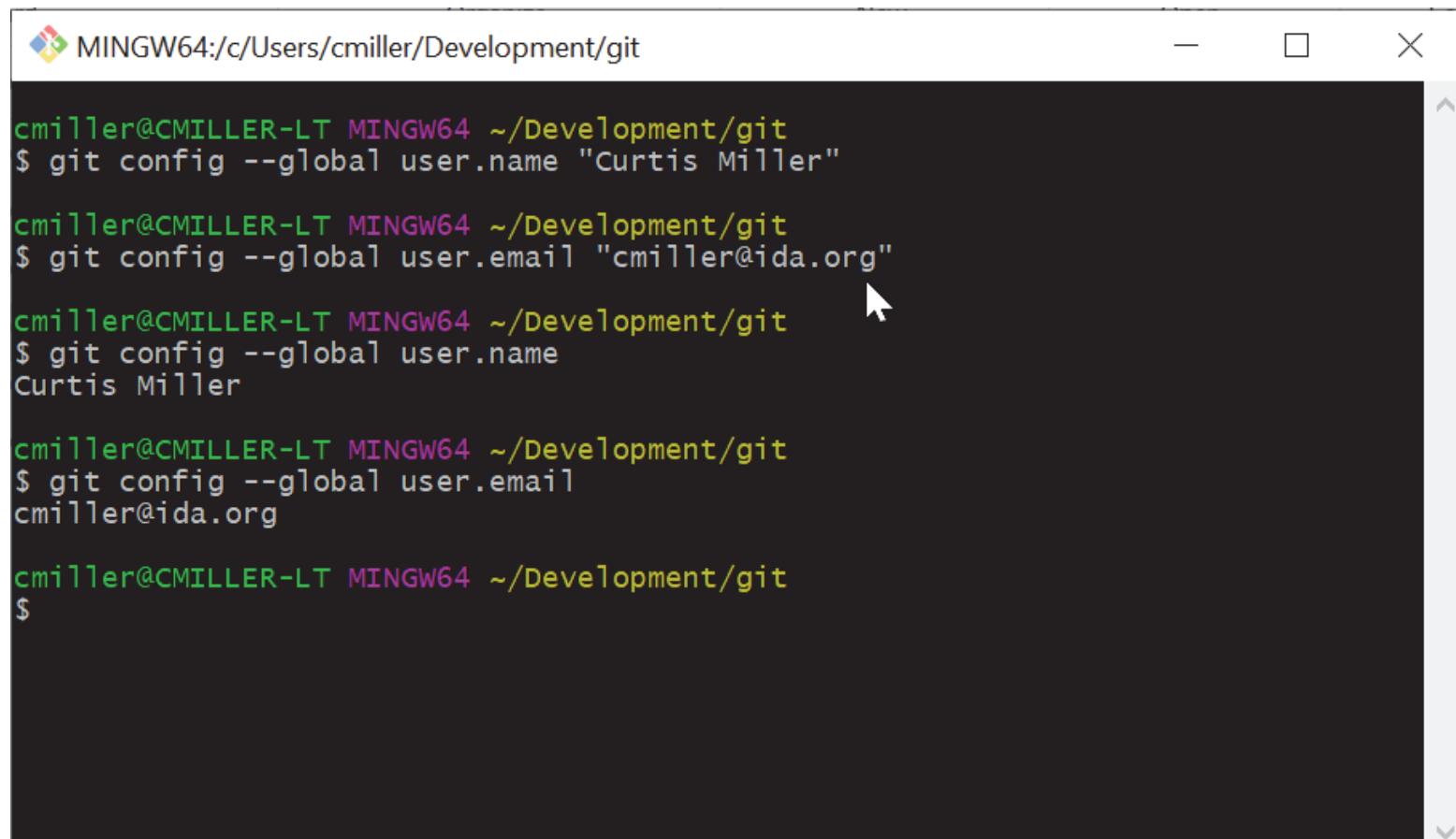
no changes added to commit (use "git add" and/or "git commit -a")
cmiller@CMILLER-LT MINGW64 ~/Development/git/sfd-lit-review (master)
$ |
```

A screenshot of a terminal window titled "MINGW64:c/Users/cmiller/Development/git/sfd-lit-review". The user runs several Git commands: navigating to the "Development" directory, listing files, changing to the "git" directory, listing its contents, changing to the "sfd-lit-review" subdirectory, checking the status of the repository, and finally listing the contents of the current directory again. The terminal uses color-coded syntax highlighting for commands and output.

GUI: Graphical User Interface

- 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

Set up initial user parameters using git-config



The screenshot shows a terminal window titled "MINGW64:/c/Users/cmiller/Development/git". The user is setting global Git configuration parameters. The commands entered are:

```
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

The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "MINGW64:/c/Users/cmiller/Development/". Below that is a command-line interface with several lines of text:

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

Annotations with yellow boxes and arrows point to specific parts of the terminal output:

- An annotation labeled "Username (before @) and computer name (after @)" points to the first line of the command history, "cmiller@CMILLER-LT".
- An annotation labeled "Working directory" points to the path "~/Development/git" in the second line of the command history.
- An annotation labeled "Prompt awaiting your command" points to the "\$" sign at the end of the fourth line of the command history.

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.

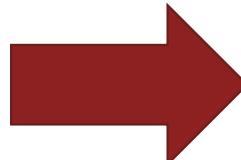
Similarly, tell Git my e-mail address

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...]  
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
```

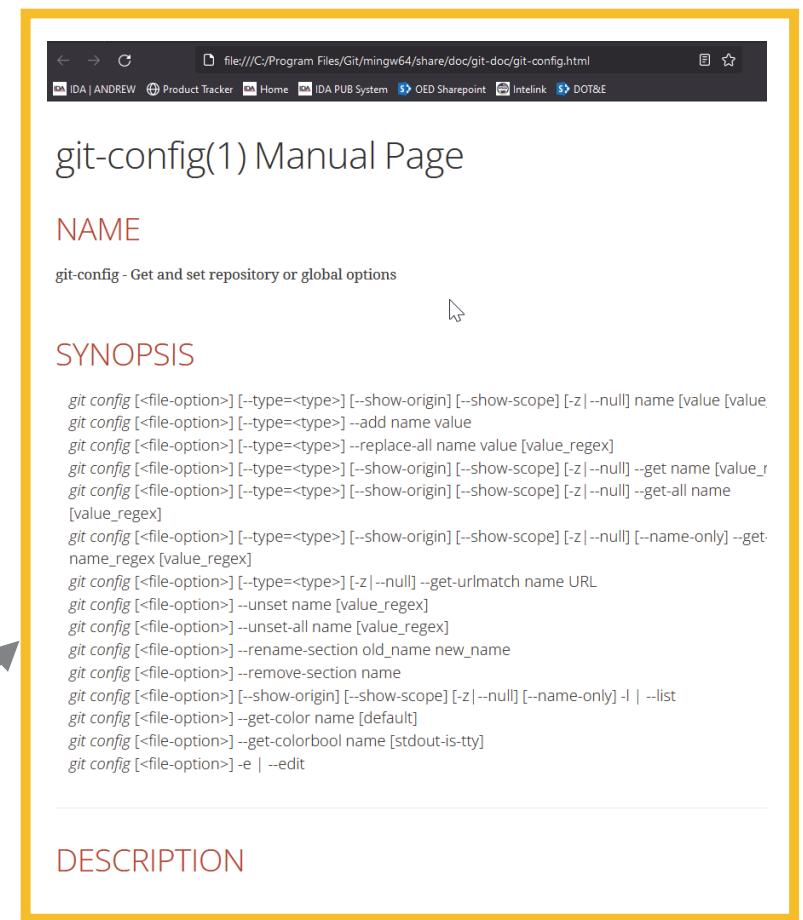
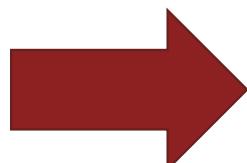
DESCRIPTION

“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
```

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



git-config(1) Manual Page

NAME

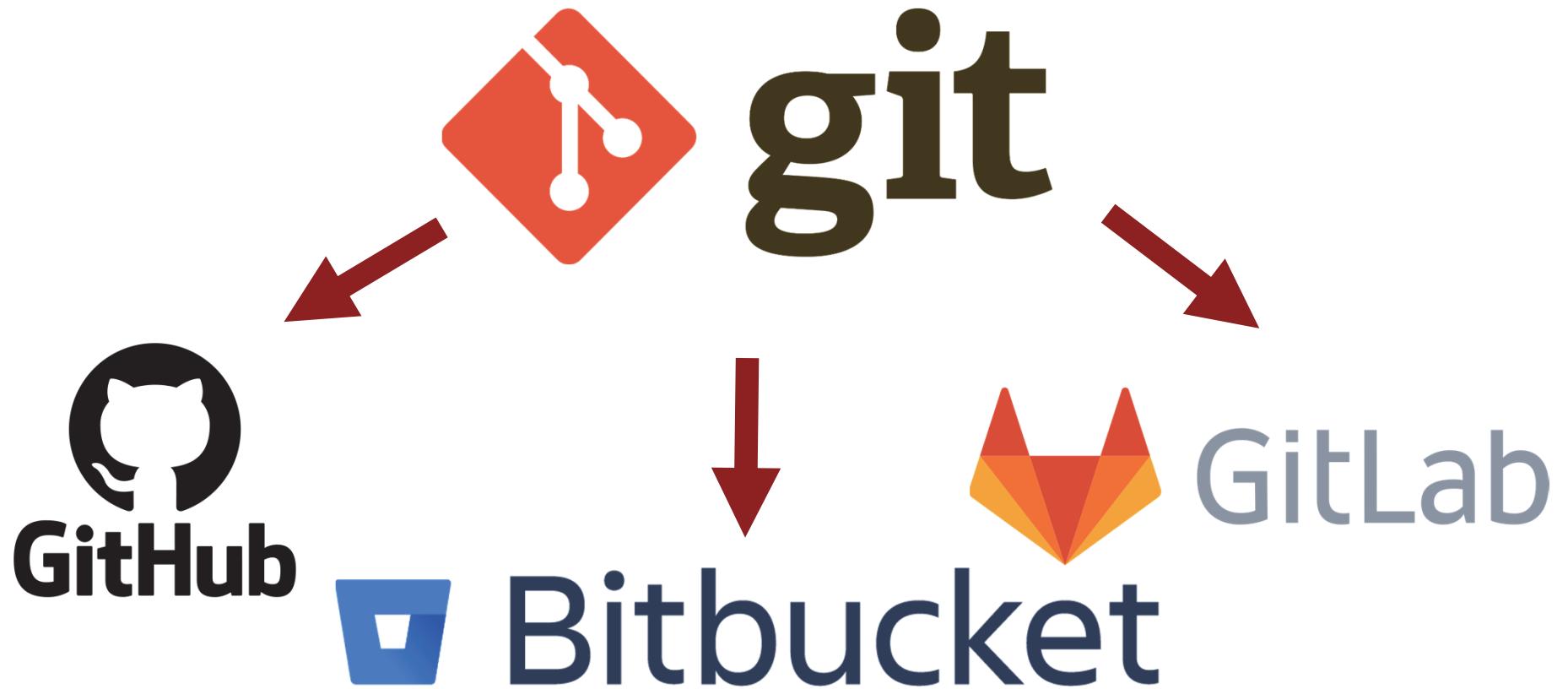
git-config - Get and set repository or global options

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...]  
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 [value...]  
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-colorbool name [default]  
git config [<file-option>] --get-colorbool name [stdout-is-tty]  
git config [<file-option>] -e | --edit
```

DESCRIPTION

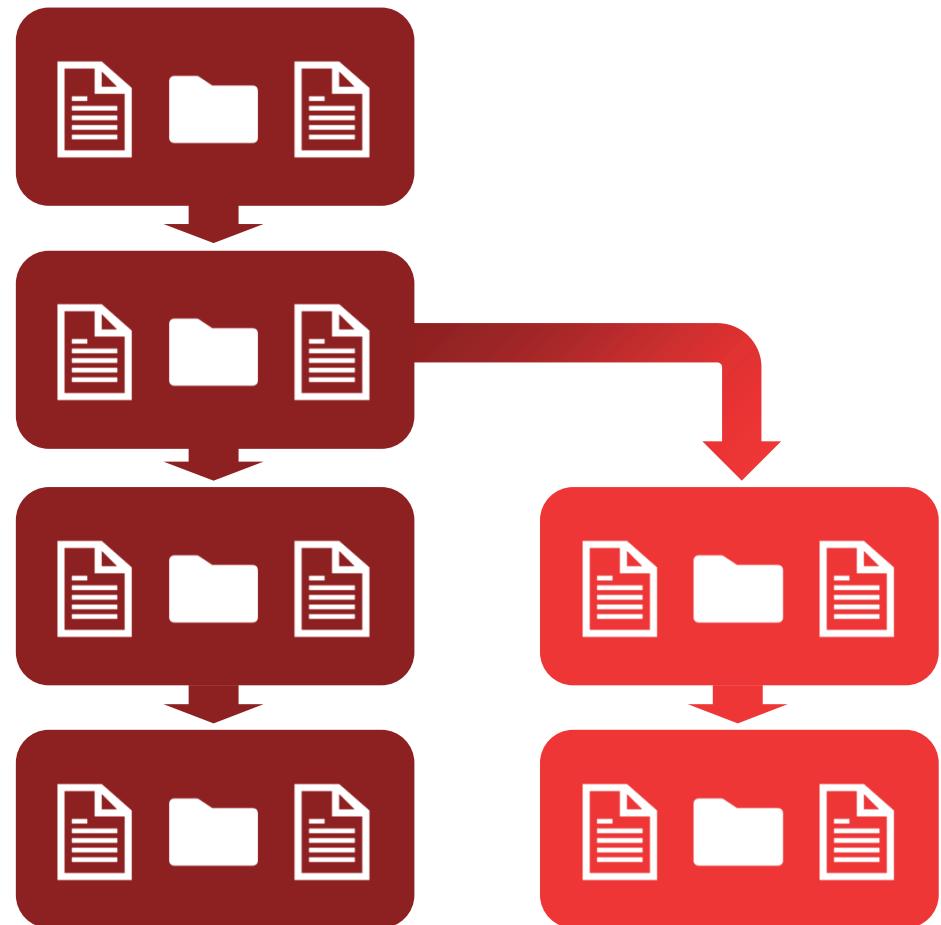
Git powers popular code sharing websites such as GitHub



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)

The image displays three separate screenshots of web-based Git hosting platforms:

- Github.com**: The homepage features a dark background with a central illustration of a small character floating near Earth, surrounded by network connections. Text on the page includes "Where the world builds software" and statistics: "73+ million Developers", "4+ million Organizations", "200+ million Repositories", and "84% Fortune 100".
- GitLab.com**: The homepage has a light-colored header with the GitLab logo and navigation links. The main content area is titled "The DevOps Platform has arrived." It includes a sub-headline "Deliver software faster with better security and collaboration in a single platform.", a "Get free trial" button, and a "Watch demo" video player.
- Bitbucket.com**: The homepage features a light blue header with the Bitbucket logo and navigation links. It highlights "Guess who's back? Join us for Team '22 in Las Vegas or attend digitally! Event starts April 5th. Sign up now." Below this, it says "Built for professional teams" and describes Bitbucket as "more than just Git code management". It includes a "Get it free" button and a snippet of a pull request titled "bugfix/BUG-123 remove extra padding" showing code changes in a file named "select.css".

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

The screenshot shows the GitHub homepage with a dark theme. At the top, there's a navigation bar with links to IDA | ANDREW, Product Tracker, Home, IDA PUB System, OED Sharepoint, Intelink, and DOT&E. Below the navigation is a search bar labeled "Search or jump to..." and a sidebar with "Recent Repositories". The sidebar lists several repositories owned by the user "ntguardian", such as "CPAT", "Training-Systems-Using-Python-Statistical-Modeling", "radian", "MCHT", "dataviscourse-pr-congressrelations", "Curtis-Miller-Private-Repo", and "ntguardian.github.io". There are buttons for "New" and "Find a repository...". To the right of the sidebar are two promotional banners: one for learning Git and GitHub without code, and another for joining GitHub Global Campus.

Recent Repositories

New

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!

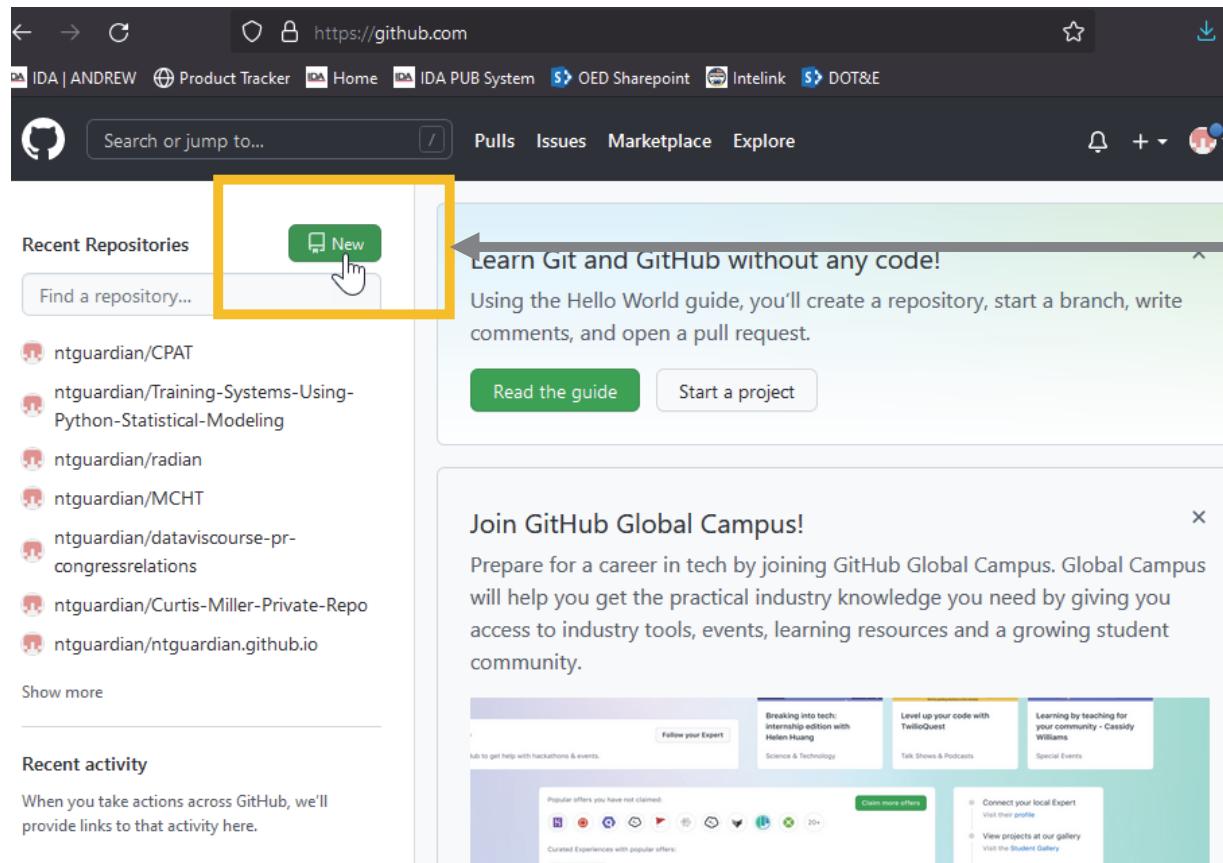
Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide Start a project

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



A screenshot of the GitHub homepage (<https://github.com>). The top navigation bar includes links for IDA | ANDREW, Product Tracker, Home, IDA PUB System, OED Sharepoint, Intelink, and DOT&E. Below the bar are buttons for Pulls, Issues, Marketplace, and Explore. A search bar says "Search or jump to...". On the left, there's a "Recent Repositories" sidebar with a "Find a repository..." input field. A yellow box highlights the "New" button, which has a hand cursor icon pointing at it. A modal window titled "Learn Git and GitHub without any code!" contains the text: "Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request." It has "Read the guide" and "Start a project" buttons. Another modal window titled "Join GitHub Global Campus!" discusses career opportunities and features like "Follow your Expert", "Breaking into tech: internship edition with Helen Huang", "Level up your code with TwilioQuest", and "Learning by teaching for your community - Cassidy Williams". A callout box on the right says "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 * 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.

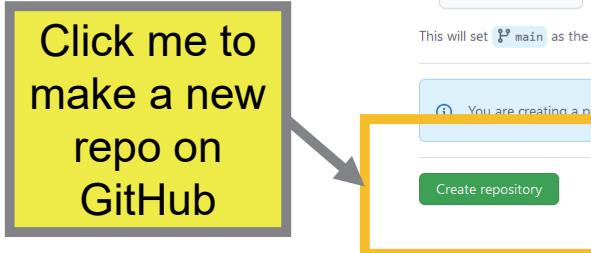
Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).

Choose a license A license tells others what they can and can't do with your code. [Learn more](#).

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.



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

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The repository is public and has one commit. The commit history shows:

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

Below the commit history, there is a message encouraging the user to add a README: "Help people interested in this repository understand your project by adding a README." A green button labeled "Add a README" is visible.

On the right side of the page, there is an "About" section with the following information:

- This is a repository for demonstrating how to use Git.
- 0 stars
- 1 watching
- 0 forks

There are also sections for "Releases" (No releases published) and "Packages" (No packages published).

Now we can view our repo online

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The URL 'https://github.com/ntguardian/DATAWorks22-Git-Intro' is highlighted in yellow at the top. A callout box points to it with the text 'Repo identifier (note connection to URL)'. Another callout box points to the repository name 'ntguardian / DATAWorks22-Git-Intro' with the text 'Repo URL, important for sharing'. A third callout box points to the 'main' branch dropdown with the text 'Which branch we are viewing (will discuss branches later)'. A fourth callout box points to the file list on the left with the text 'Some files already in the repo, created by GitHub (click to view)'. The file list shows '.gitignore' and 'LICENSE' files, both created at 'now' with 'Initial commit'. A cursor arrow is visible at the bottom center.

Repo identifier (note connection to URL)

Repo URL, important for sharing

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

The screenshot shows a GitHub file viewer for the LICENSE file in the DATAWorks22-Git-Intro repository. The top bar indicates the file is in the main branch. The page title is "DATAWorks22-Git-Intro / LICENSE". On the left, there's a sidebar with the repository name and a link to the repository's page. The main content area shows the MIT License text. It includes sections for "Permissions" (Commercial use, Modification, Distribution, Private use), "Limitations" (Liability, Warranty), and "Conditions" (License and copyright notice). A note states that this is not legal advice. Below the license text, it shows the initial commit by ntguardian and a recent commit. A "History" button is highlighted with a yellow box. At the bottom, the file content is displayed in a code editor-like interface with lines numbered 1 to 21. Buttons for "Raw", "Blame", and "Edit" are at the bottom right.

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

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The repository is public. The 'Code' tab is selected. A dropdown menu is open over the 'Code' button, showing options: 'Clone', 'HTTPS' (selected), 'SSH', 'GitHub CLI (New)', a copy link button, and instructions 'Use Git or checkout with SVN using the web URL.' Below the dropdown, there's a list of files: '.gitignore' and 'LICENSE'. A large blue box highlights the 'Clone' section of the dropdown.

ntguardian / DATAWorks22-Git-Intro Public

Code Issues Pull requests Actions Projects Wiki Security

main Go to file Add file Code

Clone
HTTPS SSH GitHub CLI (New)
https://github.com/ntguardian/DATAWorks

.gitignore LICENSE

Help people interested in this repository README

Open with GitHub Desktop

Download ZIP

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

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The 'Code' tab is selected. A dropdown menu is open under the 'Code' button, showing options: 'Clone', 'HTTPS' (selected), 'SSH', 'GitHub CLI (New)', a copy icon, and instructions 'Use Git or checkout with SVN using the web URL.' Below the dropdown are links for 'Open with GitHub Desktop' and 'Download ZIP'. Two yellow callout boxes with arrows point to specific elements: one points to the 'Code' button with the text 'Click this to open a dialog to allow easy cloning', and another points to the copy icon in the dropdown with the text 'Click to copy the repo URL for cloning'.

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



The screenshot shows a terminal window with the following text:

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

Two yellow callout boxes with arrows point to specific parts of the command:

- A box labeled "The `git clone` command" points to the first word of the command.
- A box labeled "The URL you copied; wrap in quotes to be safe" points to the URL part of the command.

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

The screenshot shows a Windows File Explorer window with the path `c:\Users\cmiller\Development\git`. Inside this folder, there is a subfolder named `DATAWorks22-Git-Intro`. This folder contains several files and subfolders: `DATAWorks22-Git-Intro`, `latex-templates`, `metamodel-lit-review`, and `sfd-lit-review`. Below the main folder, there is a folder named `.git` which contains three files: `.gitignore` and `LICENSE`, along with a file named `README.md` (which is not visible in the screenshot). A yellow callout box points to the `DATAWorks22-Git-Intro` folder with the text "Now it's on our computer". Another yellow callout box points to the `.git` folder with the text "What's inside the directory; it matches what we saw on GitHub (except for .git; ignore this folder)". A third yellow callout box points to the terminal output with the text "Output once you run the command; everything is okay".

```
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.
```

Now it's on our computer

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)

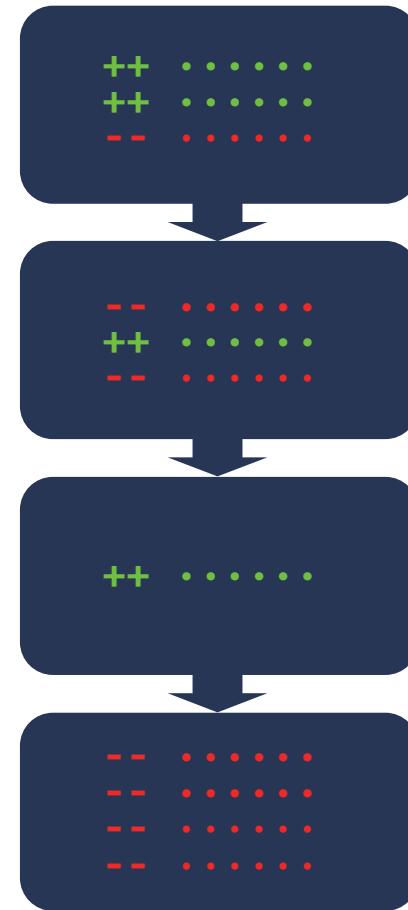
Summary: Making and getting repos

Command	Description	Examples
<code>git init</code>	Create a new repo	<code>git init</code>
<code>git clone</code>	Clone an existing repository	<code>git clone "https://github.com/someuser/someproject"</code>

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!

The screenshot shows a code editor window with a file named `README.md` open. The content of the file is:

```
1 # DATAWorks 2022 Introduction to Git
2
3 This is a Git tutorial showing basic Git usage, including creating a repo,
4 adding and tracking files, pushing/pulling from the repo, branching, and
5 merging.
```

To the right of the code editor is a file browser window showing the contents of a directory named `DATAWorks22-Git-Intro`. The browser lists the following files:

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

A yellow callout box with a black border is overlaid on the file browser area, containing the text:

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.

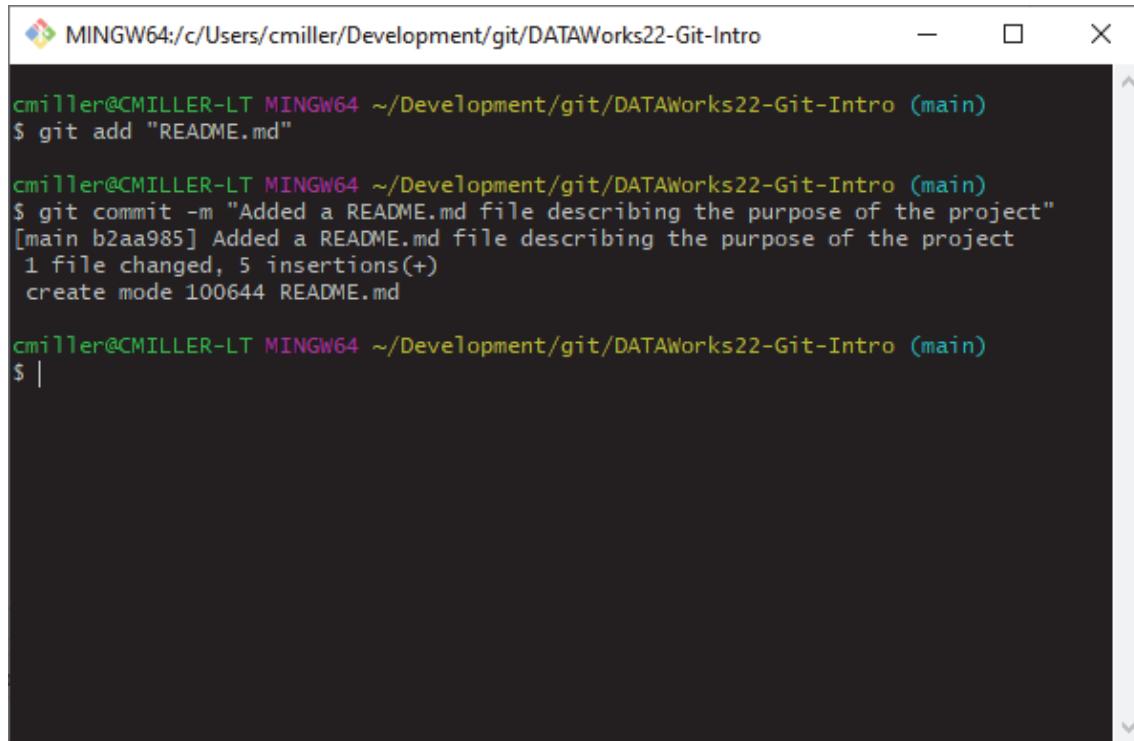
A yellow box highlights the `README.md` file in the file browser. A yellow arrow points from the text in the callout box to the `README.md` file in the file browser.

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.

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



The screenshot shows a terminal window titled "MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro". The user has run the command \$ git add "README.md". Then, they run \$ git commit -m "Added a README.md file describing the purpose of the project". The output shows a commit message: [main b2aa985] Added a README.md file describing the purpose of the project. It indicates 1 file changed, 5 insertions(+), and creates mode 100644 README.md.

```
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

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

Name the file with the changes to track

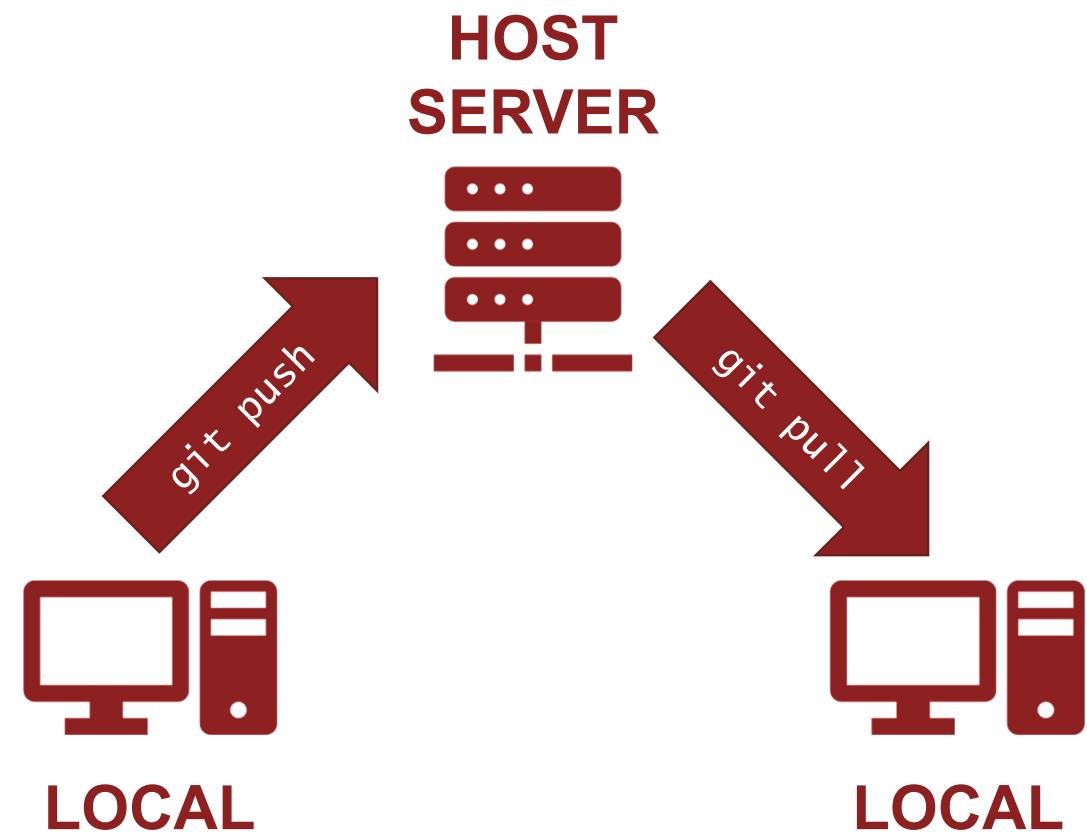
```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
$ git add "README.md"
$ git commit -m "Added a README.md file describing the purpose of the project"
[main b2aa98] Added a README.md file describing the purpose of the project
 1 file changed, 5 insertions(+)
 create mode 100644 README.md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

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

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

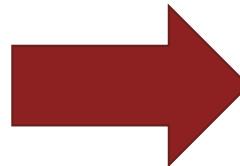
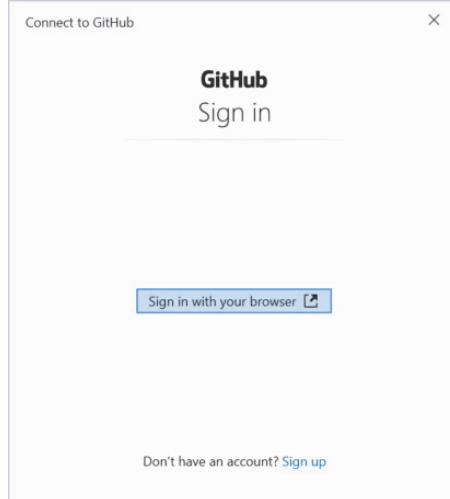
Use `git push`/`git pull` to keep your local repo copy up-to-date

- 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

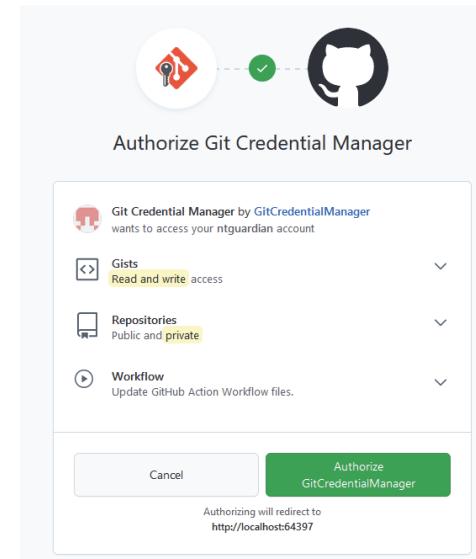


Pushing to GitHub for the first time may require authentication

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



```
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'.
```

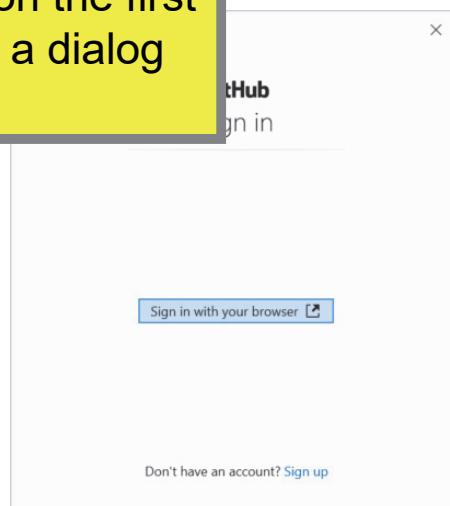


Pushing to GitHub for the first time may require authentication

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

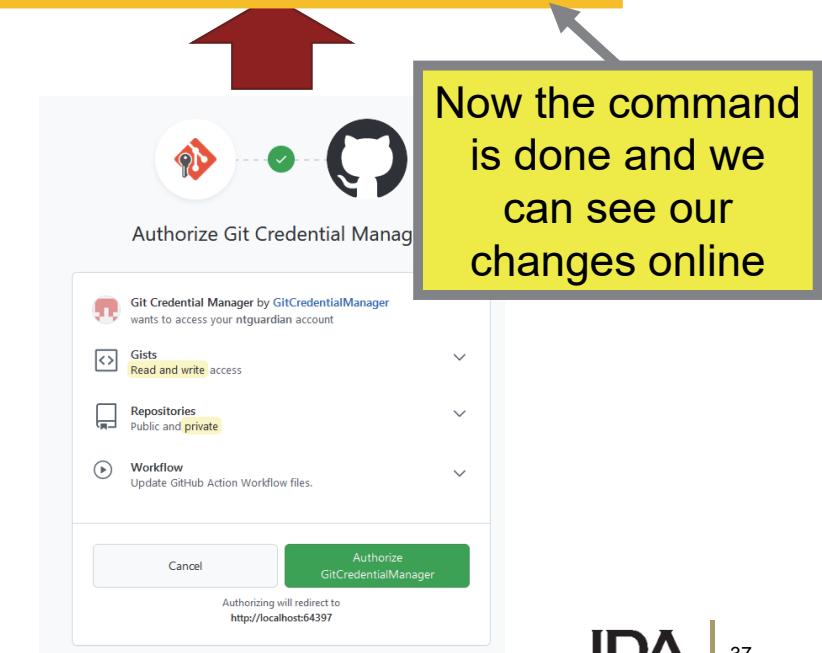
```
$ git push -u
```

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



Follow the
instructions...

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

A screenshot of a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The repository is public. The 'Code' tab is selected. Below it, there's a summary: 'main' branch, 1 branch, 0 tags. A 'Code' dropdown menu is open. The commit history shows:

File	Commit Message	Time Ago
.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

Below the commits, the README.md file is shown with its content:

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!

The screenshot shows the RStudio interface with two files open:

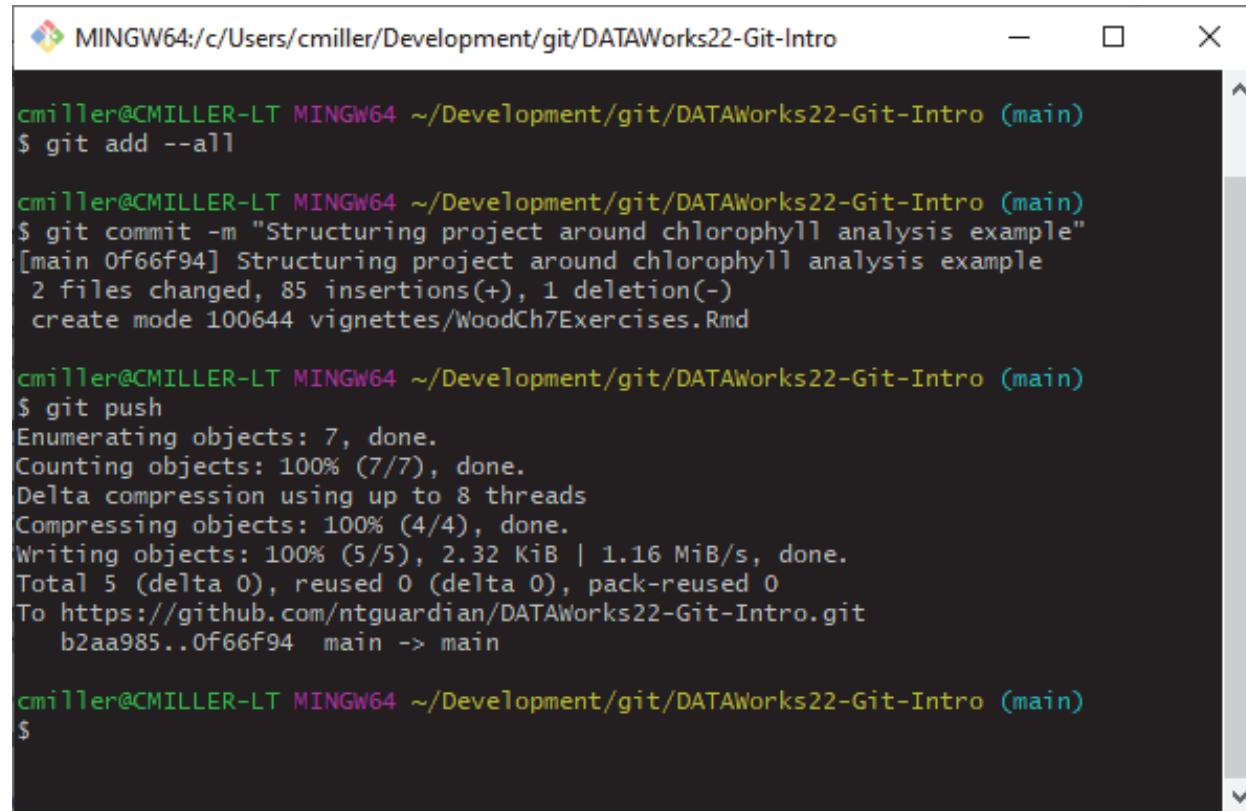
- README.md**:
This file contains a basic Git tutorial. It includes sections on creating a repo, adding files, pushing/pulling, branching, merging, and demonstrating a vignette setup.
- WoodCh7Exercises.Rmd**:
This file contains R code for a vignette. It includes a code block for setting up the environment, followed by a large block of text explaining the purpose of the vignette and the relationship between satellite and direct measurements of chlorophyll concentration.

Below the files, the file system navigation bar shows the path: Miller, Curtis G > Development > git > DATAWorks22-Git-Intro > vignettes. A file list on the right shows the contents of the vignettes directory:

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

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-Intro
$ 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
$
```

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

The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, 1 branch, 0 tags, 'Go to file', 'Add file', and 'Code'. Below is a list of files:

- vignettes**: Structuring project around chlorophyll analysis example (0f66f94, 13 minutes ago, 3 commits)
- .gitignore
- LICENSE
- README.md**: Initial commit (Structuring project around chlorophyll analysis example, 13 minutes ago, 3 hours ago)

A yellow box highlights the 'vignettes' file, and a callout notes: "Indicates the most up-to-date state of the project". Arrows point from the 'README.md' commit message to the 'Initial commit' in the file list, and from the 'Initial commit' back to the 'vignettes' file, with a callout stating: "This file has not been altered so nothing has changed". A large yellow box surrounds the entire commit history area, with a callout: "There are new files in the project". Another callout points to the 'README.md' commit message: "Now using the most recent commit message since this file was altered". The main content area below the files contains the following text:

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.

Let's see the history of the project so far

The screenshot shows a GitHub repository interface. At the top, it displays 'main' branch, 1 branch, and 0 tags. Below this is a commit list:

Author	Commit Message	Date	Commits
Curtis Miller	Structuring project around chlorophyll analysis example	13 minutes ago	0f66f94
	Structuring project around chlorophyll analysis example	13 minutes ago	3 commits
vignettes	Initial con		
.gitignore	Initial con		
LICENSE	Initial con		
README.md	Structuring		

A yellow callout box points to the commit hash '0f66f94' with the text: "An identifier of the commit we are looking at; this is the first seven characters of the commit's identifier, known as its ‘hash’". Another yellow callout box points to the '3 commits' link with the text: "Click this to see the history of the project".

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.

Our project has a short history

The screenshot shows a GitHub commit history for the 'main' branch. There are three commits listed:

- Structuring project around chlorophyll analysis example** by Curtis Miller committed 24 minutes ago. It includes a copy icon, a commit hash (0f66f94), and a diff icon.
- Added a README.md file describing the purpose of the project** by Curtis Miller committed 1 hour ago. It includes a copy icon, a commit hash (b2aa985), and a diff icon.
- Initial commit** by ntguardian committed 3 hours ago. It includes a verified badge, a copy icon, a commit hash (3c0de05), and a diff icon.

At the bottom, there are 'Newer' and 'Older' navigation buttons.

From here we can explore that history

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

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

0f66f94

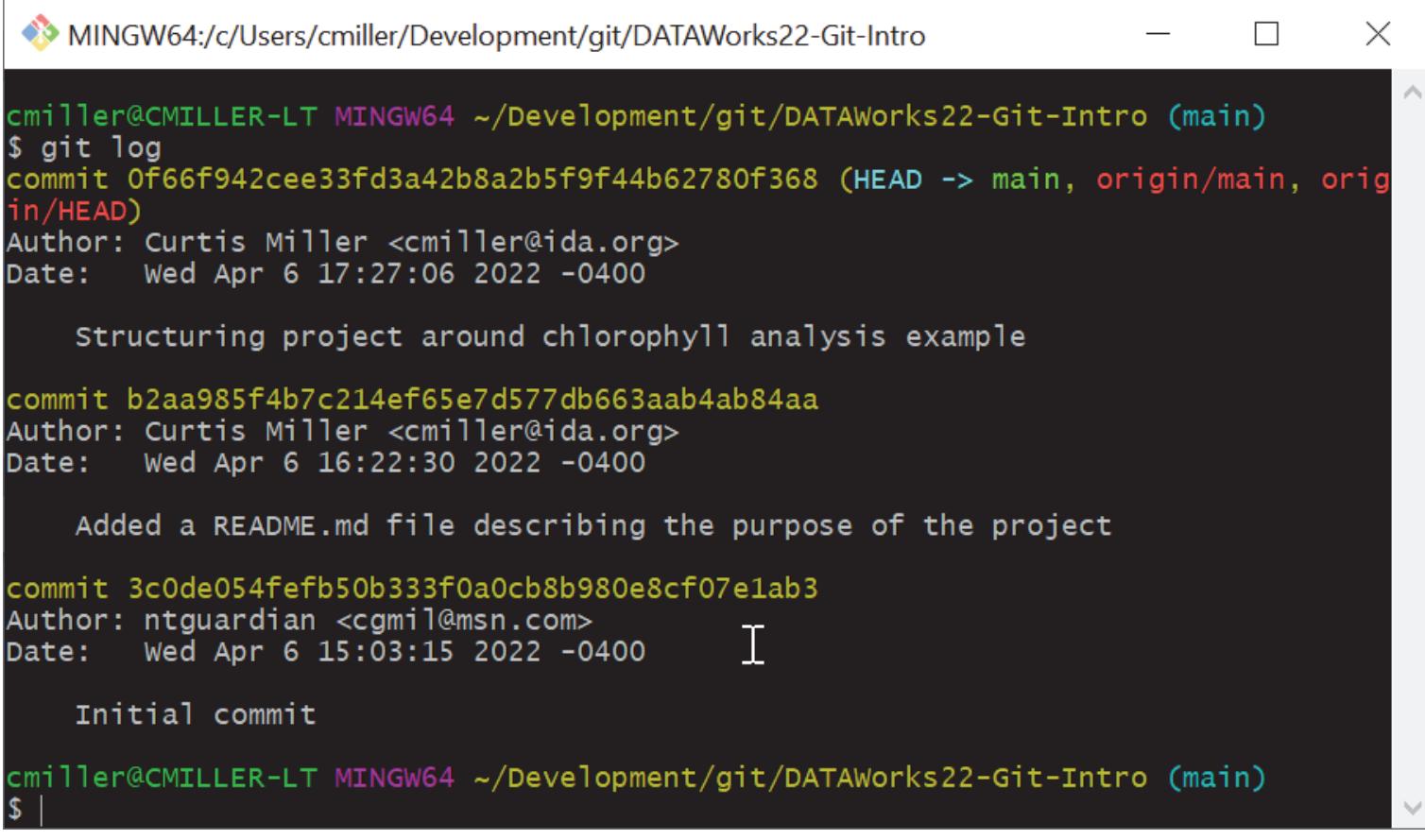
b2aa985

Verified

Newer Older

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



The screenshot shows a terminal window titled "MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro". The window displays the output of a "git log" command. The log shows four commits:

- commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368 (HEAD -> main, origin/main, origin/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

git log is the command for summarizing a repo's history

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

MINGW64:~/Development/git/DATAWorks22-Git-Intro (main)

```
git log
commit 0f66f942cee33fd3a42b8a25f9f44b62780f368 (HEAD -> main, origin/main, origin/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
```

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

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

Commit hash identifier (often abbreviated to first 7 characters)

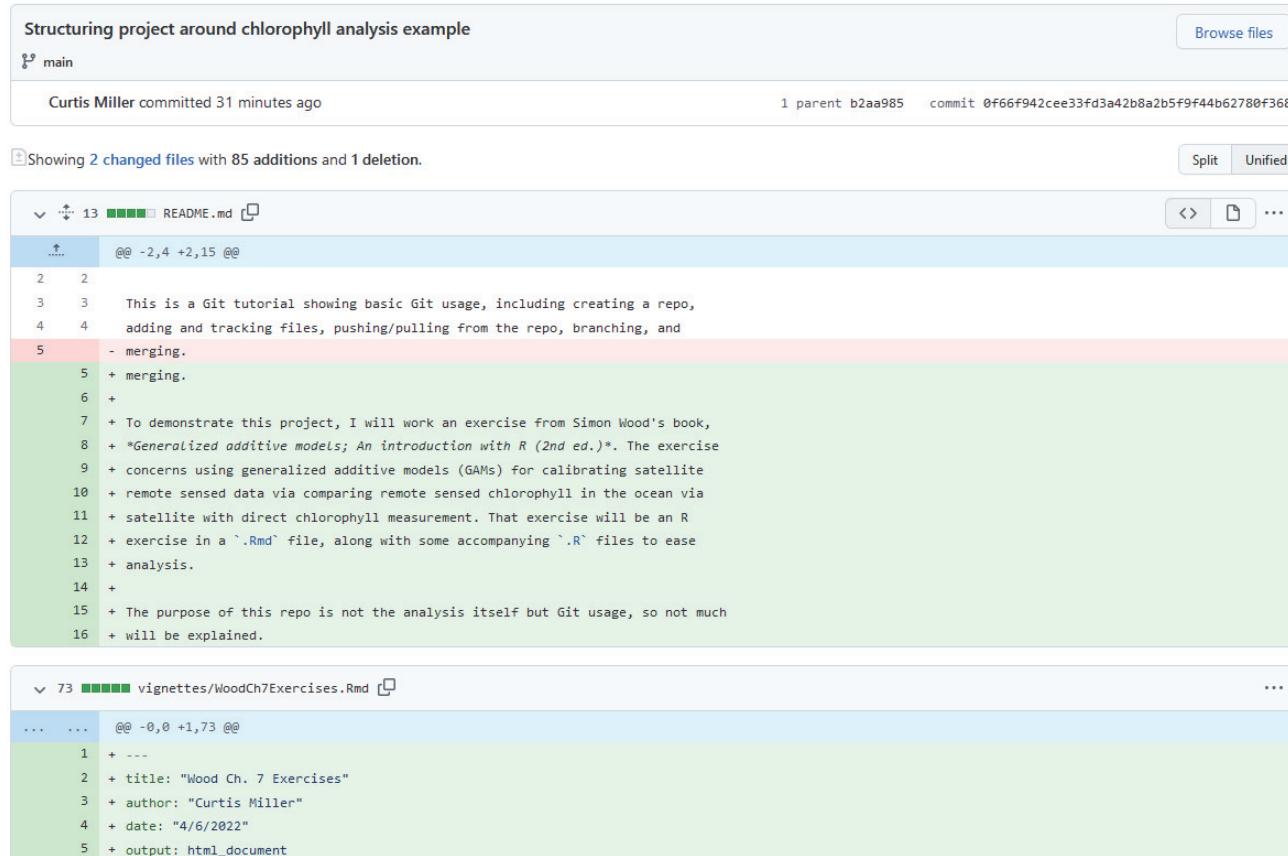
Commit message describing what change was made

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-MTNGW64:~/Development/git/DATAworks22-Git-Intro (main)
$ git log --graph --oneline --decorate --all
* 0f66f94 (HEAD -> main, origin/main, origin/HEAD) Structuring project around chlorophyll 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*



The screenshot shows a GitHub commit interface. At the top, it says "Structuring project around chlorophyll analysis example" and "main". It was committed by "Curtis Miller" 31 minutes ago, with 1 parent commit (0f66f942cee33fd3a42b8a2b5f9f44b62780f368). Below this, it says "Showing 2 changed files with 85 additions and 1 deletion." The first file shown is "README.md", which has 13 changes. The diff shows additions (green) and deletions (red). The second file shown is "vignettes/WoodCh7Exercises.Rmd", which has 73 changes. The diff shows additions (green) and deletions (red).

```
diff --git a/README.md b/README.md
@@ -2,4 +2,15 @@
 2 2
 3 3 This is a Git tutorial showing basic Git usage, including creating a repo,
 4 4 adding and tracking files, pushing/pulling from the repo, branching, and
-5 - merging.
+5 + merging.
 6 +
 7 + To demonstrate this project, I will work an exercise from Simon Wood's book,
 8 + *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
 9 + concerns using generalized additive models (GAMs) for calibrating satellite
10 + remote sensed data via comparing remote sensed chlorophyll in the ocean via
11 + satellite with direct chlorophyll measurement. That exercise will be an R
12 + exercise in a `.Rmd` file, along with some accompanying `*.R` files to ease
13 + analysis.
14 +
15 + The purpose of this repo is not the analysis itself but Git usage, so not much
16 + will be explained.
```

```
diff --git a/vignettes/WoodCh7Exercises.Rmd b/vignettes/WoodCh7Exercises.Rmd
@@ -0,0 +1,73 @@
 1 + ---
 2 + title: "Wood Ch. 7 Exercises"
 3 + author: "Curtis Miller"
 4 + date: "4/6/2022"
 5 + 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)

The screenshot shows a Git commit interface with the following details:

- Commit Details:** Commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368, 1 parent b2aa985.
- File Changes:** 2 changed files with 85 additions and 1 deletion.
- README.md Changes:** Lines 3 and 5 are highlighted in red, indicating deletion. Lines 3 and 5 are also highlighted in green, indicating addition. The commit message describes the purpose of the project and its analysis.
- vignettes/WoodCh7Exercises.Rmd Changes:** The entire file is shown in green, indicating it is a whole new file. The commit message specifies the title, author, date, and output type.

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

```
rmillerr@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

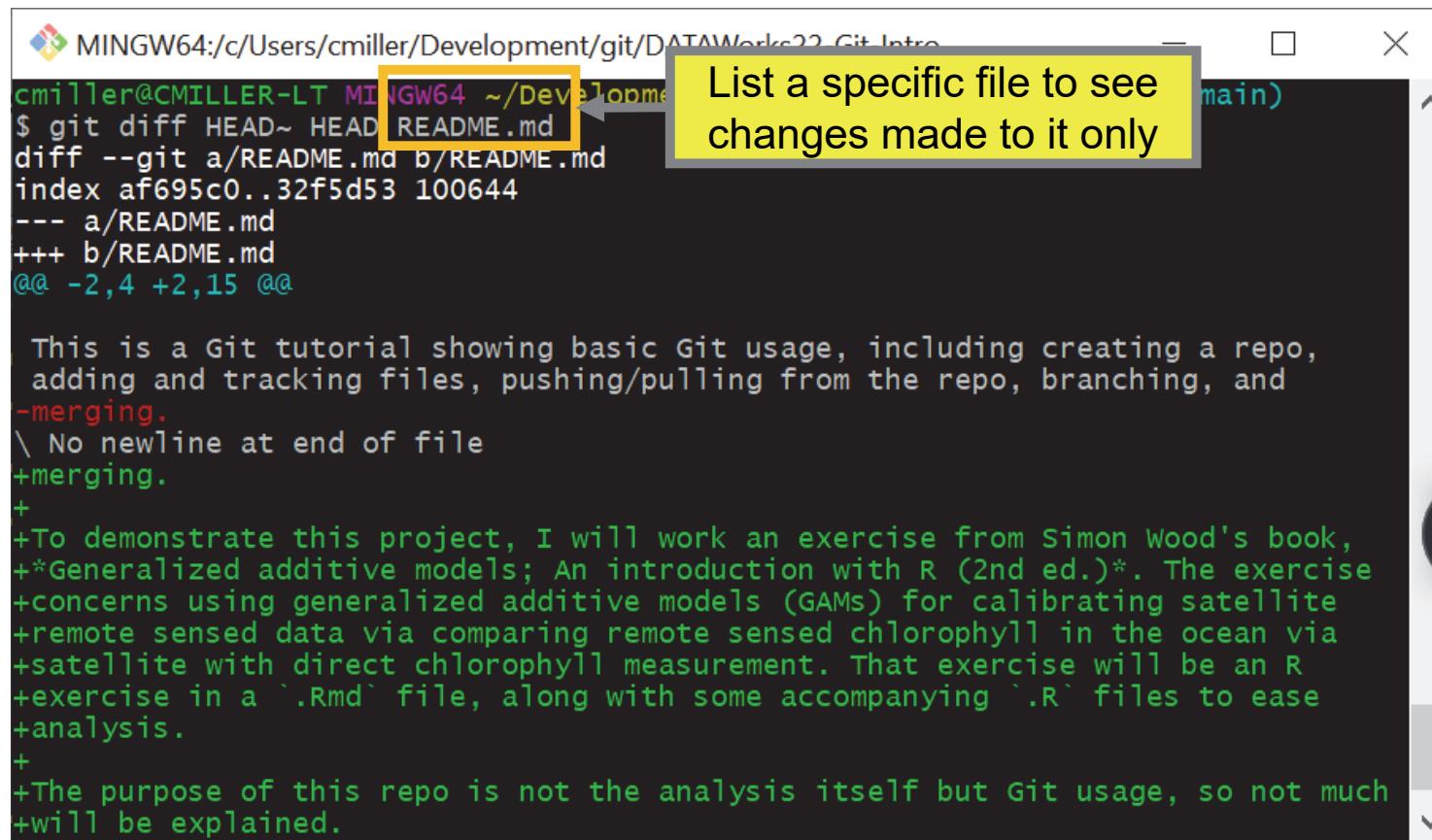
Annotations in the screenshot:

- Red: what was removed**: Points to the deleted text in the diff output.
- Green: what was added**: Points to the inserted text in the diff output.
- Location in file**: Points to the line numbers in the diff output.
- What is being compared**: Points to the file paths in the diff output.

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
@@ -2,4 +2,15 @@
This is a Git tutorial showing how to use Git to add, commit, and track files. It also covers 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 exercise
+concerns using generalized additive models (GAMs) for calibrating satellite
+remote sensed data via comparing remote sensed chlorophyll in the ocean
+with satellite with direct chlorophyll measurement. That exercise will be an
+exercise in a `Rmd` file, along with some accompanying `R` files to support
+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
::
```

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



The screenshot shows a terminal window titled "MINGW64:/c/Users/cmiller/Development/git/DATAWorks22_Git Intro". The command \$ git diff HEAD~ HEAD README.md is run, resulting in the following output:

```
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.
```

A yellow callout box points to the command \$ git diff HEAD~ HEAD README.md with the text "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

The screenshot shows a GitHub repository interface. At the top, there are navigation links: 'b2aa985f4b' (with a dropdown arrow), '1 branch' (with a dropdown arrow), '0 tags' (with a dropdown arrow), 'Go to file' (button), and 'Code' (button). Below this, a commit history table is displayed:

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

Below the commit history, there is a section for the 'README.md' file, which contains the following text:

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

The screenshot shows a GitHub commit history for a repository. At the top left, a dropdown menu is highlighted with a yellow box, showing the commit hash `b2aa985f4b`. A yellow callout box points to this area with the text "This describes which commit we're looking at". To the right of the dropdown, there are buttons for "Go to file" and "Code". Below the dropdown, it says "1 branch" and "0 tags". The commit history lists four commits:

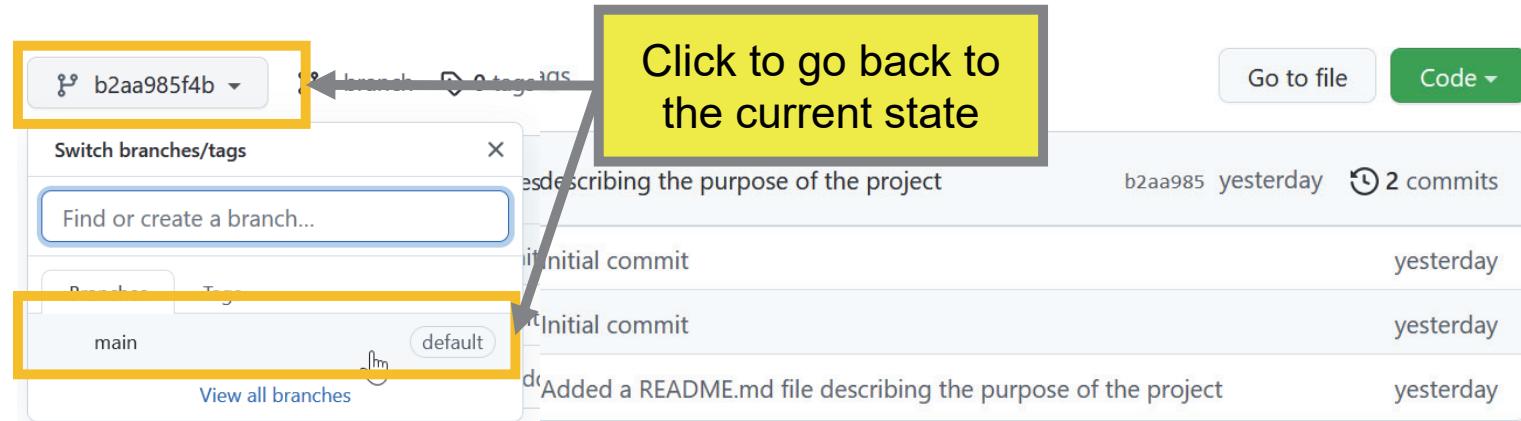
Author	Message	Date	Commits
Curtis Miller	Added a README.md file describing the purpose of the project	yesterday	2 commits
	.gitignore	Initial commit	yesterday
	LICENSE	Initial commit	yesterday
	README.md	Added a README.md file describing the purpose of the project	yesterday

Below the commit history, there is a file viewer for the `README.md` file, showing its content and a edit icon.

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

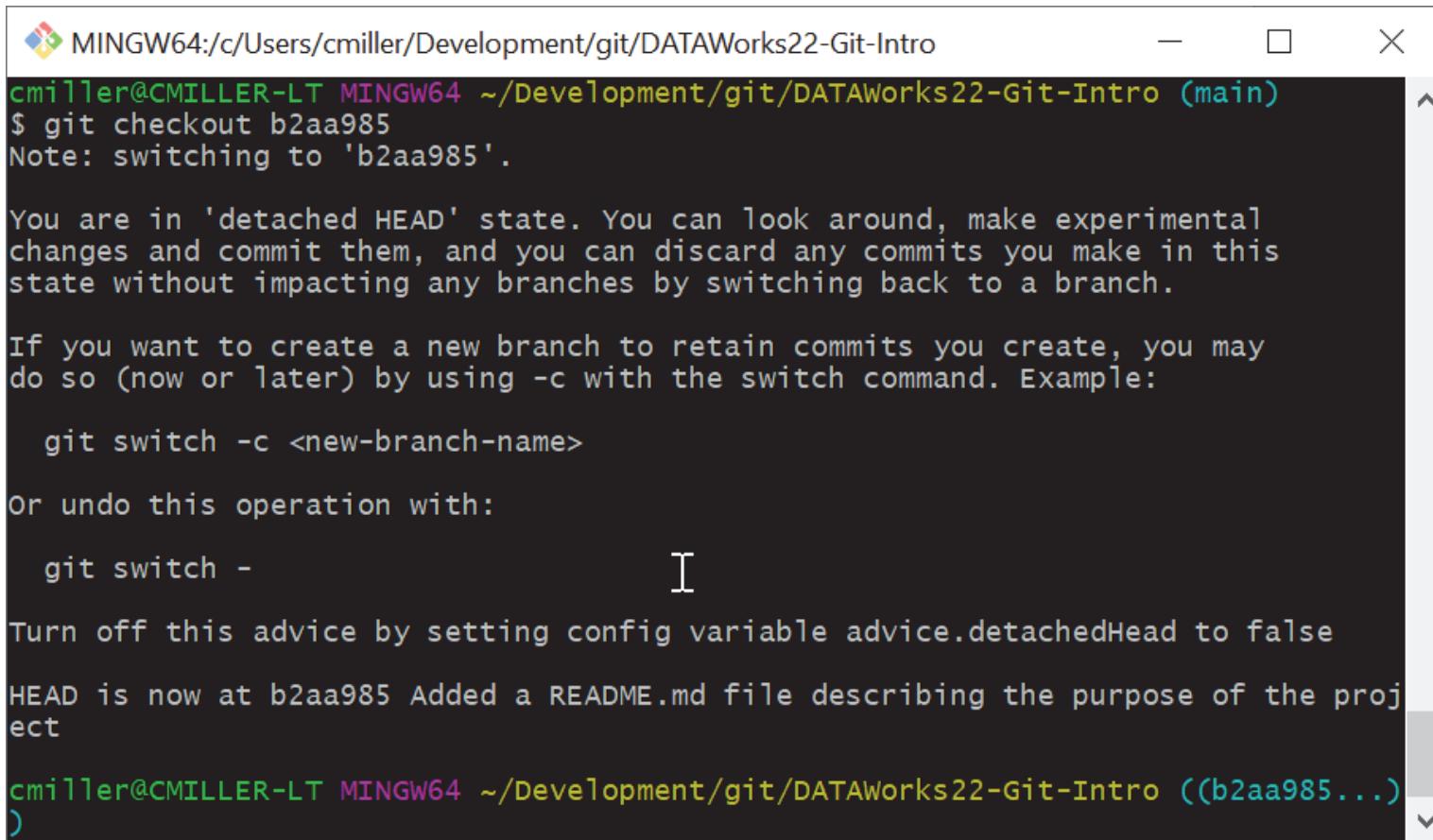


README.md

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

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

The terminal window shows the command `git checkout b2aa985` being run. A yellow box highlights the commit hash `b2aa985`. A callout bubble points to this box with the text "The commit we are checking out". The terminal output indicates switching to a detached HEAD state, which is explained in a callout bubble: "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)". Another callout bubble on the right says "Notice how this changed...". The terminal also shows the command `git switch -c` and the message "HEAD is now at b2aa985 Added a README.md file describing the purpose of ect".

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro
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 operation with
git switch -
Turn off this advice with
git config --global advice.detachedHead false
HEAD is now at b2aa985 Added a README.md file describing the purpose of ect
```

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...

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

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

A screenshot of a file browser interface. On the left, a sidebar shows a list of files: .git, .gitignore, LICENSE, and README.md. The .git folder is highlighted with a yellow box. A yellow callout box points to it with the text: "Tracked files not in the old commit have been deleted". In the center, a list of commits is shown:

- 2022-04-07 17:13 File folder
- 2022-04-06 15:27 Text Document
- 2022-04-06 15:07

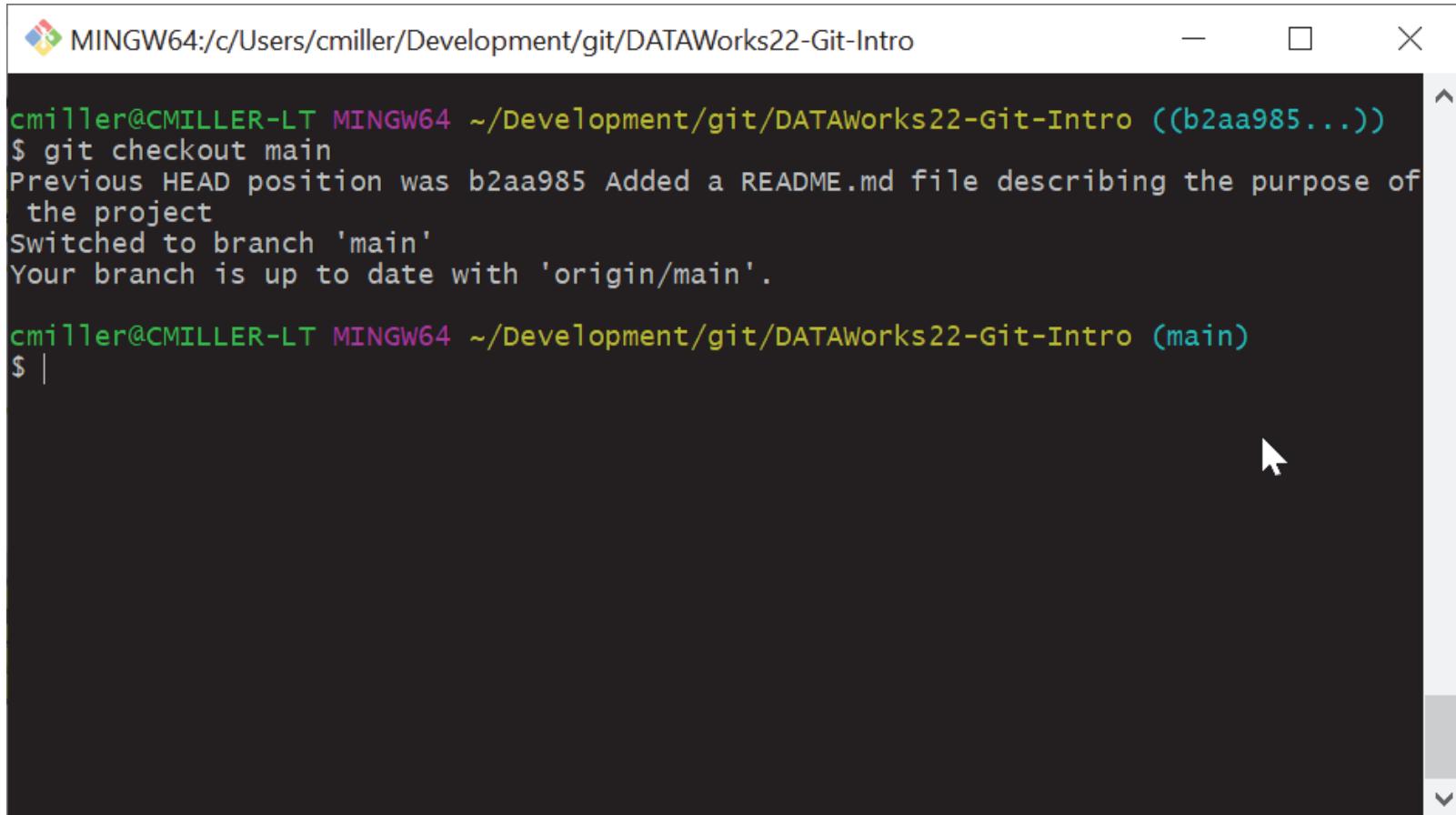
An arrow points from the second commit to a GitHub commit page for "README.md". The commit message reads:

```
1 # DATAworks 2022 Introduction to Git
2
3 This is a Git tutorial showing basic Git usage, including creating a repo,
4 adding and tracking files, pushing/pulling from the repo, branching, and
5 merging.
```

A yellow callout box above the commit message says: "Files have been changed to their old state".

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?”



A screenshot of a terminal window titled "MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro". The window shows the following command and its output:

```
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'.
```

The terminal prompt ends with a dollar sign and a vertical bar, indicating where the user can type their next command.

You check out the main branch

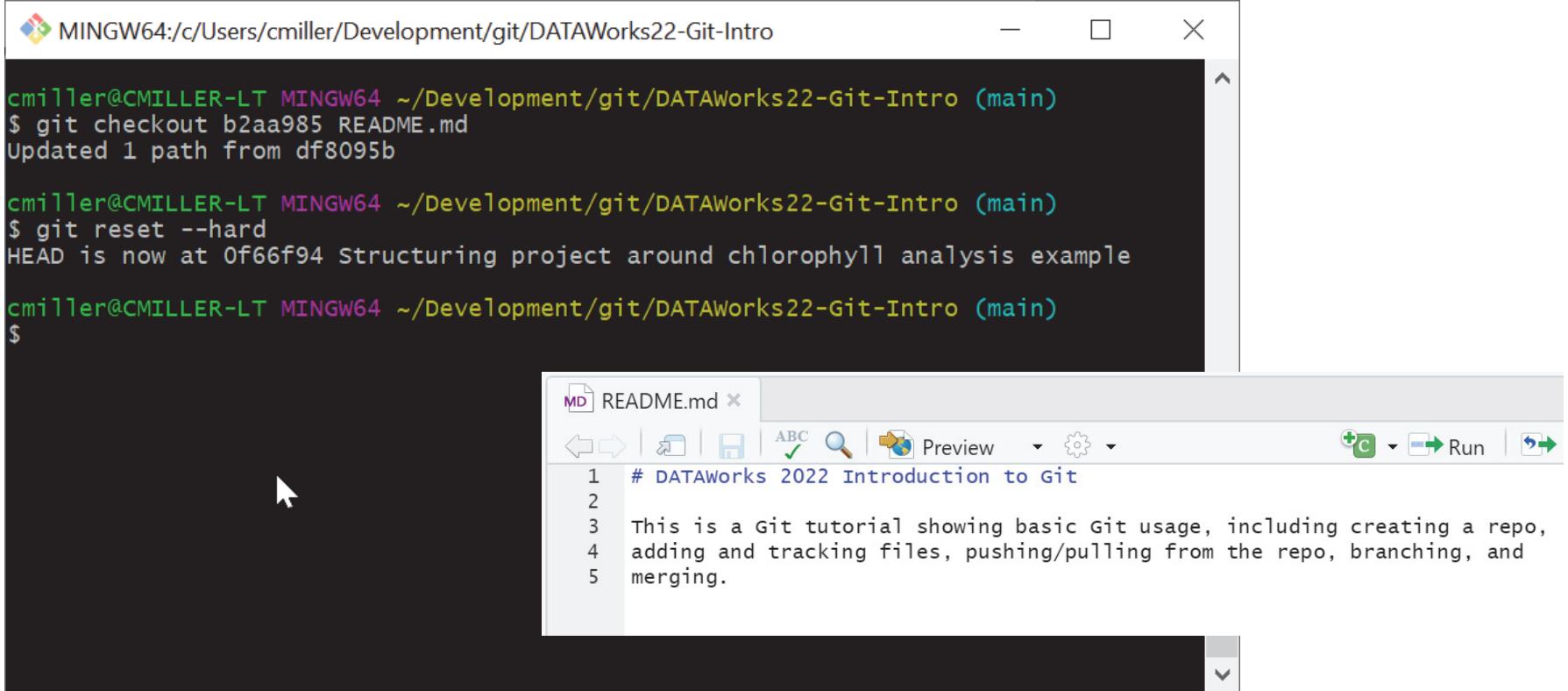
The screenshot shows a terminal window with the following text:

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro
$ git checkout main
Previous HEAD position was b2aa...
the project
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Annotations with yellow boxes and arrows highlight specific parts of the terminal output:

- A yellow box surrounds the command `$ git checkout main`. An arrow points from this box to a yellow box containing the text: "Check out a branch (our repo only has one, the main branch)".
- A yellow box surrounds the commit hash `((b2aa985...))`. An arrow points from this box to a yellow box containing the text: "e describing the purpose of".
- A yellow box surrounds the text `(main)`. An arrow points from this box to a yellow box containing the text: "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.”



The image shows a terminal window and a code editor window side-by-side.

Terminal Window (MINGW64):

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

Code Editor Window (README.md):

```
MD README.md x
← → ↻ ↻ ABC 🔎 Preview ⚙️ +C Run ⏪
1 # DATAWorks 2022 Introduction to Git
2
3 This is a Git tutorial showing basic Git usage, including creating a repo,
4 adding and tracking files, pushing/pulling from the repo, branching, and
5 merging.
```

Just say what file you want

The screenshot shows a terminal window and a code editor side-by-side.

Terminal Output:

```
MINGW64:/c/Users/cmiller/Development/git/DATAworks22-Git-Intro
cmiller@CMILLER-LT MINGW64 ~/Development/
$ git checkout b2aa985 README.md
Updated 1 path from df8095b

cmiller@CMILLER-LT MINGW64 ~/Development/ DATAworks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f6af94 Structuring project analysis example

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (main)
$
```

Annotations in yellow boxes:

- "Name the file you want" points to the command `git checkout b2aa985 README.md`.
- "Up next..." points to the command `git reset --hard`.

Code Editor:

A code editor window titled "README.md" is shown, displaying the following content:

```
# 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 Document
LICENSE	2022-04-06 15:27	File

2

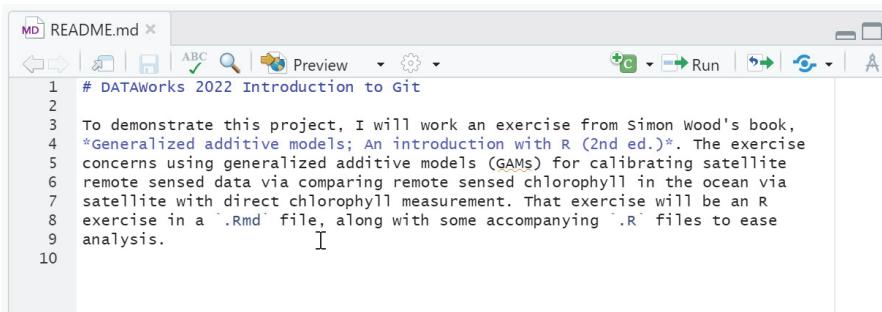
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring project around chlorophyll analysis example
```

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 Document
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



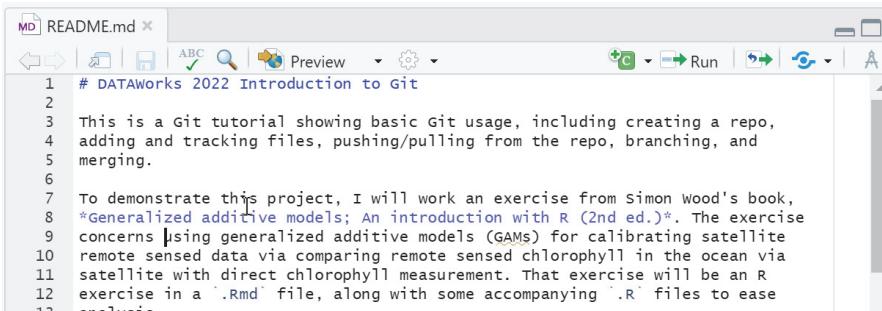
The screenshot shows a code editor window titled "README.md". The file contains the following text:

```
1 # DATAworks 2022 Introduction to Git
2
3 To demonstrate this project, I will work an exercise from simon wood's book,
4 *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
5 concerns using generalized additive models (GAMs) for calibrating satellite
6 remote sensed data via comparing remote sensed chlorophyll in the ocean via
7 satellite with direct chlorophyll measurement. That exercise will be an R
8 exercise in a '.Rmd' file, along with some accompanying '.R' files to ease
9 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



The screenshot shows a code editor window titled "README.md". The file contains the following text, identical to the first screenshot but with a few changes highlighted in red:

```
1 # DATAworks 2022 Introduction to Git
2
3 This is a Git tutorial showing basic Git usage, including creating a repo,
4 adding and tracking files, pushing/pulling from the repo, branching, and
5 merging.
6
7 To demonstrate this project, I will work an exercise from simon wood's book,
8 *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
9 concerns using generalized additive models (GAMs) for calibrating satellite
10 remote sensed data via comparing remote sensed chlorophyll in the ocean via
11 satellite with direct chlorophyll measurement. That exercise will be an R
12 exercise in a '.Rmd' file, along with some accompanying '.R' files to ease
13 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!**

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

2

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

```
215 # Function implemented in Rcpp for speed
216 # It accepts the matrix y and the evaluation of the kernel function stored
217 # in kern_vals, and returns a vector containing estimated long-run variances
218 # at points t
219 sigma <- get_lrv_vec_cpp(y, kern_vals, max_l)
220
221 # "Equivalent" R code (slower)
222 # covs <- sapply(1:n, function(t) {
223 #   sapply(0:(n - 1), function(l) {
224 #     mean(y[1:(n - 1),t]*y[(l + 1):n,t])
225 #   })
226 # })
227 #
228 # sigma <- sapply(2:(n - 2), function(t) {
229 #   covs[0 + 1, t] + 2 * sum(vkernel(1:(n - 1)/h) * covs[1:(n - 1), t])
230 # })
231
232 if (any(sigma < 0)) {
233   warning(paste("A negative variance was computed! This may be due to a bad",
234                 "kernel being chosen."))
235 }
236
237 sigma
```

Using filenames to keep a history...

Commenting out/keeping old useless code...

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

The screenshot shows an email inbox interface with the following details:

- Header:** New report draft, Current Mailbox
- Filter:** All Unread By Date ↑
- 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 [EXT] New report draft

Code Snippet (Right side):

```
emented in Rcpp for speed
e 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)

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

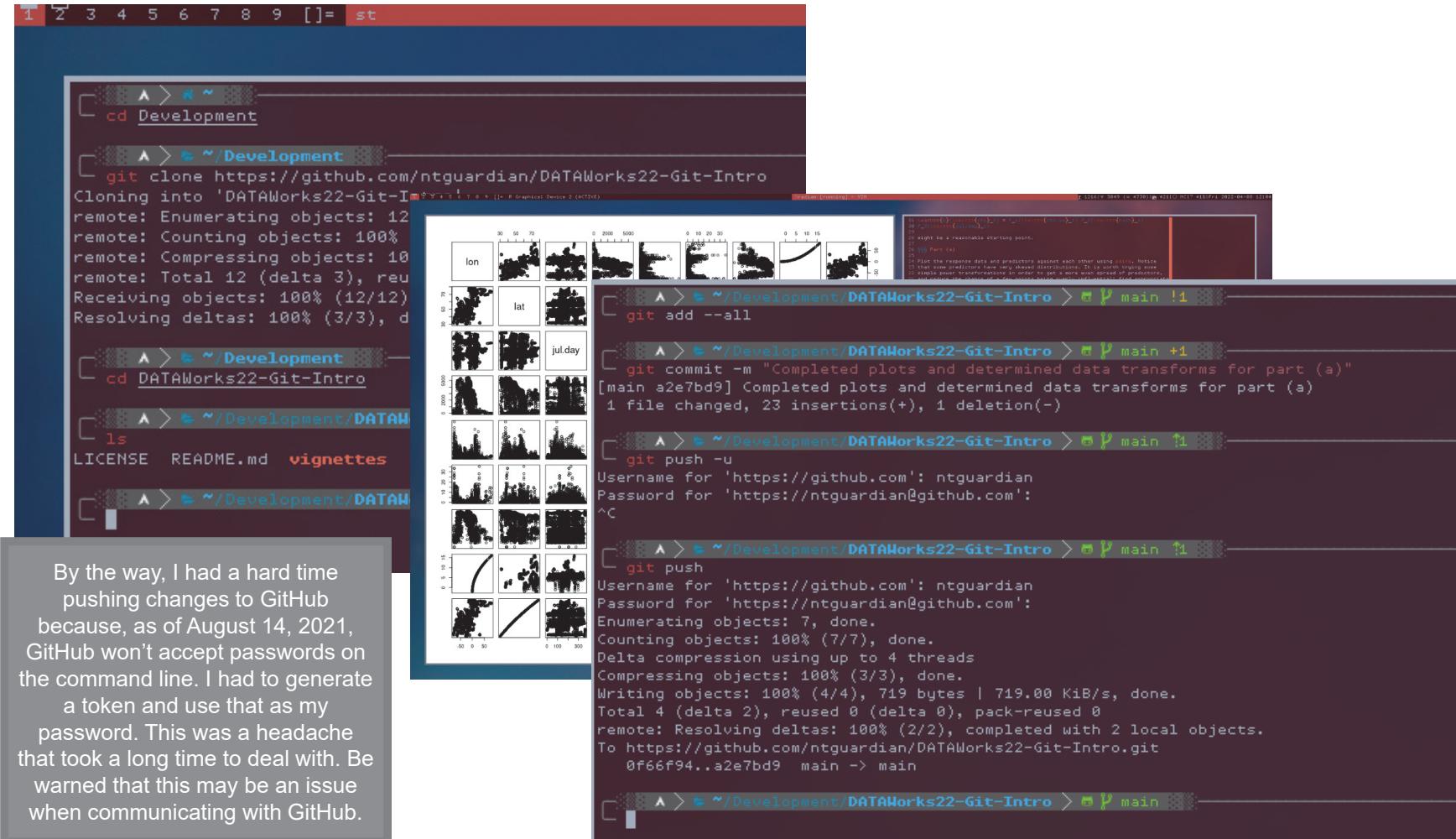
0)) {
("A negative variance was computed! This may be due to a bad",
"kernel being chosen."))
```

Using filenames to keep a history...

Using e-mail to manage collaboration/records...

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

A > ~ Development
cd Development

A > ~/Development
git clone https://github.com/ntguardian/DATAWorks22-Git-Intro
Cloning into 'DATAWorks22-Git-Intro'...
remote: Enumerating objects: 12
remote: Counting objects: 100%
remote: Compressing objects: 100%
remote: Total 12 (delta 3), reused 0 from cache
Receiving objects: 100% (12/12)
Resolving deltas: 100% (3/3), done

A > ~/Development
cd DATAWorks22-Git-Intro

A > ~/Development/DATAW
ls
LICENSE README.md vignettes

A > ~/Development/DATAW

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.

A > ~/Development/DATAWorks22-Git-Intro
git add --all
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(-)

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

A > ~/Development/DATAWorks22-Git-Intro
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
```

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

The screenshot shows a GitHub repository interface. At the top, there are navigation buttons: 'main' (with a dropdown arrow), '1 branch' (with a dropdown arrow), '0 tags', 'Go to file', 'Add file', and a green 'Code' button with a dropdown arrow. Below this is a table of recent commits:

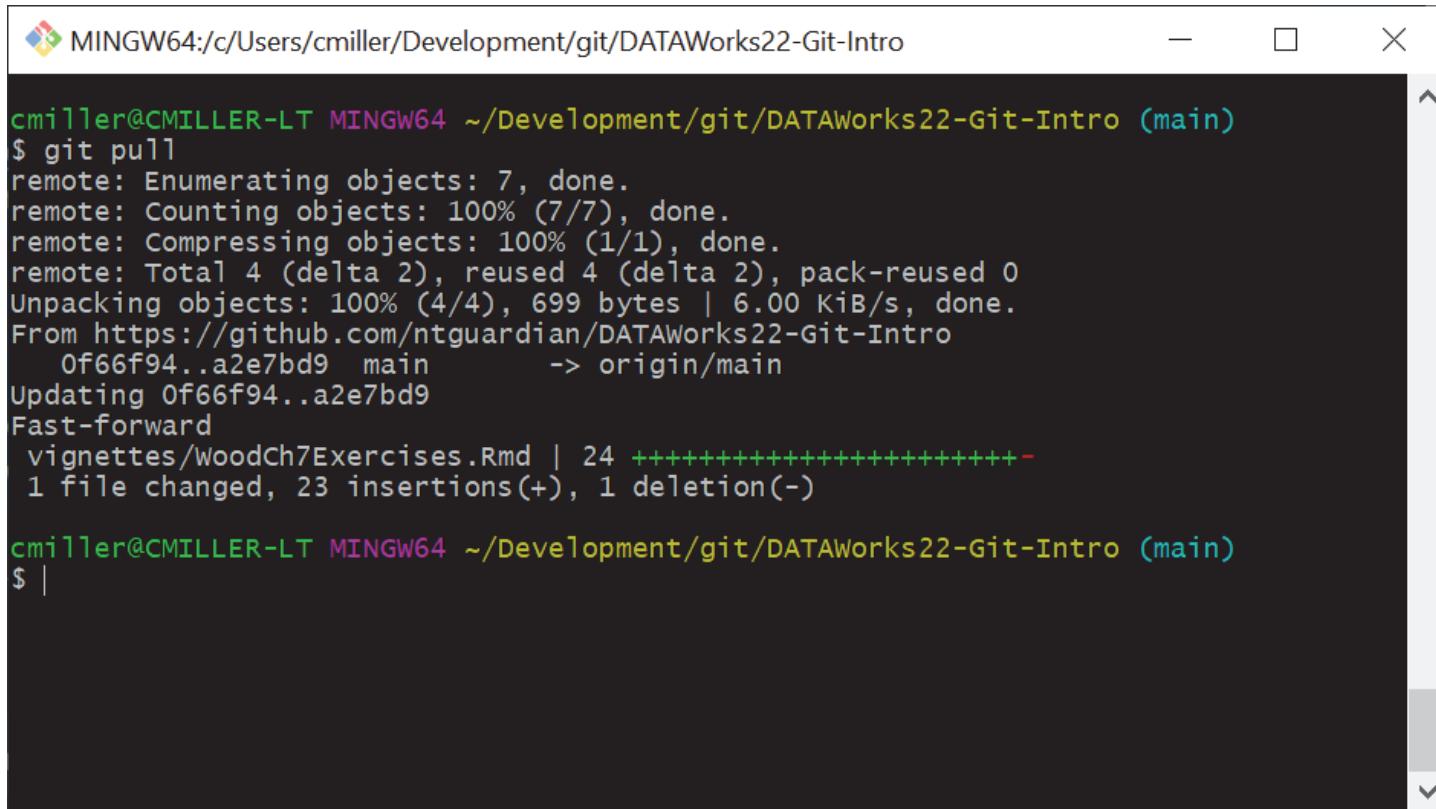
Author	Commit Message	Date	Commits
ntguardian	Completed plots and determined data transforms for part (a)	a2e7bd9 4 hours ago	4 commits
vignettes	Completed plots and determined data transforms for part (a)	4 hours ago	
.gitignore	Initial commit	2 days ago	
LICENSE	Initial commit	2 days ago	
README.md	Structuring project around chlorophyll analysis example	2 days ago	

Below the commits is a preview of the 'README.md' file content:

```
DATAWorks 2022 Introduction to Git
```

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

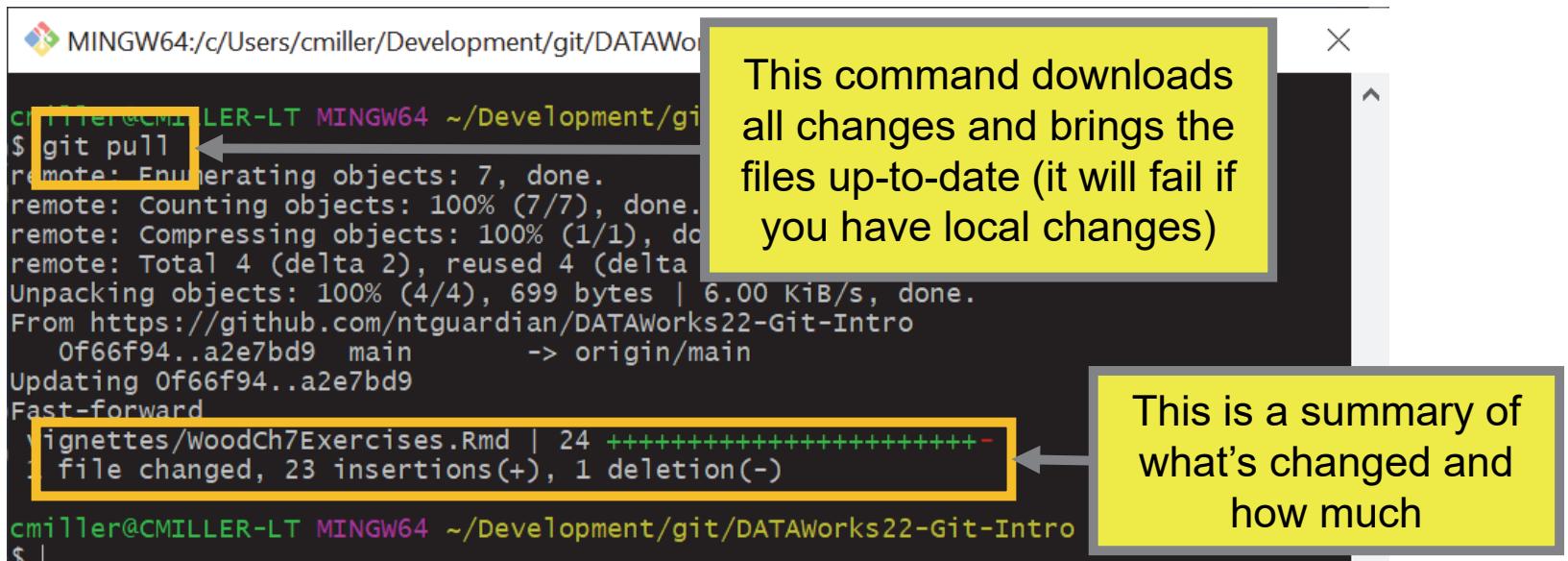
Use git pull to bring the local repo on your hard drive up-to-date



```
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



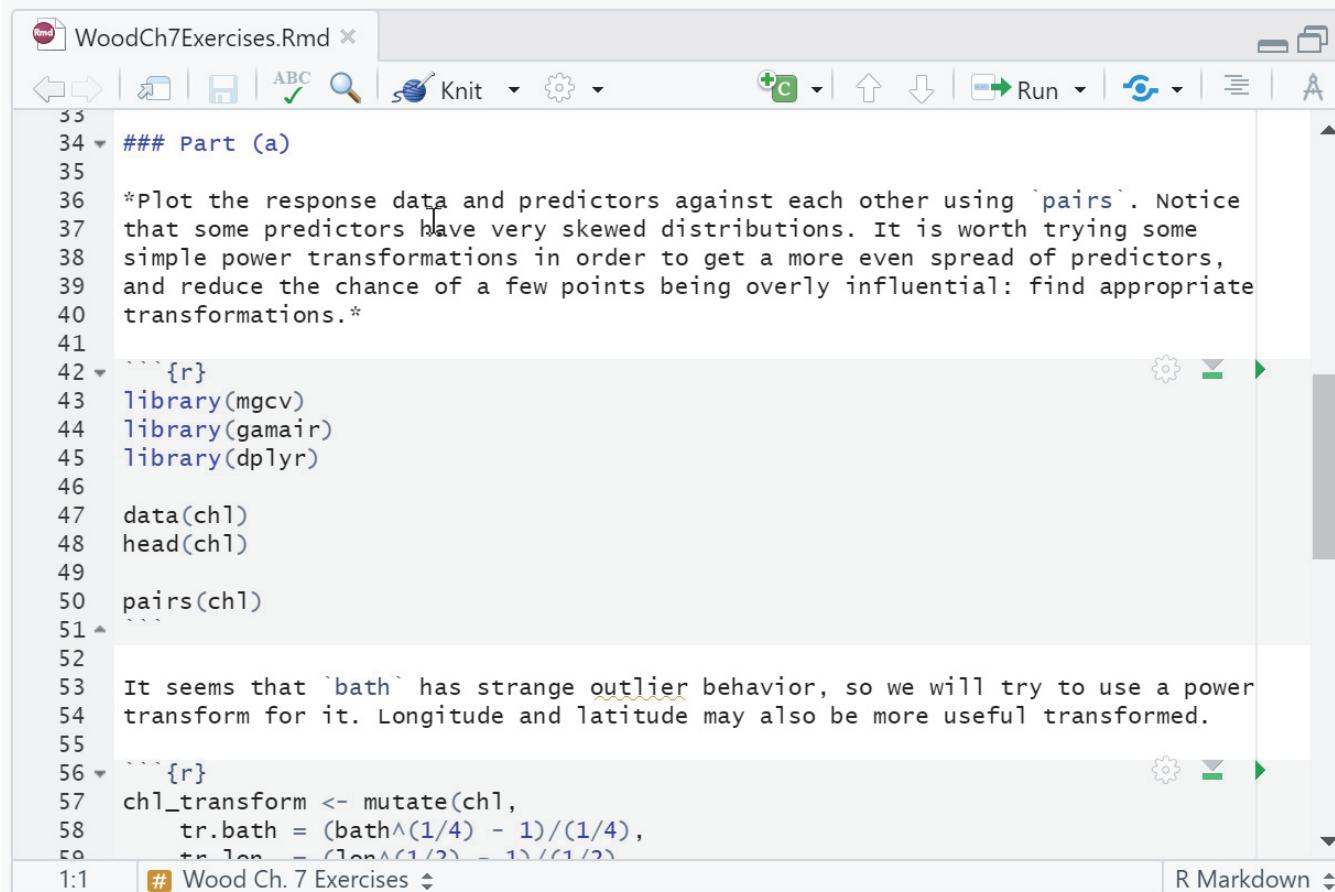
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 2), reused 4 (delta 2)
Unpacking objects: 100% (4/4), 699 bytes | 6.00 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



The screenshot shows the RStudio interface with the following code in the 'WoodCh7Exercises.Rmd' file:

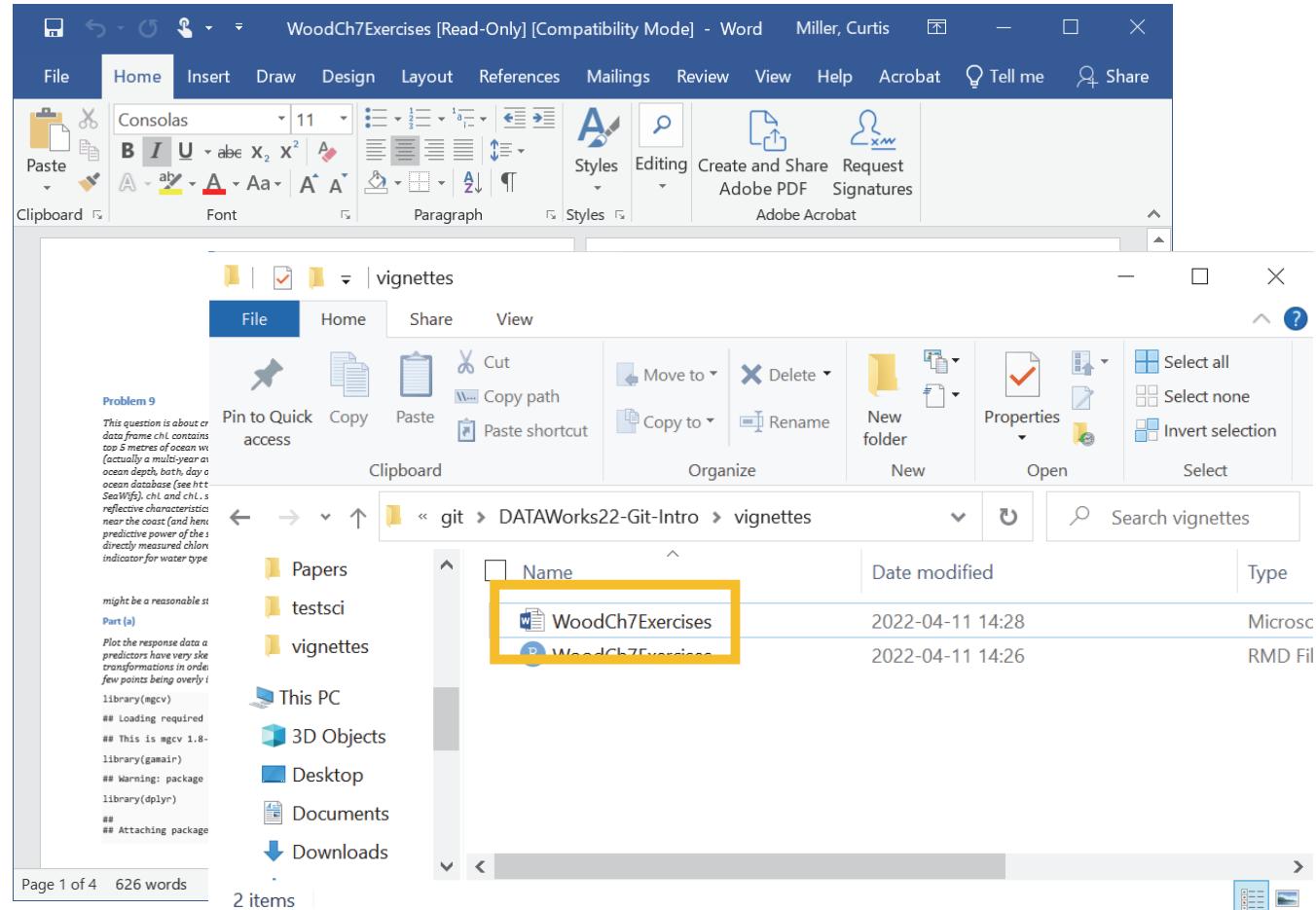
```
33
34 ### Part (a)
35
36 *Plot the response data and predictors against each other using `pairs`. Notice
37 that some predictors have very skewed distributions. It is worth trying some
38 simple power transformations in order to get a more even spread of predictors,
39 and reduce the chance of a few points being overly influential: find appropriate
40 transformations.*
41
42 ```{r}
43 library(mgcv)
44 library(gamair)
45 library(dplyr)
46
47 data(ch1)
48 head(ch1)
49
50 pairs(ch1)
51
52
53 It seems that `bath` has strange outlier behavior, so we will try to use a power
54 transform for it. Longitude and Latitude may also be more useful transformed.
55
56 ```{r}
57 chl_transform <- mutate(ch1,
58       tr.bath = (bath^(1/4) - 1)/(1/4),
59       tr.lon = (lon^(1/2) - 1)/(1/2)
1:1 # Wood Ch. 7 Exercises
```

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

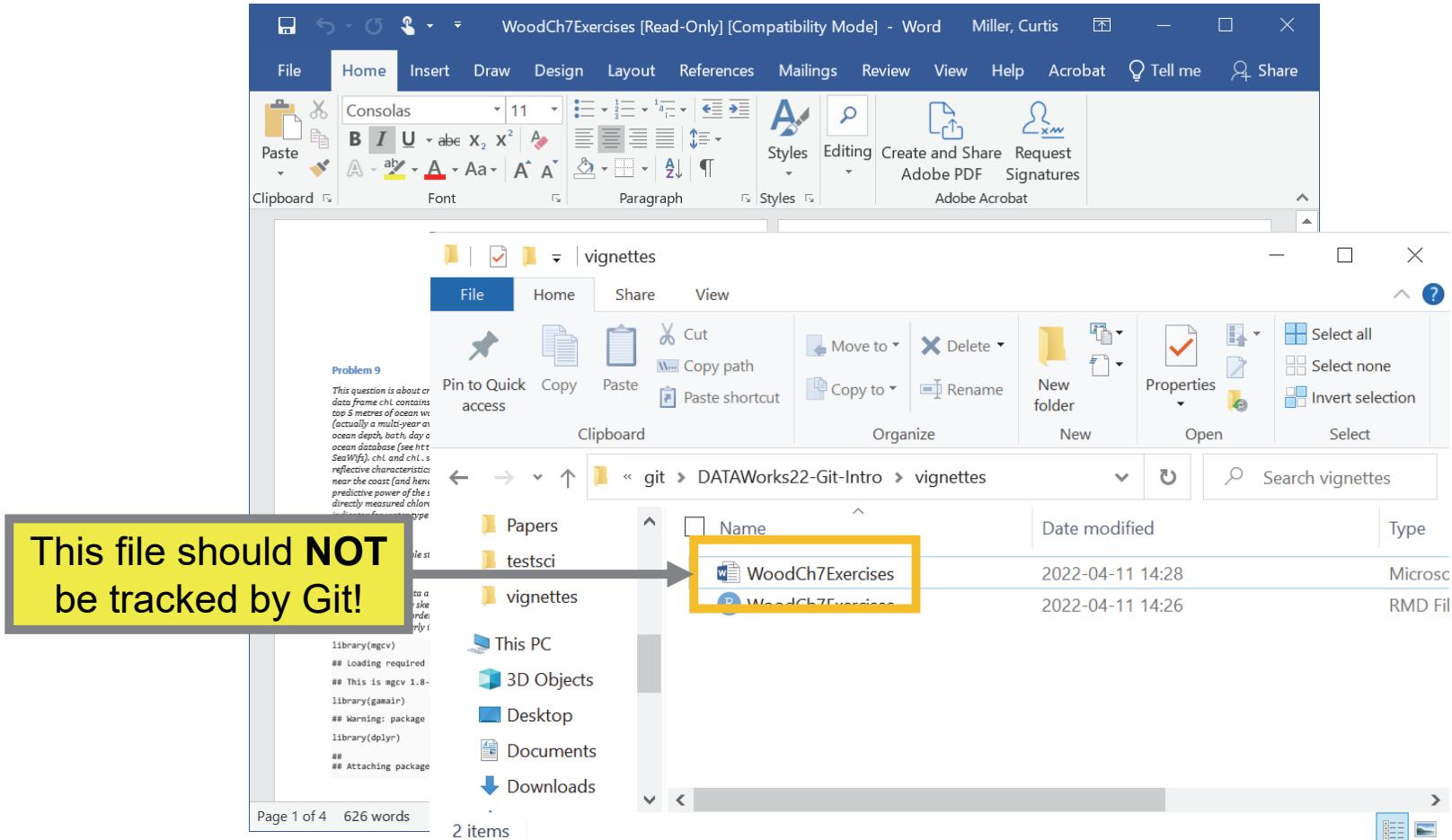
The screenshot shows a Microsoft Word document window titled "WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word". The ribbon menu is visible at the top. The main content area displays an R Markdown document. The document title is "Wood Ch. 7 Exercises" and it is dated "4/6/2022". It contains a section titled "Problem 9" which describes a dataset named "chl" containing chlorophyll measurements. Below this, there is a code block and a plot. The code block starts with `##` and includes several library imports like mgcv, gamair, dplyr, and a warning about gamair being built under R version 4.1.3. The plot is a 4x4 grid of histograms showing distributions of variables: lon, lat, jul.day, bath, chl, chl.sw, R, and S. A note below the plot states: "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." At the bottom of the Word document, it says "Page 1 of 4 626 words".

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...



... but we don't want to track this file



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)



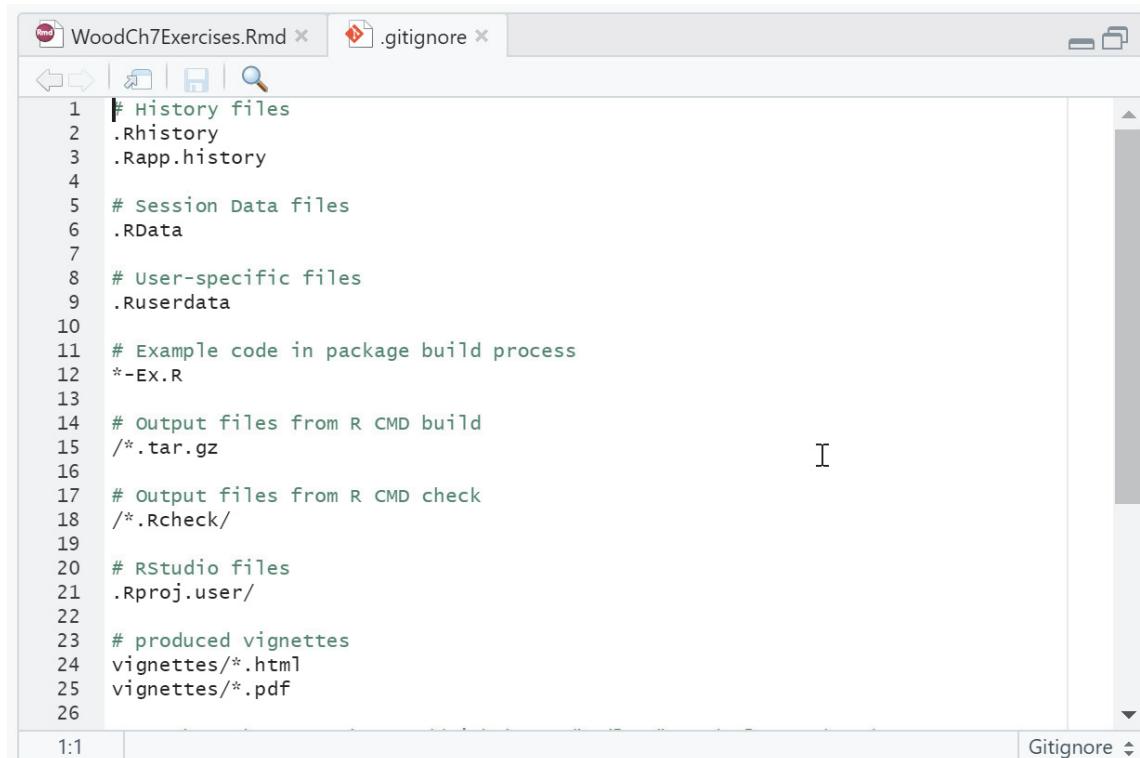
Files generated by other files



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



The screenshot shows a code editor window with two tabs: "WoodCh7Exercises.Rmd" and ".gitignore". The ".gitignore" tab is active, displaying the following content:

```
1 # History files
2 .Rhistory
3 .Rapp.history
4
5 # Session Data files
6 .RData
7
8 # User-specific files
9 .Ruserdata
10
11 # Example code in package build process
12 -*-Ex.R
13
14 # output files from R CMD build
15 /*.tar.gz
16
17 # output files from R CMD check
18 /*.Rcheck/
19
20 # RStudio files
21 .Rproj.user/
22
23 # produced vignettes
24 vignettes/*.html
25 vignettes/*.pdf
26
```

- `.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

The screenshot shows a code editor with two tabs: "WoodCh7Exercises.Rmd" and ".gitignore". The ".gitignore" tab is active, displaying the following content:

```
1 # History files
2 .Rhistory
3 .Rapp.history
4
5 # Session Data files
6 .RData
7
8 # User-specific files
9 .Ruserdata
10
11 # Example code
12 -*-Ex.R
13
14 # Output files from R CMD build
15 /*.tar.gz
16
17 # Output files from R CMD check
18 /*.Rcheck/
19
20 # RStudio files
21 .Rproj.user/
22
23 # produced vignettes
24 vignettes/*.*html
25 vignettes/*.*pdf
```

Annotations explain specific entries:

- "Ignore any file ending in .Rdata anywhere in the repo" points to line 6.
- "Comment; understood by a reader, ignored by Git" points to line 8.
- "Ignore file like "Test-Ex.R" or "my_func_Ex.R" anywhere" points to line 12.
- "In the base directory, ignore files ending in .tar.gz" points to line 15.
- "Ignore any directories with .Rcheck in their name" points to line 18.
- "Ignore all files in this directory" points to line 21.
- "In the vignettes directory, ignore files ending in .html or .pdf" points to line 24.

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

```
--  
23 # produced vignettes  
24 vignettes/*.html  
25 vignettes/*.pdf |  
26 vignettes/*.docx|  
27
```

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-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
```

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
```

The screenshot shows a GitHub commit history for the repository 'DATAWorks22-Git-Intro'. The commit details are as follows:

- Commit message: 'Added knit support for Word files (generated files should be ignored)' by 'Curtis Miller'.
- Commit hash: '0ce384f'.
- Date: '2 minutes ago'.
- File changes: 'WoodCh7Exercises.Rmd'.
- Description: 'Added knit support for Word files (generated files should be ignored)'.
- Last commit hash: 'a2e7bd9..0ce384f'.

... But it did track the other changes

```
cmiller@CMILLER-LT ~
$ git commit -m "Add vignettes"
[main 0ce384f] Added vignettes
 2 files changed, 4 insertions(+)

cmiller@CMILLER-LT ~
$ git push
```

```
(main)
+ vignettes/*.docx
should be ignored
should be ignored)
```

The screenshot shows a GitHub repository interface. On the left, a sidebar displays a commit history:

- Curtis Miller Added knit support for WordCh7Exercises.Rmd
- WoodCh7Exercises.Rmd

The main area shows two file differences:

.gitignore

```
@@ -23,6 +23,7 @@
23 23 # produced vignettes
24 24 vignettes/*.html
25 25 vignettes/*.pdf
26 + vignettes/*.docx
27
28 # OAuth2 token, see https://github.com/hadley/httr/releases/tag/v0.3
29 .httr-oauth
```

vignettes/WoodCh7Exercises.Rmd

```
@@ -2,7 +2,9 @@
2 2 title: "Wood Ch. 7 Exercises"
3 3 author: "Curtis Miller"
4 4 date: "4/6/2022"
5 - output: html_document
5 + output:
6 +   word_document: default
7 +   html_document: default
8 8 ---
9
10 10 `r setup, include=FALSE`
```

On the right, a commit summary is shown:

0ce384f 2 minutes ago History

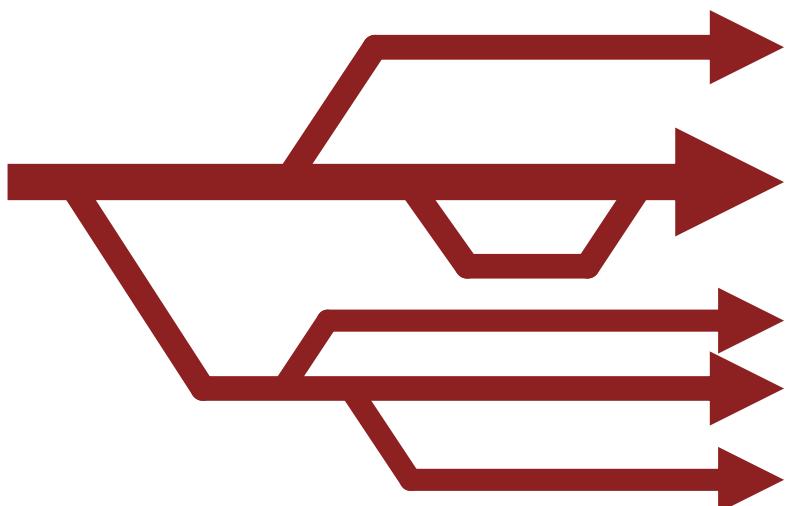
2 minutes ago

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

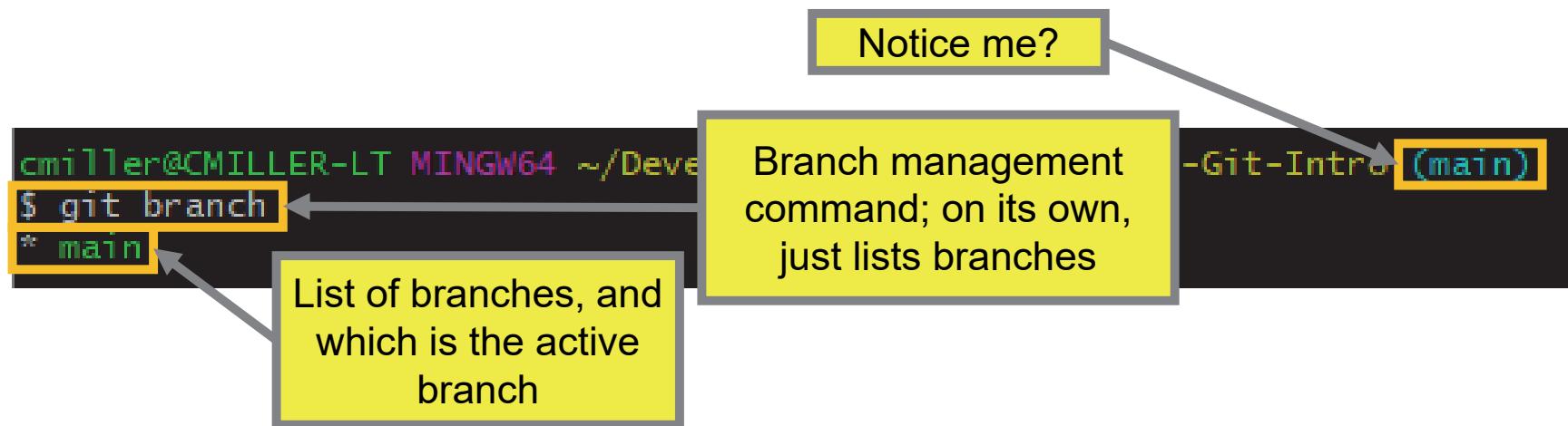


- Commits represent moving forward and backward in time while branches allow for differences between files to exist simultaneously
- Git can track the difference between branches and propagate a change that affects both branches while maintaining what makes them distinct
- Changes made in one branch 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



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'
```

Notice me?

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

```
$ git branch
```

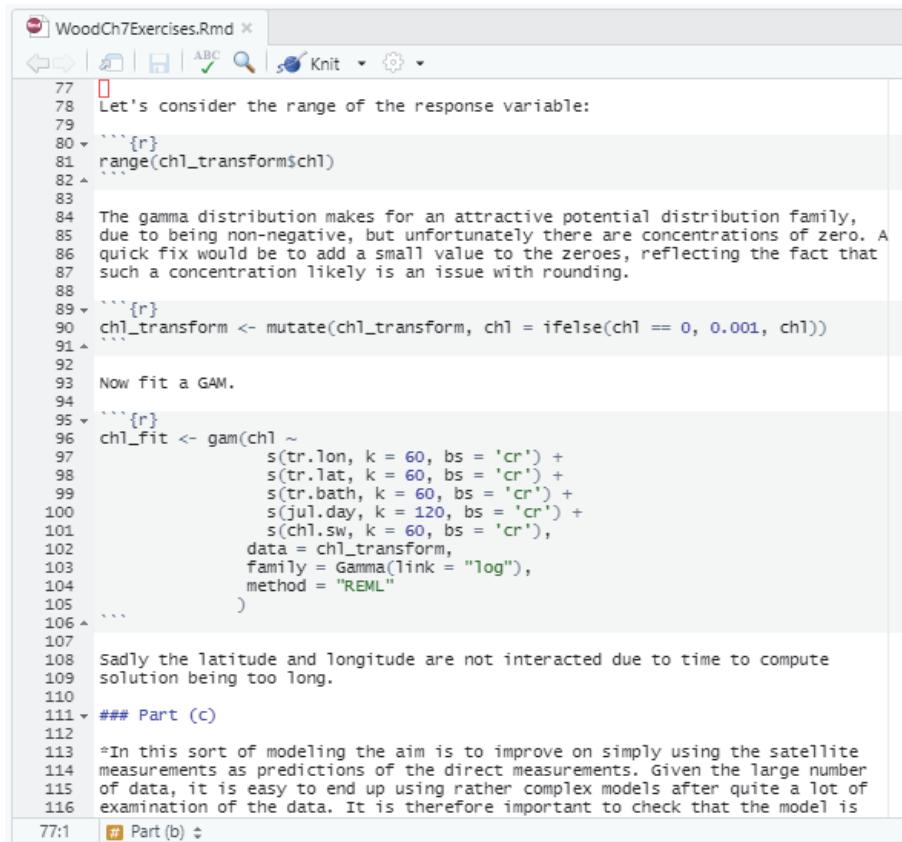
```
* cmiller
  main
```

List of branches; cmiller
is the active branch

(cmiller)

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



```
WoodCh7Exercises.Rmd x
Knit ▾

77 Let's consider the range of the response variable:
78
79
80 range(chl_transform$chl)
81
82
83
84 The gamma distribution makes for an attractive potential distribution family,
85 due to being non-negative, but unfortunately there are concentrations of zero. A
86 quick fix would be to add a small value to the zeroes, reflecting the fact that
87 such a concentration likely is an issue with rounding.
88
89 chl_transform <- mutate(chl_transform, chl = ifelse(chl == 0, 0.001, chl))
90
91
92 Now fit a GAM.
93
94
95 chl_fit <- gam(chl ~
96     s(tr.lon, k = 60, bs = 'cr') +
97     s(tr.lat, k = 60, bs = 'cr') +
98     s(tr.bath, k = 60, bs = 'cr') +
99     s(jul.day, k = 120, bs = 'cr') +
100    s(chl.sw, k = 60, bs = 'cr'),
101    family = Gamma(link = "log"),
102    method = "REML"
103
104
105
106
107 Sadly the latitude and longitude are not interacted due to time to compute
108 solution being too long.
109
110
111 ## Part (c)
112
113 In this sort of modeling the aim is to improve on simply using the satellite
114 measurements as predictions of the direct measurements. Given the large number
115 of data, it is easy to end up using rather complex models after quite a lot of
116 examination of the data. It is therefore important to check that the model is
```



```
[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

```
git pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (2/2) done.
remote: Total 5 (delta 0), reused 0 (delta 0)
Unpacking objects:
From https://github.com/curtis-miller/DATAWorks22-Git-Intro
  a2e7bd9..0ce384f
Updating a2e7bd9..0ce384f
Fast-forward
 .gitignore
 vignettes/WoodCh7E.Rmd
 2 files changed, 4 insertions(+)

git checkout -b probs6n8
Switched to a new branch 'probs6n8'

vim vignettes/WoodCh7Exercises.Rmd

git add --all

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(+)

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

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/new/probs6n8
remote:
To https://github.com/ntguardian/DATAWorks22-Git-Intro.git
 * [new branch]      probs6n8 -> probs6n8
branch 'probs6n8' set up to track 'origin/probs6n8'.
```

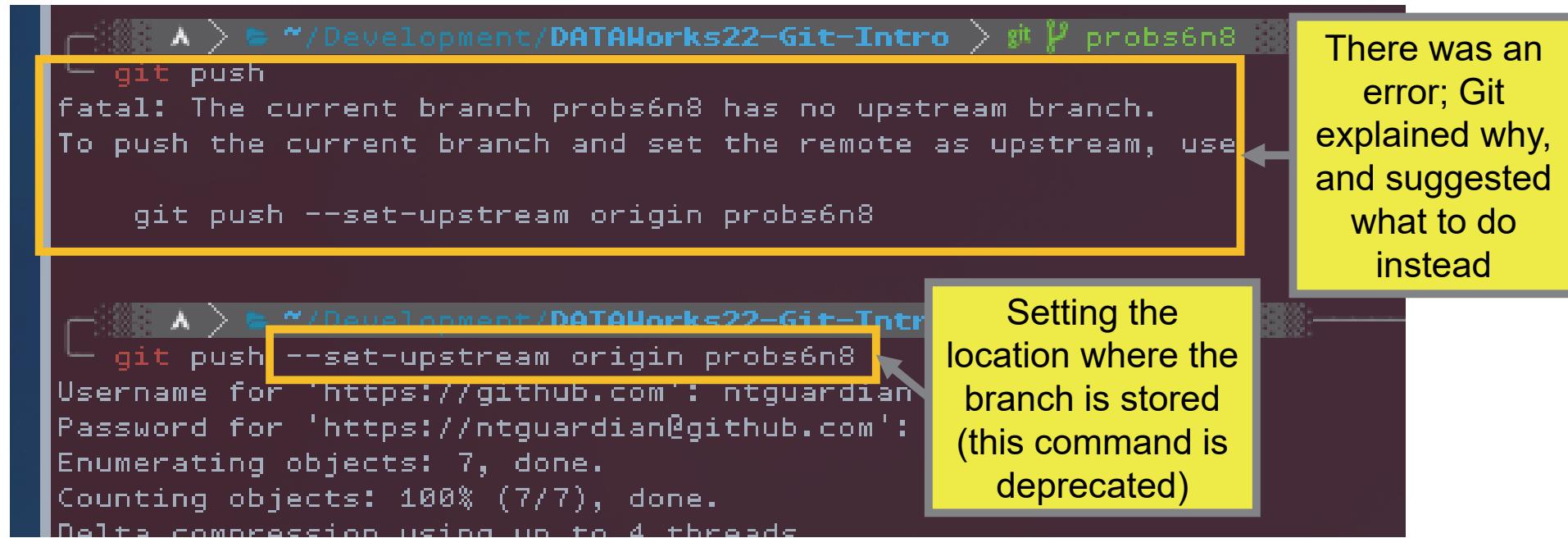
Note how to add a branch to the external repo

```
A > ~/Development/DATAWorks22-Git-Intro > git p 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

A > ~/Development/DATAWorks22-Git-Intro > git p 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



```
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
```

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

```
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
```

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 branch you want to check out.
  See git-pull(1) for details.

git pull <remote> <branch>
```

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

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

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

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 MTNGW64 ~/Development  
$ git push -u origin cmiller ← Setting the location where the  
Enumerating objects: 15, done. branch is stored  
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'.
```

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

```
cmitter@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'.

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

Name the branch to checkout

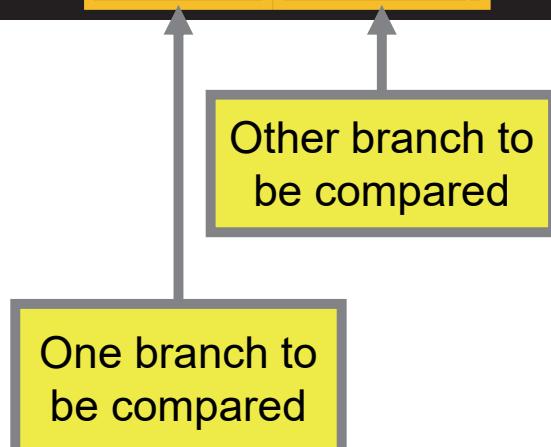
It's now the active branch; it's what we see on our computer

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-IT MTNGW64 ~/Development/git/DATAworks22-Git-Intro (probs6n8)
$ git diff cmiller probs6n8
```



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(comment = "##")
 ...
 
+## Problem 6
+
+*The data frame `ipo` of 'Initial Public Offerings' between 1960 and 2000.
```

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

```
### 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

**“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.”**

```
cmitter@CMTILLER-IT MINGW64 ~/Development/git/
$ git diff --name-only
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 $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
cmiller@CMILLER-LT MINGW64 $ 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

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git push
Enumerating objects: 11, done
Counting objects: 100
Delta compression using up to 4 threads
Compressing objects:
Writing objects: 100% (8/8), 1.12 KiB | 0 bytes/s
Total 8 (delta 4), reused 0 objects (0 bytes), pack-reused 0 objects
remote: Resolving deltas: 100% (4/4)
To https://github.com/cmiller/DATAWorks22-Git-Intro.git
 Oce384f..0569dd3
```

The screenshot shows a GitHub repository interface. On the left, a terminal window displays a git push command and its output, showing a merge commit from branch 'main' to 'main'. On the right, the GitHub repository page for 'DATAWorks22-Git-Intro' is shown. The repository has 3 branches and 0 tags. A recent commit by 'Curtis Miller' is visible, along with other files like 'vignettes', '.gitignore', 'LICENSE', and 'README.md'. A modal window titled 'Switch branches/tags' is open, showing the current branch 'main' selected. Other branches listed are 'cmiller' and 'probs6n8'. The 'View all branches' link is at the bottom of the modal.

## Our log shows a more interesting structure now

```
cmiller@CMILLER-LT MTNGW64 ~/Development/git
$ git log --graph --oneline --all
* 0569dd3 (HEAD -> main, origin/main, origin) (main)
 For a more interesting
 visualization
* 20cfa28 but did not complete deviance computation on test set
* ed2d1fe Changed model to align with the one requested
* 01361ea (origin/cmiller) Restore old GAM model and start working on train/test
 split
* 0a00e1b Attempting part (b) using log link/gamma family
| * f3c3b46 (origin/probs6n8, probs6n8) Addi or probs 6 and 8
||/
* 0ce584f 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
```

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

```
188 Now fit a GAM.
189
190
191 191 {r}
192 gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
193
194 chl_model <- chl ~
195 s(tr.bath, k = 60, bs = 'cr') +
196 s(jul.day, k = 120, bs = 'cr') +
197 s(chl.sw, k = 60, bs = 'cr')
198
199 chl_fit <- gam(chl_model,
200 data = chl_transform,
201 family = Gamma(link = "log"),
202 method = "REML"
203)
204
205 summary(chl_fit)
206 206 main
207
208
209
210
211
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...

```
165 ← ### Part (b)
166
167 *Using `mgcv::gam`, try modelling the data using a model of the sort suggested
168 (but with predictors transformed as in part (a)). Make sure that you use an
169 appropriate family. It will probably help to increase the default basis
170 dimensions used for smoothing, somewhat (especially for `jul.day`). Use the
171 '"cr"' basis to avoid excessive computational cost.*
```

```
172
173 ````{r}
174 gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
175 ````
```

```
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
Automatic merg
188 Now fit a GAM.
189
190 - ````{r}
191 + sssssss HEAD
192 gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
193 =====
194 chl_model <- chl ~
195 s(tr.bath, k = 60, bs = 'cr') +
196 s(jul.day, k = 120, bs = 'cr') +
197 s(chl.sw, k = 60, bs = 'cr')
198
199 chl_fit <- gam(chl_model,
200 data = chl_transform,
201 family = Gamma(link = "log"),
202 method = "REML"
203)
204
205 summary(chl_fit)
206 >>>>> main
207 ^#
208
209 - ### Part (c)
210
211 *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
Automatic merg
188 Now fit a GAM.
189
190 - ````{r}
191 + ````{r} HEAD
192 gam(chl ~ s(lat) + s(lon), data = chl, family = gaussian) ← Lines from this branch
193
194 chl_model <- chl ~
195 s(tr.bath, k = 60, bs = 'cr') +
196 s(jul.day, k = 120, bs = 'cr') +
197 s(chl.sw, k = 60, bs = 'cr')
198
199 chl_fit <- gam(chl_model,
200 data = chl_transform,
201 family = Gamma(link = "log"),
202 method = "REML")
203
204 summary(chl_fit)
205
206 ````{r} main ← Lines from incoming branch
207
208
209 - ### Part (c)
210 + *In this sort of modeling the aim is to improve on simply using
```

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
Automatic merg
188 Now fit a GAM.
189
190 191 +-----+ HEAD
192 193 +-----+ main
194 chl_model <- chl ~
195 s(chl.bath, k = 60,
196 s(jul.dev, k = 120,
197 s(chl.sw, k = 60, b
198
199 chl_fit <- gam(chl_model,
200 data = chl_transform,
201 family = Gamma(link =
202 method = "REML"
203)
204
205 summary(chl_fit)
206 207 +-----+ main
208
209 210 +-----+ main
211 *** Part (c)
212 *In this sort of modeling the aim is to improve on simply using
```

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...

```
187
188 Now fit a GAM.
189
190 190 ~ ``{r}
191 chl_model <- chl ~
192 s(tr.bath, k = 60, bs = 'cr') +
193 s(jul.day, k = 120, bs = 'cr') +
194 s(chl.sw, k = 60, bs = 'cr')
195
196 chl_fit <- gam(chl_model,
197 data = chl_transform,
198 family = Gamma(link = "log"),
199 method = "REML"
200)
201
202 summary(chl_fit)
203 203 ~ ``{r}
204
205 205 ~ ### Part (c)
206
207 207 # 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)

```
187
188 Now fit a GAM.
189
190 ~~~{r}
191 ch1_model <- ch1 ~
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
```

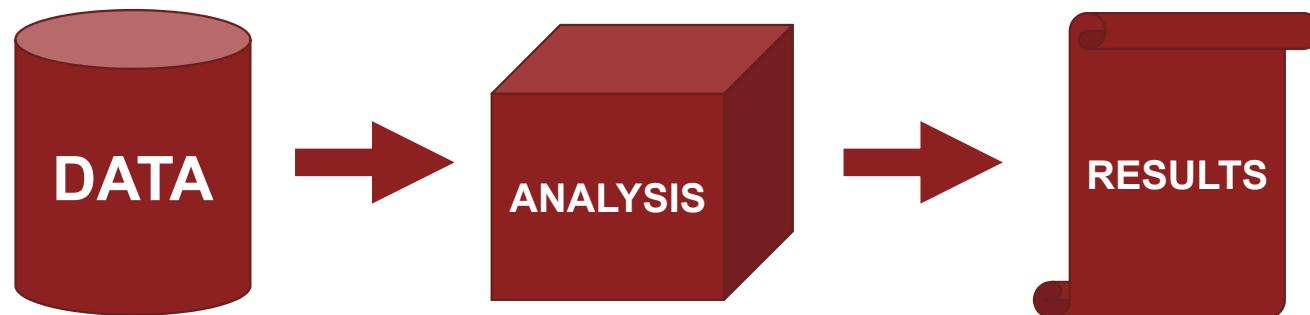
# Basic branch management commands

Command	Description	Examples	
git branch	Branch management command	git branch -d somebranch git branch -m old new	# Delete branch # 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
<code>git log</code>	See commit history; setting options can allow for making useful displays
<code>git diff</code>	See what files changed between commits and in what ways
<code>git bisect</code>	Find the first “bad” commit (say where a bug was introduced)
<code>git stash</code>	“Stash” changes, then reset to the most recent commit
<code>git grep</code>	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**

<b>1. REPORT DATE</b> 04-2022		<b>2. REPORT TYPE</b> Final		<b>3. DATES COVERED</b>	
				<b>START DATE</b>	<b>END DATE</b> Apr 2022
<b>4. TITLE AND SUBTITLE</b> DATAWorks 2022 - Introduction to git					
<b>5a. CONTRACT NUMBER</b> HQ0034-19-D-0001		<b>5b. GRANT NUMBER</b>		<b>5c. PROGRAM ELEMENT NUMBER</b>	
<b>5d. PROJECT NUMBER</b> BD-09-2299		<b>5e. TASK NUMBER</b> 229990		<b>5f. WORK UNIT NUMBER</b>	
<b>6. AUTHOR(S)</b> Miller, Curtis, G.					
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Institute for Defense Analyses 730 East Glebe Road Alexandria, Virginia 22305			<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b> NS D-33021 H 2022-000108		
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Director, Operational Test and Evaluation 1700 Defense Pentagon Washington, DC 20301			<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>  DOT&E	<b>11. SPONSOR/MONITOR'S REPORT NUMBER</b>	
<b>12. DISTRIBUTION/AVAILABILITY STATEMENT</b> Public release approved. Distribution is unlimited.					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT</b> 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.					
<b>15. SUBJECT TERMS</b> Git; Reproducible Research; reproducible analyses; Data Management; tutorial					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b> SAR	<b>18. NUMBER OF PAGES</b> 143	
<b>a. REPORT</b> Unclassified	<b>b. ABSTRACT</b> Unclassified	<b>c. THIS PAGE</b> Unclassified			
<b>19a. NAME OF RESPONSIBLE PERSON</b> John T. Haman			<b>19b. PHONE NUMBER</b> 703-845-2132		