



INSTITUTE FOR DEFENSE ANALYSES

## Introduction to Git

Dr. John T. Haman, Project Leader

Dr. Curtis G. Miller

April 2022

This publication has not been approved by the sponsor for distribution and release. Reproduction or use of this material is not authorized without prior permission from the responsible IDA Division Director.

IDA Document NS D-33021

Log: H 2022-000108

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



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

#### About This Publication

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

#### Acknowledgments

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

#### For more information:

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

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

#### Copyright Notice

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

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

Rigorous Analysis | Trusted Expertise | Service to the Nation

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

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

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

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

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

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

**Downloads**  
GUI clients and binary releases for all major platforms.

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

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

Windows GUIs Tarballs  
Mac Build Source Code

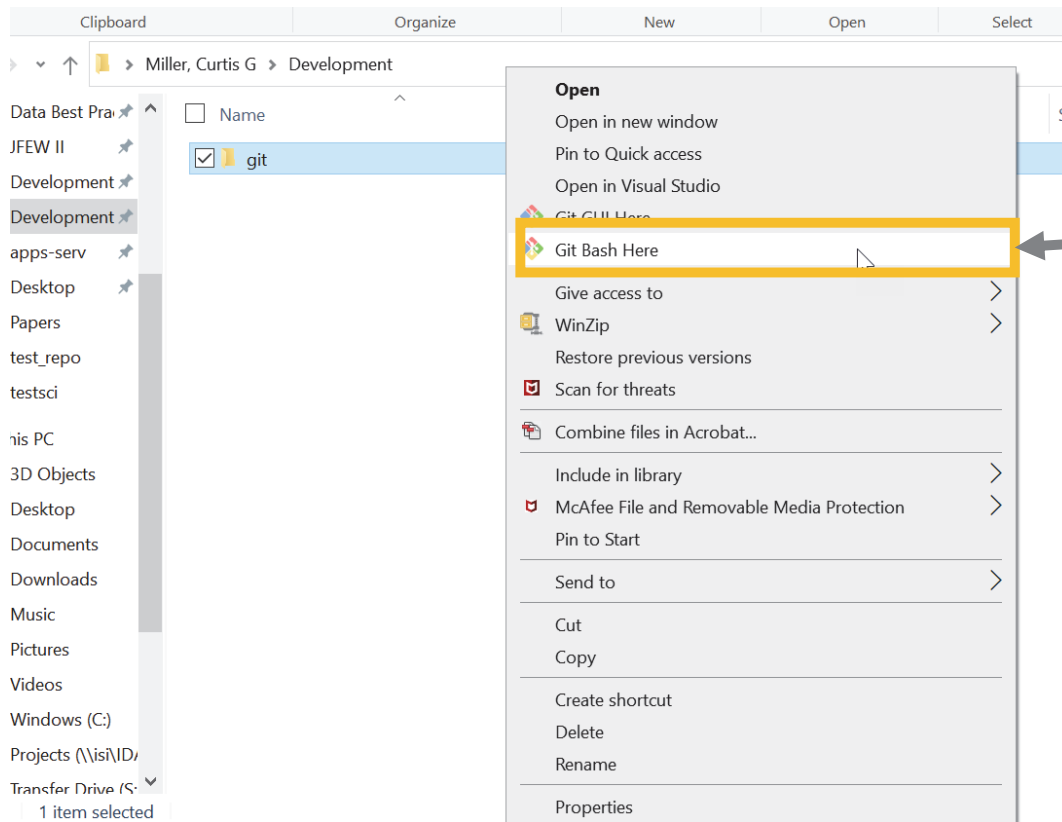
Pro Git by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Companies & Projects Using Git  
Google FACEBOOK Microsoft Twitter LinkedIn NETFLIX 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

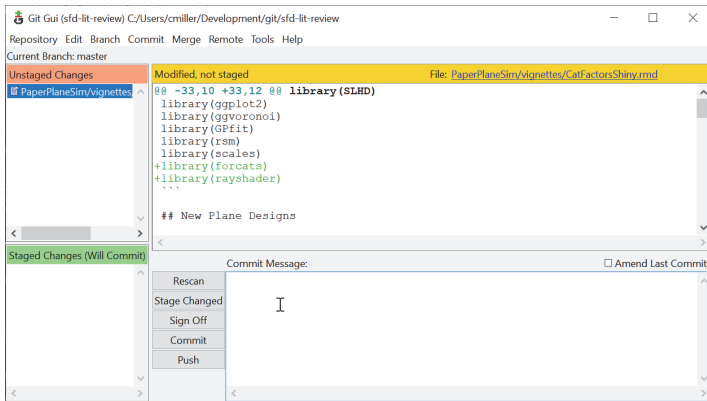
I'm going to assume you have Git installed now.

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

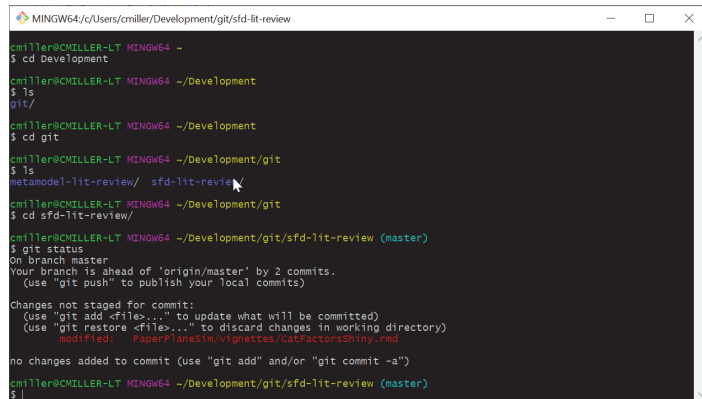


On Windows: click the folder, then right-click for this menu, then click Git Bash for a Bash terminal, or Git GUI for a graphical interface

# Git on Windows provides a GUI and a Bash command line

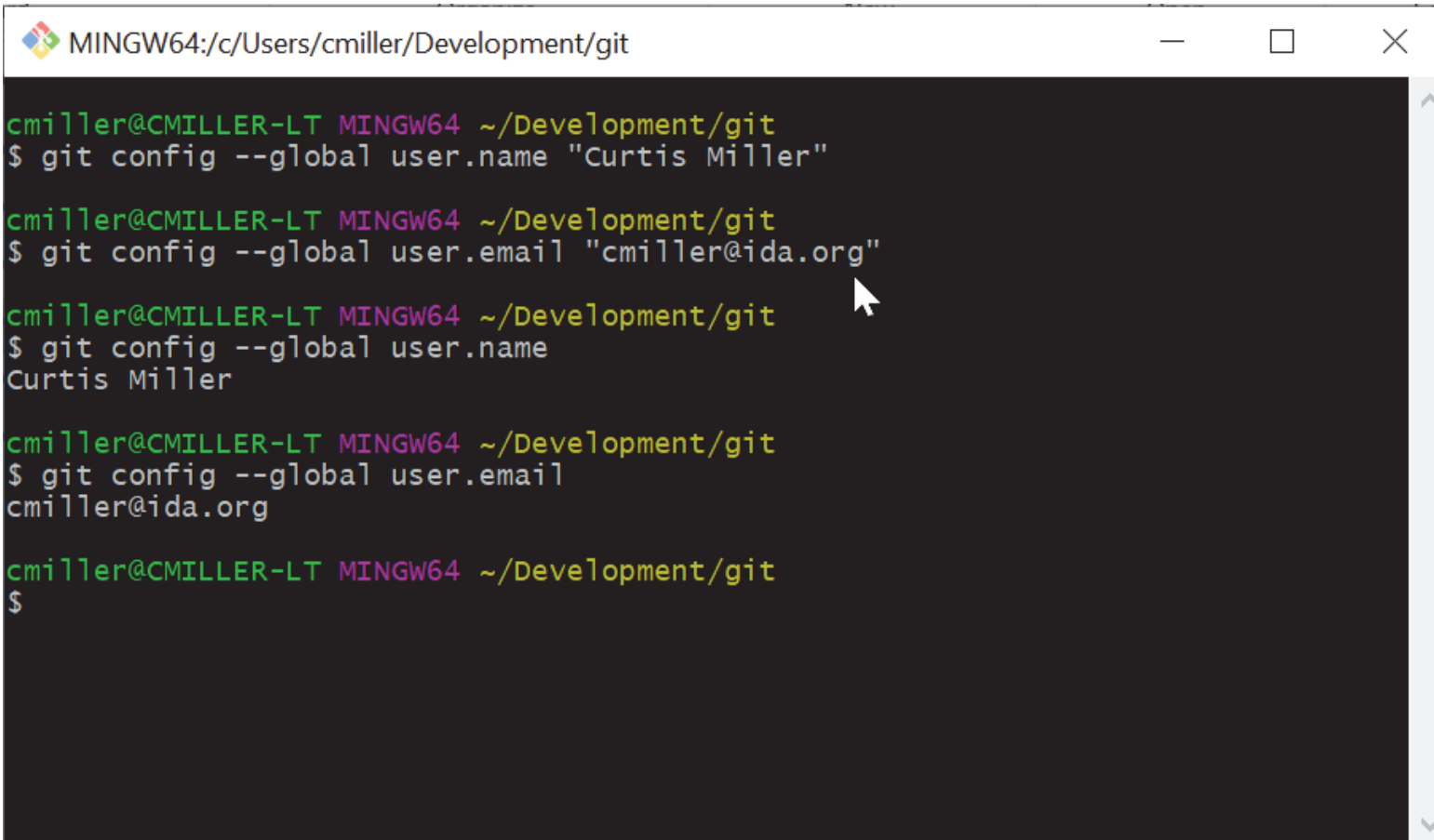


- The Windows Git installer provides both a GUI and a Bash terminal emulator for using Git
- While the GUI may seem beginner friendly, it serves mostly as a front-end to the command line interface
- Full utilization of Git's power is easiest when using the command line
- Git concepts are easier to explain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



GUI: Graphical User Interface

## Set up initial user parameters using git-config



```
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
$
```

## We can determine a lot from a Bash prompt

A terminal window titled 'MINGW64:/c/Users/cmiller/Development/' displays a series of Git configuration commands and their outputs. The prompts are annotated with yellow boxes and arrows pointing to explanatory text:

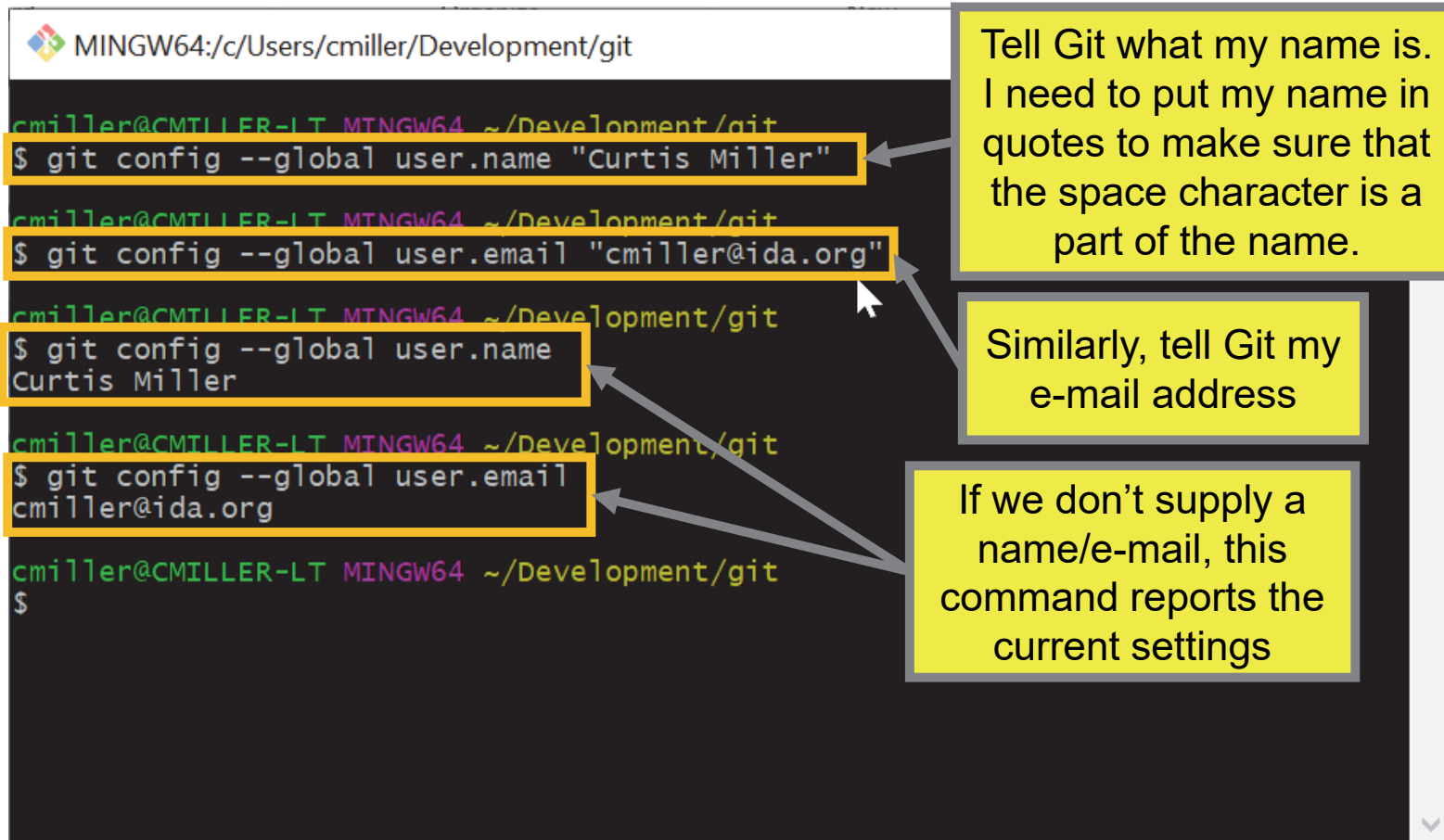
- Username (before @) and computer name (after @):** Points to the prompt 'cmiller@CMILLER-LT MINGW64 ~/Development/'.
- Working directory:** Points to the path '~/.Development/git' in the prompt.
- Prompt awaiting your command:** Points to the '\$' symbol at the end of the prompt.

```
MINGW64:/c/Users/cmiller/Development/
cmiller@CMILLER-LT MINGW64 ~/Development/
$ 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
$
```

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



The image shows a terminal window with the following text:

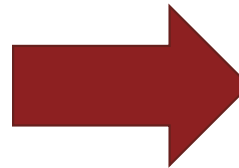
```
MINGW64:/c/Users/cmiller/Development/git  
cmiller@CMTLLER-LT MINGW64 ~/Development/git  
$ git config --global user.name "Curtis Miller"  
cmiller@CMTLLER-LT MINGW64 ~/Development/git  
$ git config --global user.email "cmiller@ida.org"  
cmiller@CMTLLER-LT MINGW64 ~/Development/git  
$ git config --global user.name  
Curtis Miller  
cmiller@CMTLLER-LT MINGW64 ~/Development/git  
$ git config --global user.email  
cmiller@ida.org  
cmiller@CMTLLER-LT MINGW64 ~/Development/git  
$
```

Callouts:

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

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

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



file:///C:/Program Files/Git/mingw64/share/doc/git-doc/git-config.html

## 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**

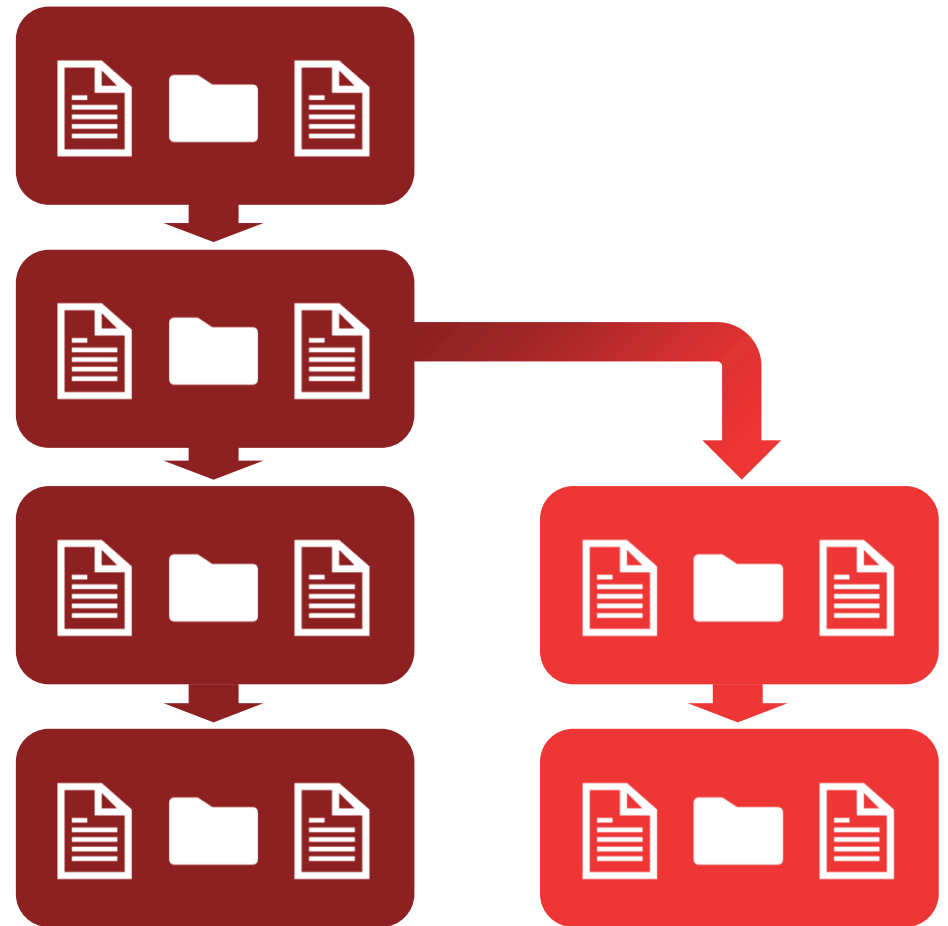
Git powers popular code sharing websites such as GitHub



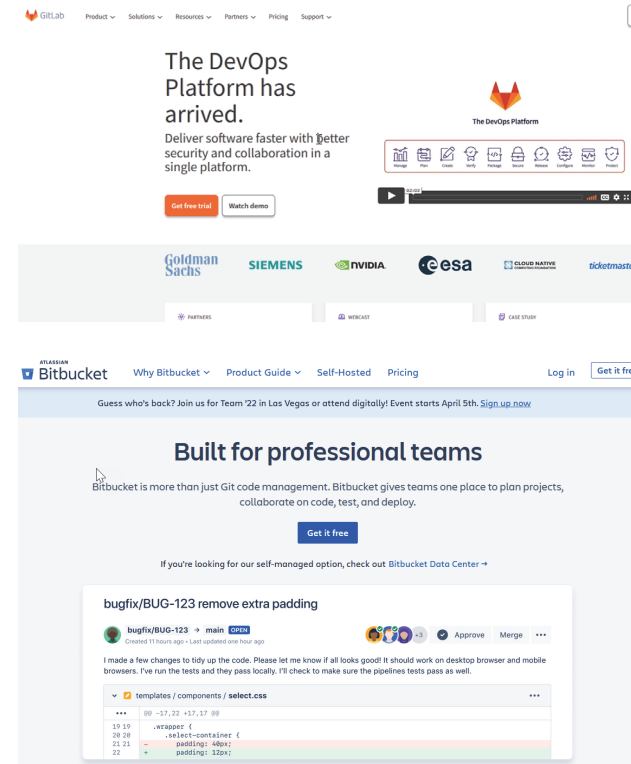
# Repos

## A Git repository is a collection of tracked files

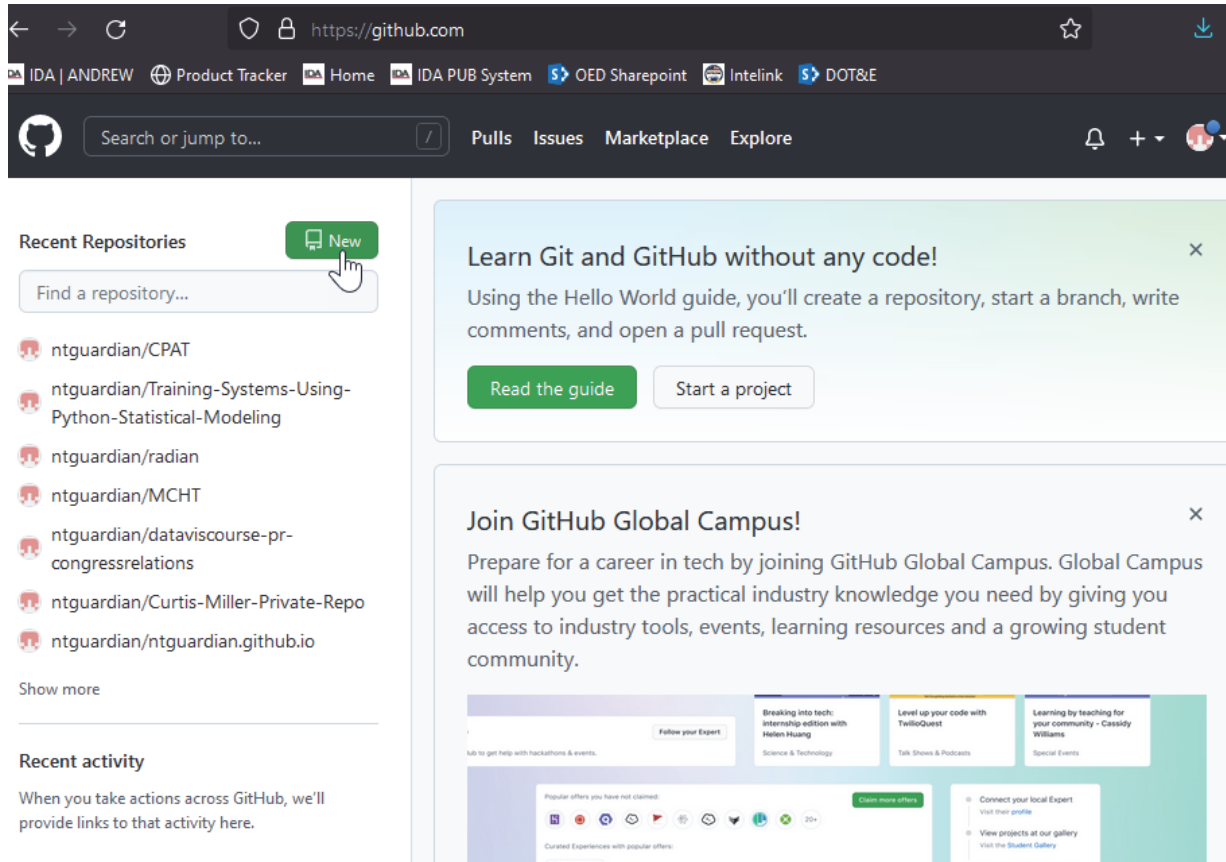
- Git tracks the history of these files and their differences across time and between branches
- Repos generally contain entire projects
- Users access the project by accessing the repo, and the project is shared by sharing the repo
- Repos are as simple as folders containing files



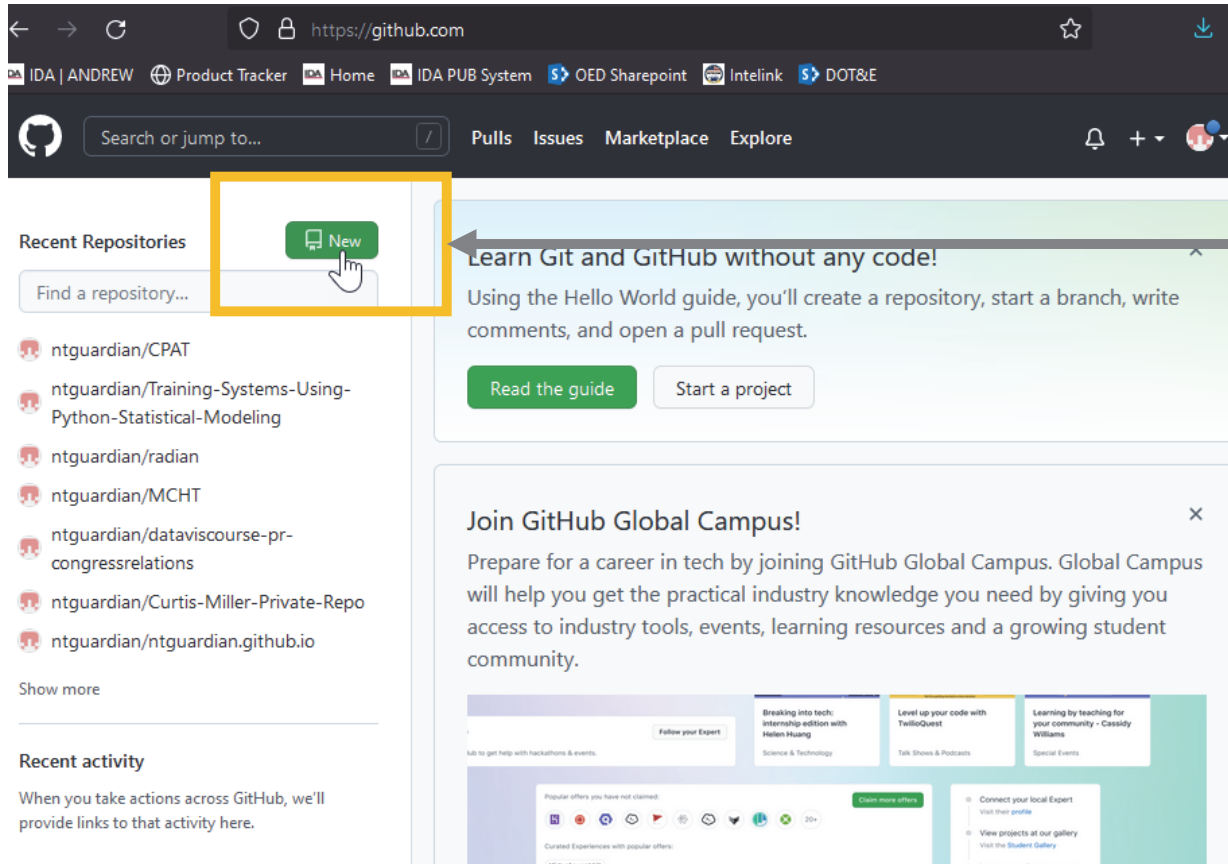
# Many web services host repos (but you don't need to use these to use Git on your own computer)



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



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



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 <sup>\*</sup> / Repository name <sup>\*</sup>  
ntguardian / DATAWorks22-Git-Intro ✓

Great repository names are short and memorable. Need inspiration? How about [bookish-train](#)?

### Description (optional)

This is a repository for demonstrating how to use Git.

- Public**  
Anyone on the internet can see this repository. You choose who can commit.
- Private**  
You choose who can see and commit to this repository.

### Initialize this repository with:

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

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

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

.gitignore template: R ▾

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

License: MIT License ▾

This will set `main` as the default branch. Change the default name in your [settings](#).

**i** You are creating a public repository in your personal account.

Create repository

Click me to  
make a new  
repo on  
GitHub

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

# Now we can view our repo online

The screenshot shows a web browser displaying the GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The browser's address bar shows the URL 'https://github.com/ntguardian/DATAWorks22-Git-Intro'. The repository is public and has 1 watch, 0 stars, and 0 forks. The repository contains three files: '.gitignore', 'LICENSE', and an 'Initial commit' by 'ntguardian' at 'now'. A blue banner prompts the user to 'Add a README'. The right sidebar shows sections for 'About', 'Releases', and 'Packages', all of which are currently empty.

ntguardian / DATAWorks22-Git-Intro (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights

main

File	Commit	Time
.gitignore	Initial commit	now
LICENSE	Initial commit	now

Help people interested in this repository understand your project by adding a README. [Add a README](#)

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

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

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

# Now we can view our repo online

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The browser address bar contains the URL 'https://github.com/ntguardian/DATAWorks22-Git-Intro'. The repository name 'ntguardian / DATAWorks22-Git-Intro' is highlighted in the top navigation bar. The 'main' branch is selected in the branch dropdown menu. The file list shows '.gitignore' and 'LICENSE', both marked as 'Initial commit'. The right sidebar shows repository statistics: 0 stars, 1 watching, and 0 forks. A blue banner at the bottom encourages adding a README.

Repo URL, important for sharing

Repo identifier (note connection to URL)

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

Which branch we are viewing (will discuss branches later)

# We can view files online

The file we are viewing, including what branch

main DATAWorks22-Git-Intro / LICENSE

Who works on the file and recent activity

ntguardian Initial commit Latest commit 3c0de05 9 minutes ago History

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

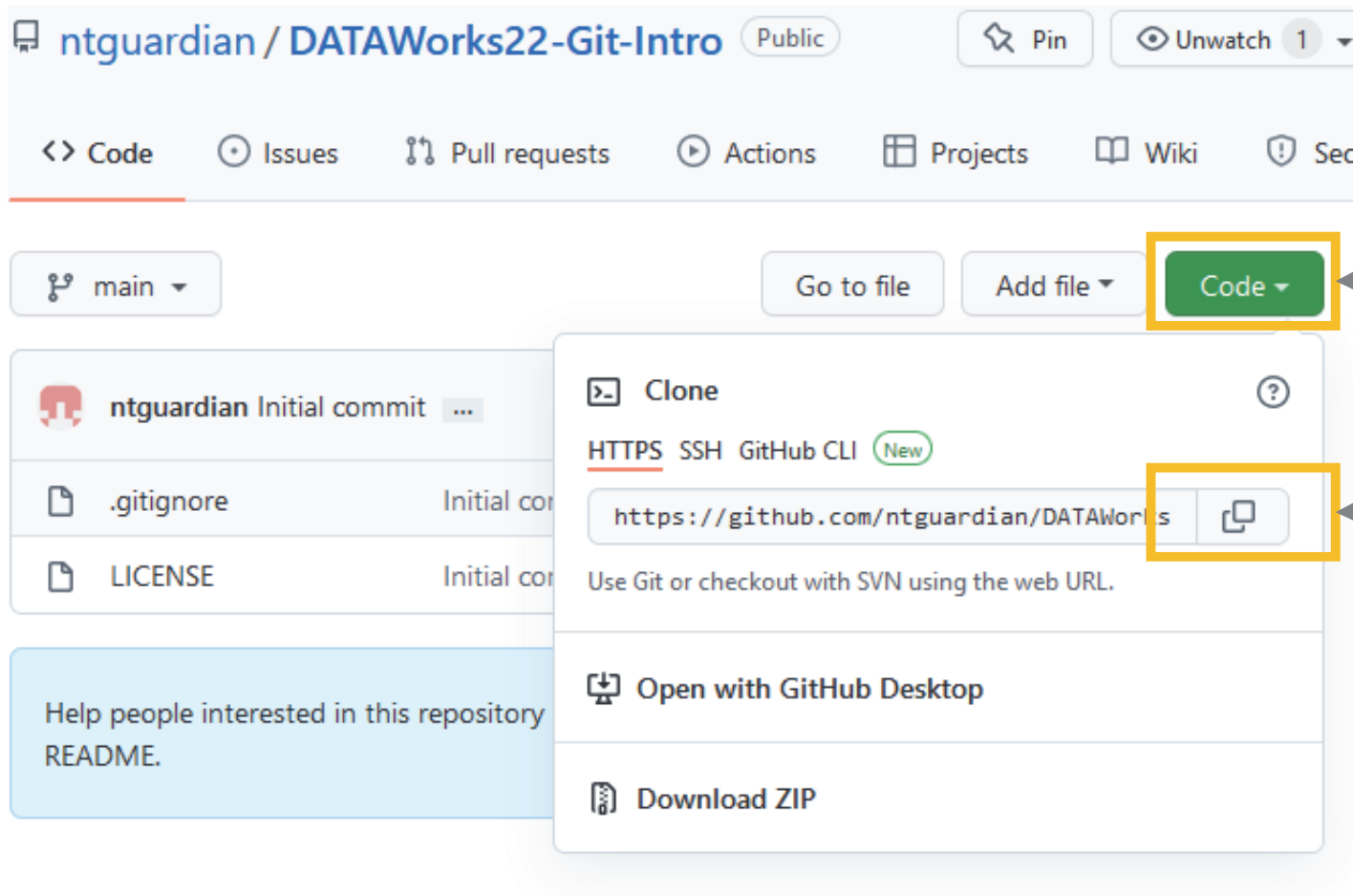
File contents

```
21 lines (17 sloc) 1.04 KB
1 MIT License
2
3 Copyright (c) 2022 ntguardian
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
```

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

The image shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The repository is public. The navigation bar includes 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', and 'Security'. The 'main' branch is selected. The 'Code' button is highlighted in green, and its dropdown menu is open, showing options to clone the repository. The 'Clone' section includes links for 'HTTPS', 'SSH', 'GitHub CLI', and 'New'. The HTTPS URL is 'https://github.com/ntguardian/DATAWorks' and is copied to the clipboard. Below the URL, it says 'Use Git or checkout with SVN using the web URL.' Other options in the dropdown are 'Open with GitHub Desktop' and 'Download ZIP'. The repository content shows files like '.gitignore' and 'LICENSE', and a commit message 'ntguardian Initial commit'. A blue box at the bottom left says 'Help people interested in this repository README.'

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



Click this to open a dialog to allow easy cloning

Click to copy the repo URL for cloning

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

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

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

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

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

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

The `git clone` command

The URL you copied; wrap in quotes to be safe

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

The image shows a terminal window with the following output:

```
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.
```

Callout boxes provide context:

- Output once you run the command; everything is okay**: Points to the terminal output.
- Now it's on our computer**: Points to the newly created folder in the file explorer.
- What's inside the directory; it matches what we saw on GitHub (except for .git; ignore this folder)**: Points to the contents of the folder: `.git`, `.gitignore`, and `LICENSE`.

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

## Summary: Making and getting repos

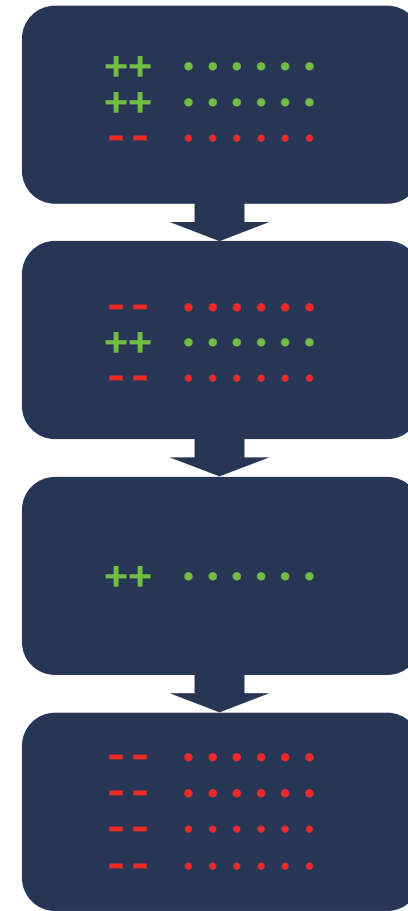
Command	Description	Examples
<code>git init</code>	Create a new repo	<code>git init</code>
<code>git clone</code>	Clone an existing repository	<code>git clone "https://github.com/someuser/someproject"</code>



# 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'. The content of the file is as follows:

```
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.
```

Below the editor, a file explorer shows the directory structure of the repository. The files listed are:

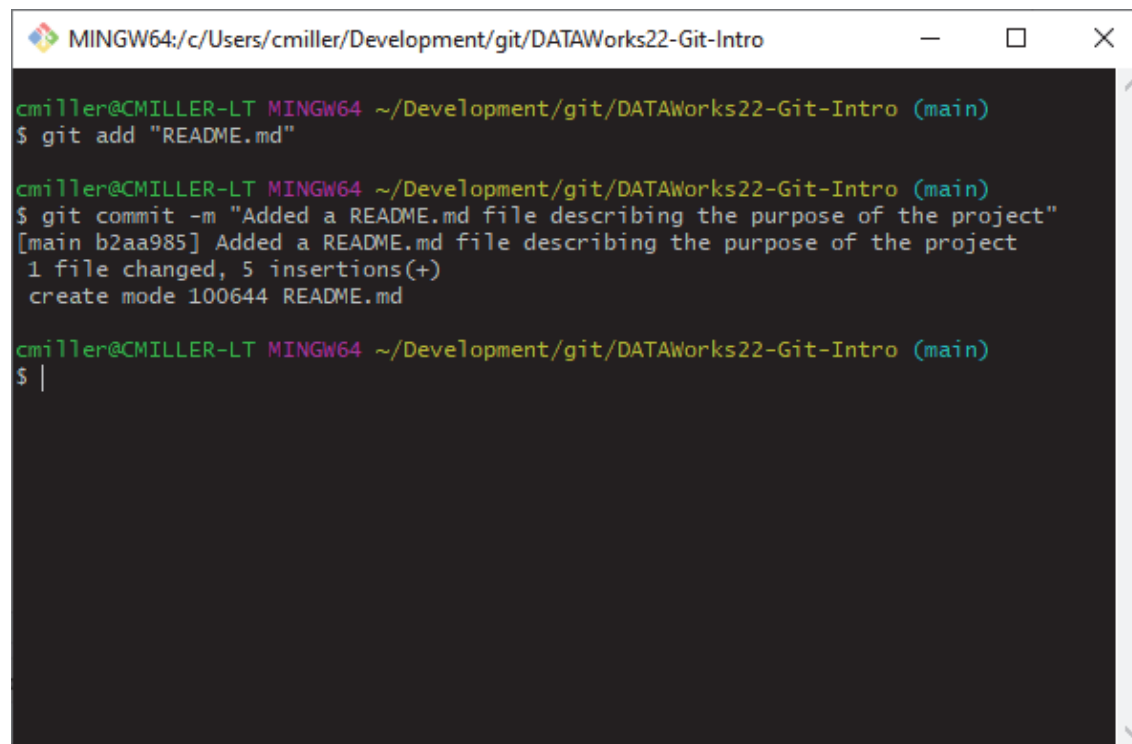
Name	Date modified
.git	2022-04-06 15:27
.gitignore	2022-04-06 15:27
LICENSE	2022-04-06 15:27
README.md	2022-04-06 16:11

The 'README.md' file is highlighted with a yellow box in the file explorer. A yellow callout box points to it with 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."

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

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

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



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

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

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

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

Name the file with the changes to track

`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

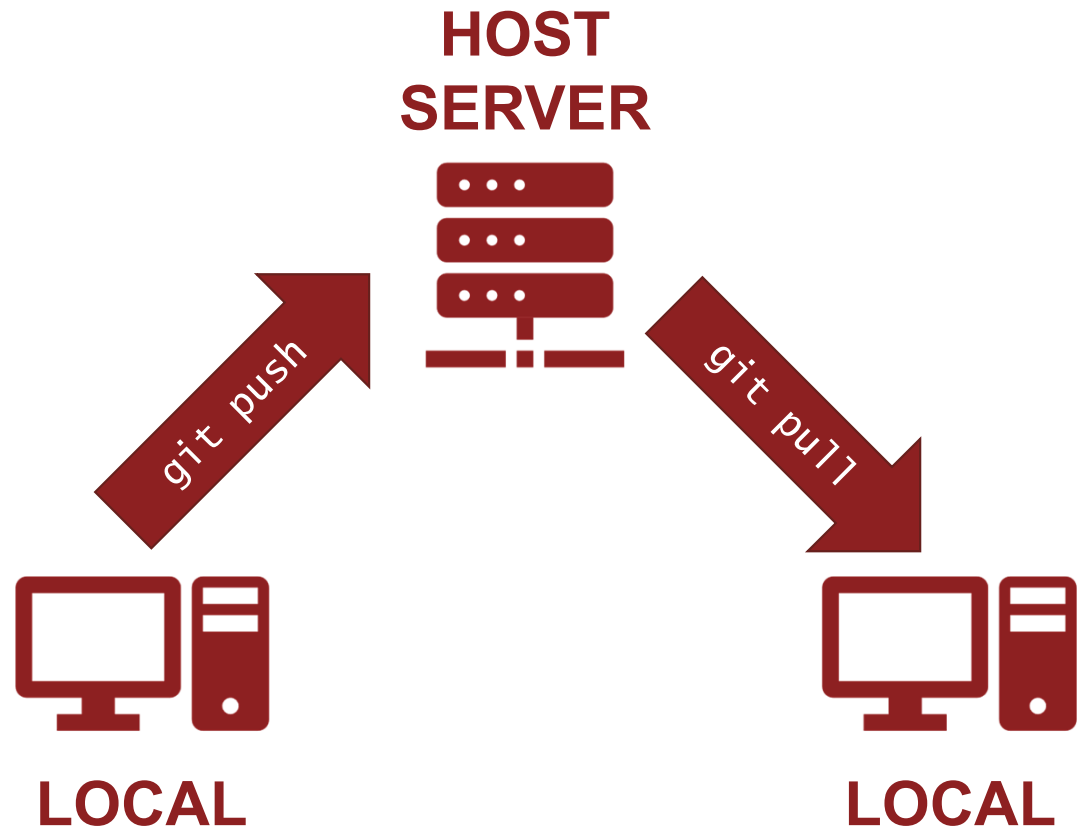
The `-m` option is for adding a message summarizing what changes were made in the commit (keep it short)

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

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

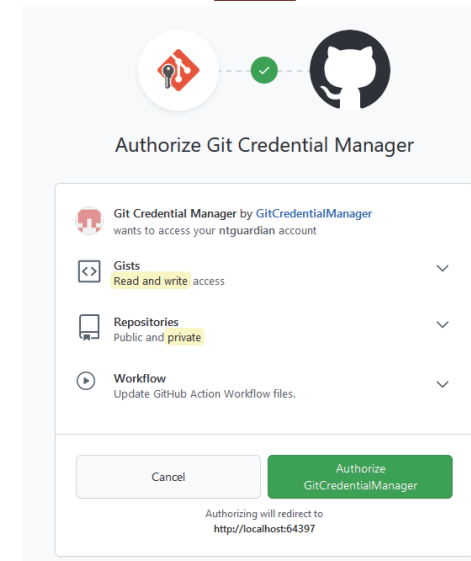
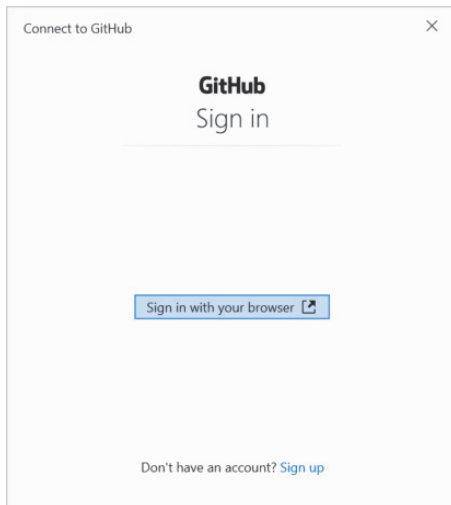
- The commands `git push` and `git pull` keep local copies of git repositories up-to-date with the external host of the repository
- After committing changes, use `git push` to propagate those changes to the host (and thus to everyone else with an up-to-date repo)
- Push changes at least at the end of 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'.
```

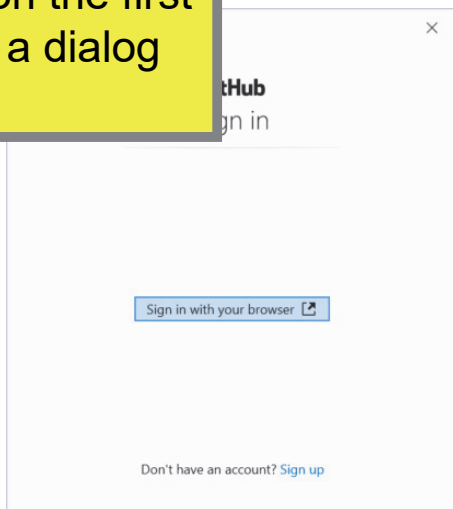


# Pushing to GitHub for the first time may require authentication

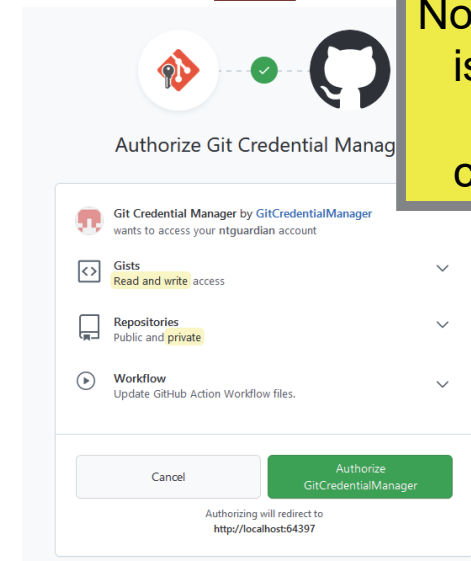
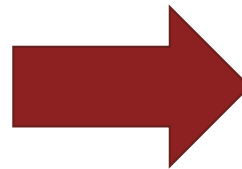
`git push` is the command for propagating changes to the external repo copy

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

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



Follow the instructions...



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

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

# The changes are now online

The screenshot shows a GitHub repository page for 'ntguardian / DATAWorks22-Git-Intro'. The repository is public and has 1 branch (main) and 0 tags. The commit history table is as follows:

File	Commit Message	Commit ID	Time	Commits
.gitignore	Initial commit		2 hours ago	
LICENSE	Initial commit		2 hours ago	
README.md	Added a README.md file describing the purpose of the project	b2aa985	26 minutes ago	2 commits

The README.md file content is:

## 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 RStudio with two windows. The top window displays the README.md file with the following content:

```
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
8 *Generalized
9 concerns using
10 remote sensed
11 satellite with
12 exercise in a
13 analysis.
14
15 The purpose of
16 will be explained
17
```

The bottom window displays the vignettes directory with the following content:

```
1 ---
2 title: "Wood Ch. 7 Exercises"
3 author: "Curtis Miller"
4 date: "4/6/2022"
5 output: html_document
6 ---
7
8 ```{r setup, include=FALSE}
9 knitr::opts_chunk$set(echo = TRUE)
10 ```
11
12 ## Problem 9
13
14 *This question is about creating models for calibration of satellite remote
15 sensed data. The data frame `chl` contains direct ship based measurements of
16 chlorophyll concentrations in the top 5 metres of ocean water, `chl`, as well as
17 the corresponding satellite estimate `chl.sw` (actually a multi-year average
18 measurement for the location and time of year), along with ocean depth, `bath`,
19 day of year, `jul.day` and location, `lon`, `lat`. The data are from the world
20 ocean database (see http://seawifs.gsfc.nasa.gov/SEAWIFS/ for information on
21 Seawifs). `chl` and `chl.sw` do not correlate all that well with each other,
22 probably because the reflective characteristics of water tend to change with
23 time of year and whether the water is near the coast (and hence full of
24 particulate matter) or open ocean. One way of improving the predictive power of
25 the satellite observations might be to model the relationship between directly
26 measured chlorophyll and remote sensed chlorophyll, viewing the relationship as
27 an indicator for water type (near shore vs. open), a model something like*
28
29 
$$f_1(\text{chl}_i) = f_1(\text{chl.sw}_i) f_2(\text{bath}_i)$$

30 
$$f_3(\text{jul.day}_i)$$

31
32 *might be a reasonable starting point.*
```

The right-hand pane shows a file explorer with the following table:

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

The bottom-right pane shows a file explorer for the vignettes directory with the following table:

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

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

## Track them changes!

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

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

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

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

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

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

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

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

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

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

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

There are new files in the project

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

The screenshot shows a GitHub repository page for 'Curtis Miller Structuring project around chlorophyll analysis example'. The repository has 1 branch (main) and 0 tags. The commit hash is 0f66f94, committed 13 minutes ago with 3 commits. The file list includes:

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

The README.md file content is displayed below:

## DATAWorks 2022 Introduction to Git

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

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

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

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

This file has not been altered so nothing has changed

README.md now has the new text

## Let's see the history of the project so far

main 1 branch 0 tags

Go to file Add file Code

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

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

README.md

### DATAWorks 2022 Introduction to Git

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

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

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








Click this to see the history of the project

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

# Our project has a short history

main ▾

Commits on Apr 6, 2022

<b>Structuring project around chlorophyll analysis example</b> Curtis Miller committed 24 minutes ago	 0f66f94	
<b>Added a README.md file describing the purpose of the project</b> Curtis Miller committed 1 hour ago	 b2aa985	
<b>Initial commit</b>  ntguardian committed 3 hours ago	Verified  3c0de05	

Newer Older

## From here we can explore that history

The screenshot shows a GitHub commit history for the 'main' branch. The commits are listed in reverse chronological order:

- Structuring project around chlorophyll analysis example** by Curtis Miller, committed 24 minutes ago. The commit hash `0f66f94` is highlighted with a yellow box.
- Added a README.md file describing the purpose of the project** by Curtis Miller, committed 1 hour ago. The commit hash `b2aa985` is highlighted with a yellow box.
- Initial commit** by ntau...dian, committed 3 hours ago. The commit hash `3c8c985` is highlighted with a yellow box.

Annotations on the image include:

- A yellow box pointing to the author name 'Curtis Miller' in the second commit, containing the text: "The commit author; this is why you needed to use git config".
- A yellow box pointing to the commit hash `0f66f94`, containing the text: "Click this to see what the project looked like at that commit".
- A yellow box pointing to the commit hash `b2aa985`, containing the text: "The hash of a commit; click to see what changed in that commit".
- A grey box at the bottom center containing the text: "None of the features seen here are exclusive to GitHub; other repo hosts have their own versions of this, and Git itself can allow looking at this information."

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



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

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git log
commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368 (HEAD -> main, origin/main, 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

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

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

```
MINGW64:~/Development/git/DATAWorks22-Git-Intro (main)
git log
commit 0166f942cee33fd3a42b8a7b5f9f44b62780f368 (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 3f0a0cb8b980e8cf07e1ab3
Author: Curtis Miller <cmiller@ida.org>
Date: Wed Apr 6 15:22:30 2022 -0400
    Added a README.md file describing the purpose of the project
```

Commit hash identifier (often abbreviated to first 7 characters)

Commit message describing what change was made

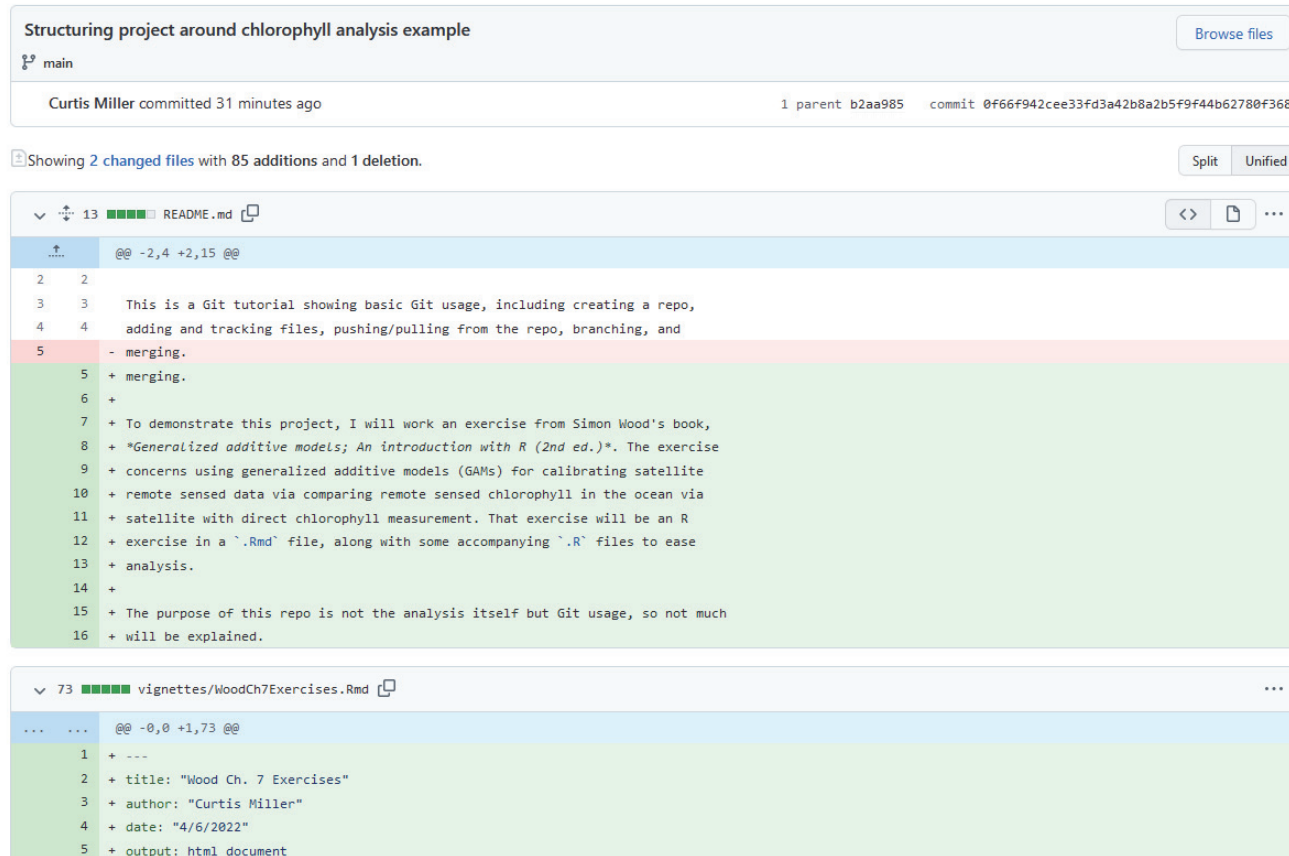
If this is too long to fit on one screen, Bash might use a program called `less` to view it; scroll up and down with the arrow keys, then press `q` to quit

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

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

```
cmiller@CMTLEP-LT-MINGW64 ~/Development/git/DATAworks22-Git-Intro (main)
$ git log --graph --oneline --decorate --all
* 0f66f94 (HEAD -> main, origin/main, origin/HEAD) Structuring project around ch
lorophyll analysis example
* b2aa985 Added a README.md file describing the purpose of the project
* 3c0de05 Initial commit
```

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



Structuring project around chlorophyll analysis example

main

Curtis Miller committed 31 minutes ago

1 parent b2aa985 commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368

Showing 2 changed files with 85 additions and 1 deletion.

Split Unified

13 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.
```

73 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)

```
Structuring project around chlorophyll analysis example
main
Curtis Miller committed 31 minutes ago 1 parent b2aa985 commit 0f66f942cee33fd3a42b8a2b5f9f44b62780f368

Showing 2 changed files with 85 additions and 1 deletion.

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

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

The full hash identifier of the commit

Additions to a file are shown in green

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

We can get the same information using `git diff`

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

## git diff can compare many entities Git tracks

git diff is the command for comparing files tracked by Git

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

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

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

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@
 
 This is a Git tutorial showing basic Git usage, including creating a repo,
 adding and tracking files, pushing/pulling from the repo, branching, and
-merging.
 \ No newline at end of file
+merging.
+
+To demonstrate this project, I will work an exercise from Simon Wood's book,
+*Generalized additive models; An introduction with R (2nd ed.)*. The exercise
+concerns using generalized additive models (GAMs) for calibrating satellite
+remote sensed data via comparing remote sensed chlorophyll in the ocean via
+satellite with direct chlorophyll measurement. That exercise will be an R
+exercise in a .Rmd file, along with some accompanying .R files to ease
+analysis.
+
+The purpose of this repo is not the analysis itself but Git usage, so not much
+will be explained.
diff --git a/vignettes/WoodCh7Exercises.Rmd b/vignettes/WoodCh7Exercises.Rmd
:|
```

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

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@
 This is a Git tutorial showing how to use Git for version control. The tutorial covers the basics of Git, including creating a repo, adding and tracking files, pushing and pulling code to and from the repo, branching, and merging.
-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
+satellite with direct chlorophyll measurement. That exercise will be
+described in a *.Rmd* file, along with some accompanying *.R* files to
+run the analysis.
+
+The purpose of this repo is not the analysis itself but Git usage, so
+the details of the analysis
+will be explained.
diff --git a/vignettes/woodCh7Exercises.Rmd b/vignettes/woodCh7Exercises.Rmd
:
```

Red: what was removed

Green: what was added

What is being compared

Location in file

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

## We can also see what changed in a specific file

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22_Git_Intro
cmiller@CMILLER-LT MINGW64 ~/Development (main)
$ git diff HEAD~ HEAD README.md
diff --git a/README.md b/README.md
index af695c0..32f5d53 100644
--- a/README.md
+++ b/README.md
@@ -2,4 +2,15 @@


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

List a specific file to see changes made to it only

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

b2aa985f4b 1 branch 0 tags Go to file Code

Curtis Miller Added a README.md file describing the purpose of the project		b2aa985 yesterday	🕒 2 commits
📄 .gitignore	Initial commit		yesterday
📄 LICENSE	Initial commit		yesterday
📄 README.md	Added a README.md file describing the purpose of the project		yesterday

README.md 

### DATAWorks 2022 Introduction to Git

---

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

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

The screenshot shows a GitHub interface. At the top, a commit hash `b2aa985f4b` is highlighted with a yellow box. To its right, a yellow callout box contains the text "This describes which commit we're looking at". Below this, the commit history is shown as a table:

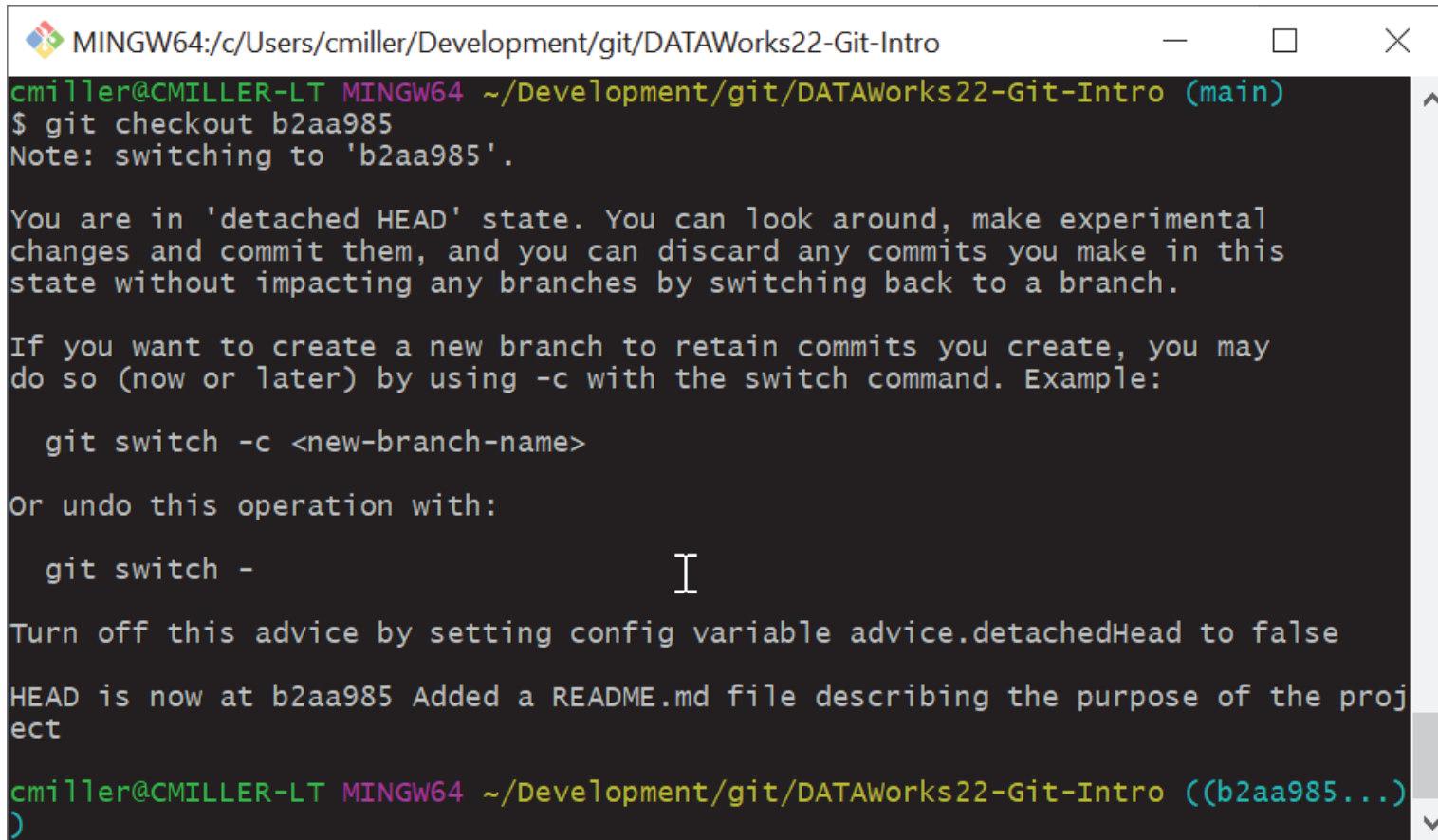
File	Commit Message	Author	Time
<code>.gitignore</code>	Initial commit	b2aa985	yesterday
<code>LICENSE</code>	Initial commit	b2aa985	yesterday
<code>README.md</code>	Added a README.md file describing the purpose of the project	b2aa985	yesterday

Below the table, the content of the `README.md` file is displayed. The title is "DATAWorks 2022 Introduction to Git". The text reads: "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

The screenshot shows a GitHub repository interface. At the top, the repository name is 'b2aa985f4b'. Below it, there's a 'Switch branches/tags' dropdown menu. The 'main' branch is selected and labeled as 'default'. A yellow callout box with the text 'Click to go back to the current state' points to the 'main' branch and the commit hash 'b2aa985f4b'. The commit history shows two commits: 'Initial commit' and 'Added a README.md file describing the purpose of the project', both dated 'yesterday'. Below the commit history, the README file content is visible, titled 'DATAWorks 2022 Introduction to Git'. The README text reads: '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 project
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

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

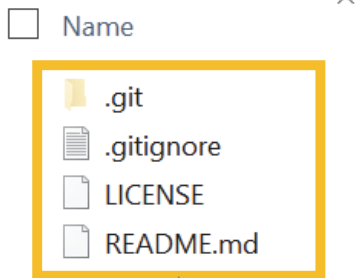
The commit we are checking out

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

Notice how this changed...

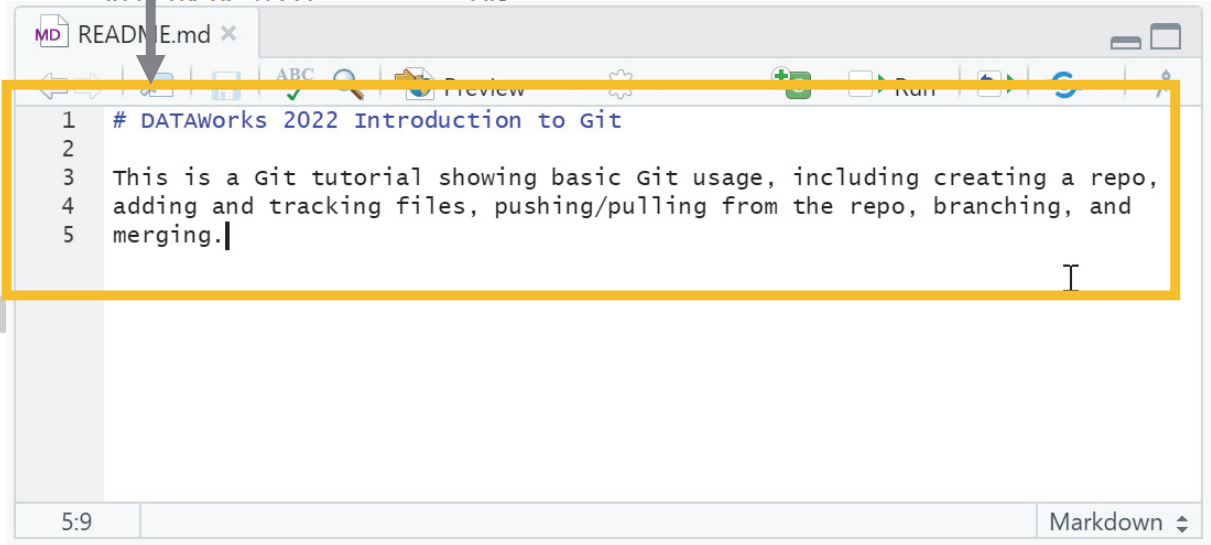
# The repo's contents on our hard drive will change when checking out a commit

Files have been changed to their old state

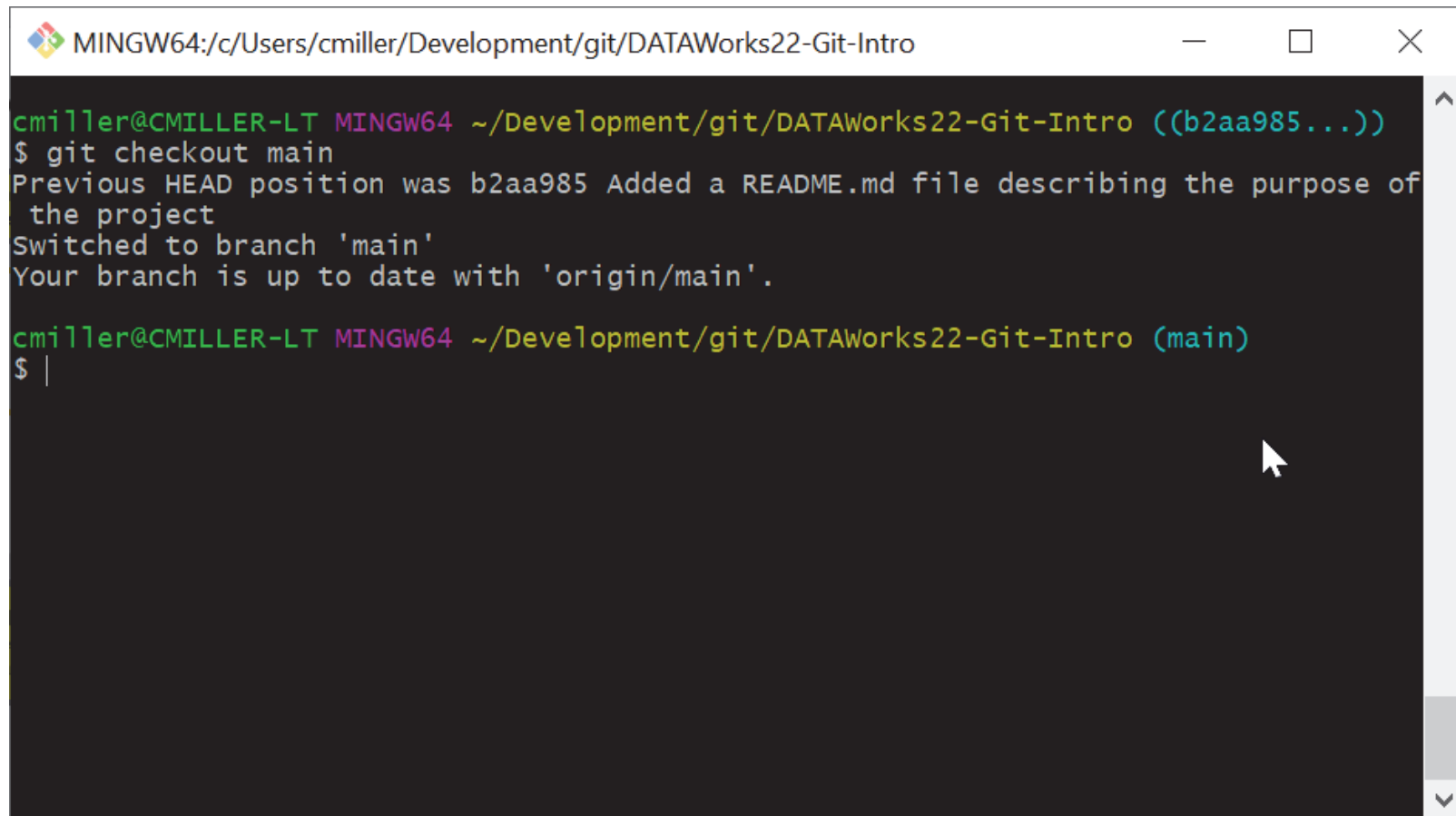


Tracked files not in the old commit have been deleted

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



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



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

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

## You check out the main branch



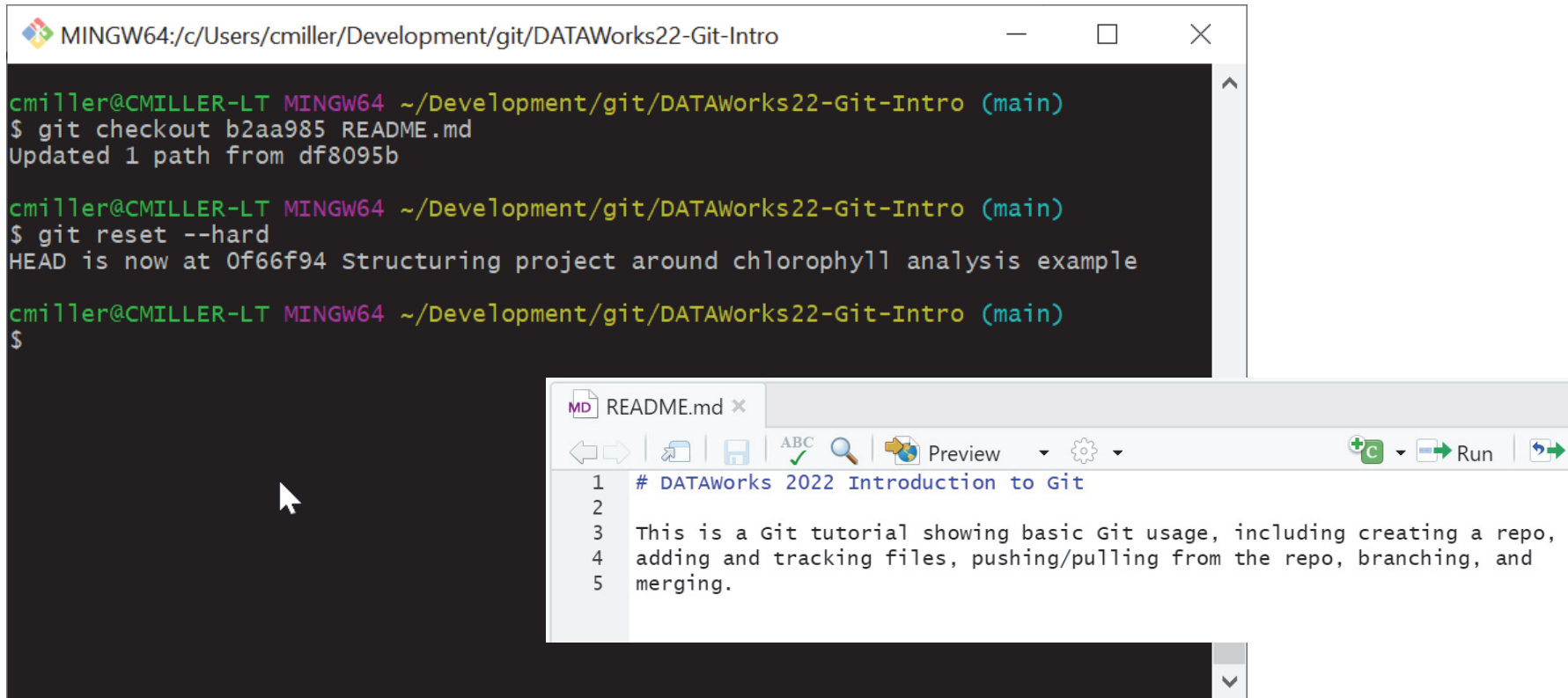
```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
cmiller@CMILLER-LT MINGW64 ~/D...-git-Intro ((b2aa985...))
$ git checkout main
Previous HEAD position was b2a...
the project
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

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

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

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

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



The image shows a terminal window and a file editor. The terminal window displays the following commands and output:

```
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

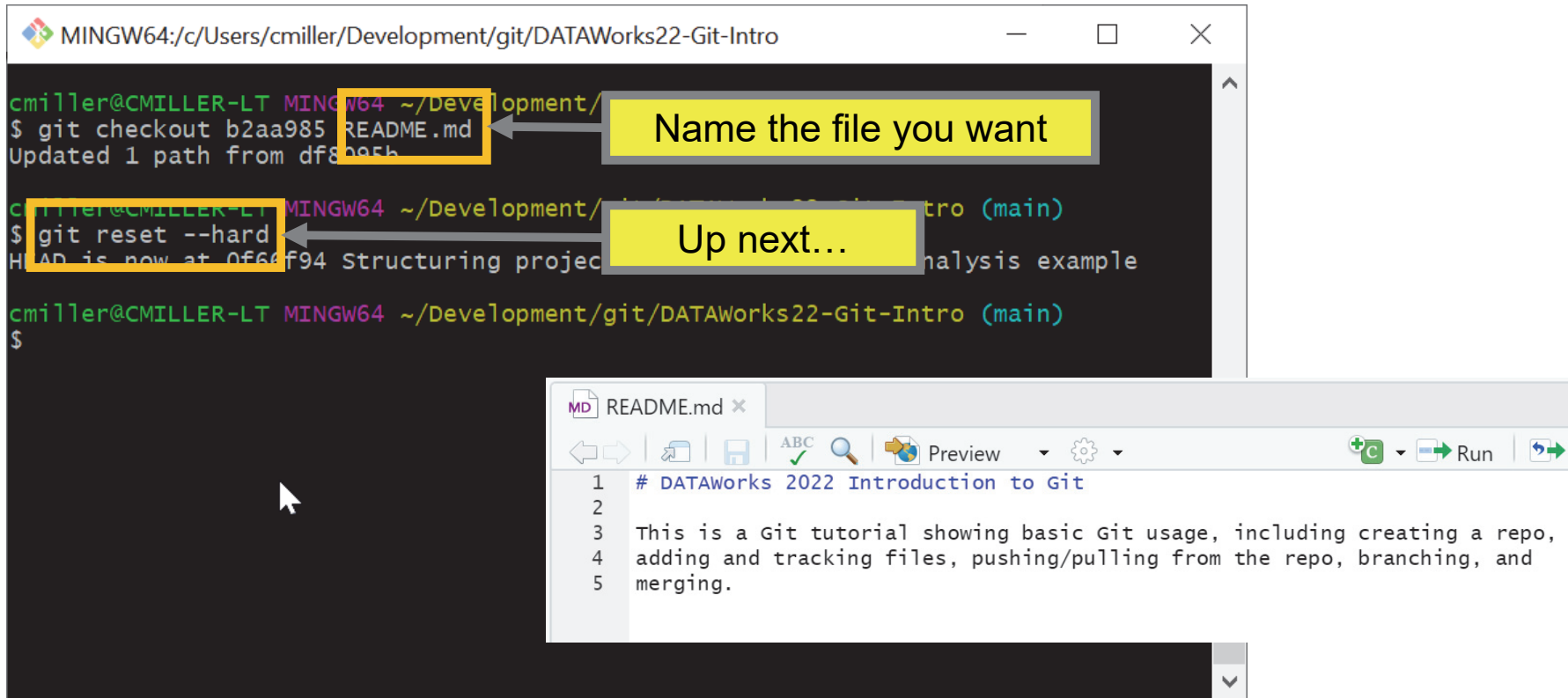
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 Structuring project around chlorophyll analysis example

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

The file editor shows the content of the README.md file:

```
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 image shows a terminal window and a file editor. The terminal window is titled "MINGW64:/c:/Users/cmiller/Development/git/DATAWorks22-Git-Intro". It displays the following commands and output:

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

cmiller@CMILLER-LT MINGW64 ~/Development/
$ git reset --hard
HEAD is now at 0f66f94 Structuring project analysis example

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





Two yellow callout boxes with arrows point to the file names in the terminal: "Name the file you want" points to "README.md" in the first command, and "Up next..." points to "README.md" in the second command.

The file editor window shows the content of "README.md":

```
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.
```

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






1

<input type="checkbox"/> Name	Date modified	Type
 .git	2022-04-07 17:38	File folder
 vignettes	2022-04-07 17:38	File folder
 .gitignore	2022-04-06 15:27	Text Documer
 LICENSE	2022-04-06 15:27	File

2

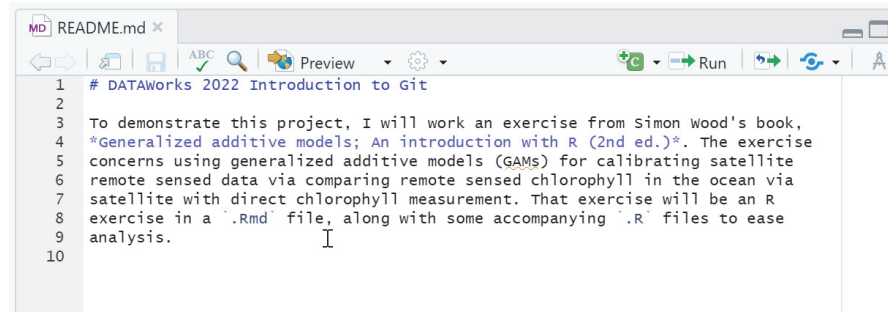
```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (main)
$ git reset --hard
HEAD is now at 0f66f94 structuring project around chlorophyll analysis example
```

3

<input type="checkbox"/> Name	Date modified	Type
 .git	2022-04-07 17:49	File folder
 vignettes	2022-04-07 17:38	File folder
 .gitignore	2022-04-06 15:27	Text Documer
 LICENSE	2022-04-06 15:27	File
 README.md	2022-04-07 17:49	MD File

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

1

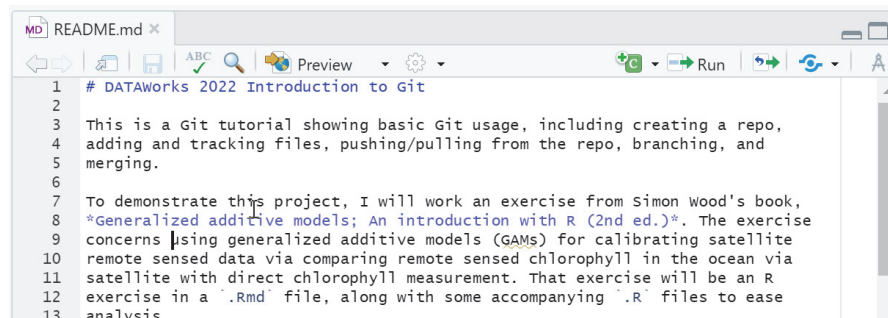


```
MD README.md x
1 # DATAworks 2022 Introduction to Git
2
3 To demonstrate this project, I will work an exercise from Simon Wood's book,
4 *Generalized additive models; An introduction with R (2nd ed.)*. The exercise
5 concerns using generalized additive models (GAMs) for calibrating satellite
6 remote 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.
10
```

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



```
MD README.md x
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
```

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

<input type="checkbox"/> 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...

<input type="checkbox"/> 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 - l),t]*y[(1 + l):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...

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

Using filenames to keep a history...

New report draft X Current Mailbox

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

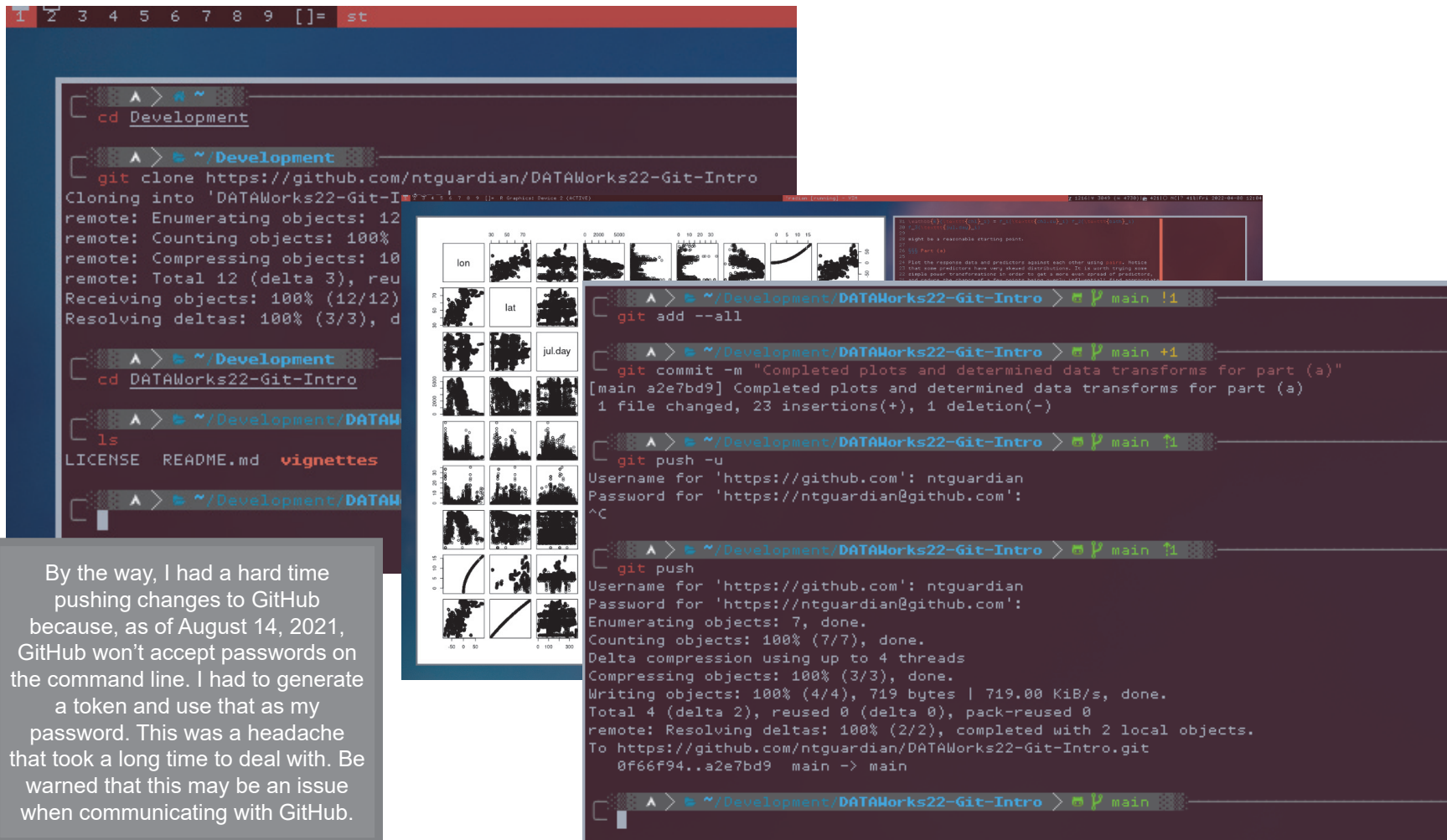
Curtis Miller

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

```
mented in Rcpp for speed
matrix y and the evaluation of the kernel function stored
and returns a vector containing estimated long-run variances
vec_cpp(y, kern_vals, max_l)
R code (slower)
/(1:n, function(t) {
(n - 1), function(l) {
/[1:(n - 1),t]*y[(1 + 1):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.")
```

Commenting out/keeping old useless code...

# A collaborator just did some stuff on their own computer...



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

```
cd Development
```

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

```
cd DATAWorks22-Git-Intro
```

```
ls
LICENSE  README.md  vignettes
```

```
lon
```

```
lat
```

```
jul.day
```

```
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(-)
```

```
git push -u
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
^C
```

```
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
```

```
git push
```

```
git push
```

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

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

The screenshot shows a GitHub repository interface. At the top, there are navigation elements: a dropdown menu for the 'main' branch, '1 branch', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and 'Code'. Below this is a commit history table with the following entries:

File	Commit Message	Time
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 table, the content of the 'README.md' file is displayed. The title is 'DATAWorks 2022 Introduction to Git'. The text below the title reads: 'This is a Git tutorial showing basic Git usage, including creating a repo, adding and tracking files, pushing/pulling'.

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

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

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

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

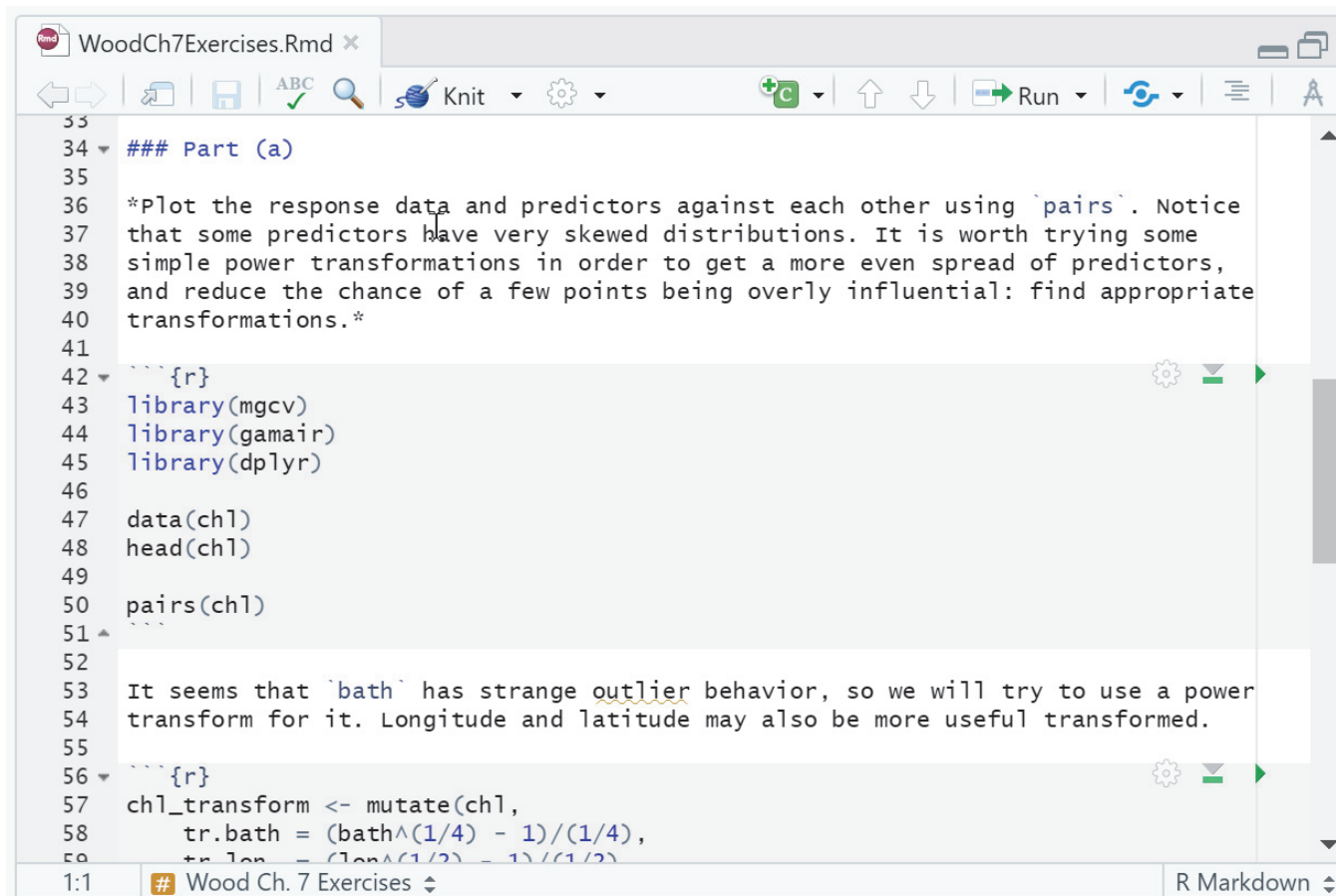
```
MINGW64:/c/Users/cmiller/Development/git/DATAWo
cmiller@CMILLER-LT MINGW64 ~/Development/git
$ git pull
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 4 (delta 2), reused 4 (delta 2)
Unpacking objects: 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



```
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 ch1_transform <- mutate(ch1,
58   tr.bath = (bath^(1/4) - 1)/(1/4),
59   tr.lon = (lon^(1/2) - 1)/(1/2)
60 )
```

1:1 # Wood Ch. 7 Exercises R Markdown

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

**Wood Ch. 7 Exercises**

Curtis Miller  
4/6/2022

**Problem 9**

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

$$\mathbb{E}(chl_i) = f_1(chl.sw_i)f_2(bath_i)f_3(jul.day_i)$$

*might be a reasonable starting point.*

**Part (a)**

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

```
library(mgcv)
## Loading required package: nlme
## This is mgcv 1.8-35. For overview type 'help("mgcv-package")'.
library(gamair)
## Warning: package 'gamair' was built under R version 4.1.3
library(dplyr)
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:nlme':
## collapse
## The following objects are masked from 'package:stats':
## filter, lag
## The following objects are masked from 'package:base':
## intersect, setdiff, setequal, union
data(chl)
head(chl)
##      lon   lat jul.day bath  chl  chl.sw
## 1 -0.0018 60.0160   248  120 1.50 0.131816
## 2 -0.0020 60.4230   267  110 0.90 1.2269796
## 3 -0.0058 72.7410   284  2739 0.40 0.3912244
## 4 -0.0173 50.5817   110  44 0.48 1.7556403
## 5 -0.0227 53.8537   212  4 0.40 2.0605892
## 6 -0.0277 53.8565   212  4 0.60 2.0605892
pairs(chl)
```

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

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

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

The image shows a Microsoft Word window in the background, titled "WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word" by Miller, Curtis. The ribbon is set to "Home" and shows various editing options. In the foreground, a Windows File Explorer window is open to the "vignettes" directory. The address bar shows the path: "git > DATAWorks22-Git-Intro > vignettes". A table of files is displayed:

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

The file "WoodCh7Exercises" is highlighted with a yellow box. The left sidebar of File Explorer shows the navigation pane with "vignettes" selected. The status bar at the bottom indicates "2 items".

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

The screenshot shows a Windows File Explorer window titled 'vignettes' with the address bar showing the path 'git > DATAWorks22-Git-Intro > vignettes'. The file 'WoodCh7Exercises' is highlighted with a yellow box. A yellow callout box with a black border contains the text 'This file should NOT be tracked by Git!' with an arrow pointing to the file. The background shows a Microsoft Word document titled 'WoodCh7Exercises [Read-Only] [Compatibility Mode] - Word' by Miller, Curtis. The document content includes a 'Problem 9' section and R Markdown code at the bottom.

```
library(mgcv)
## Loading required
## This is mgcv 1.8-
library(gamair)
## Warning: package
library(dplyr)
##
## Attaching package
```

Page 1 of 4 626 words

# 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)



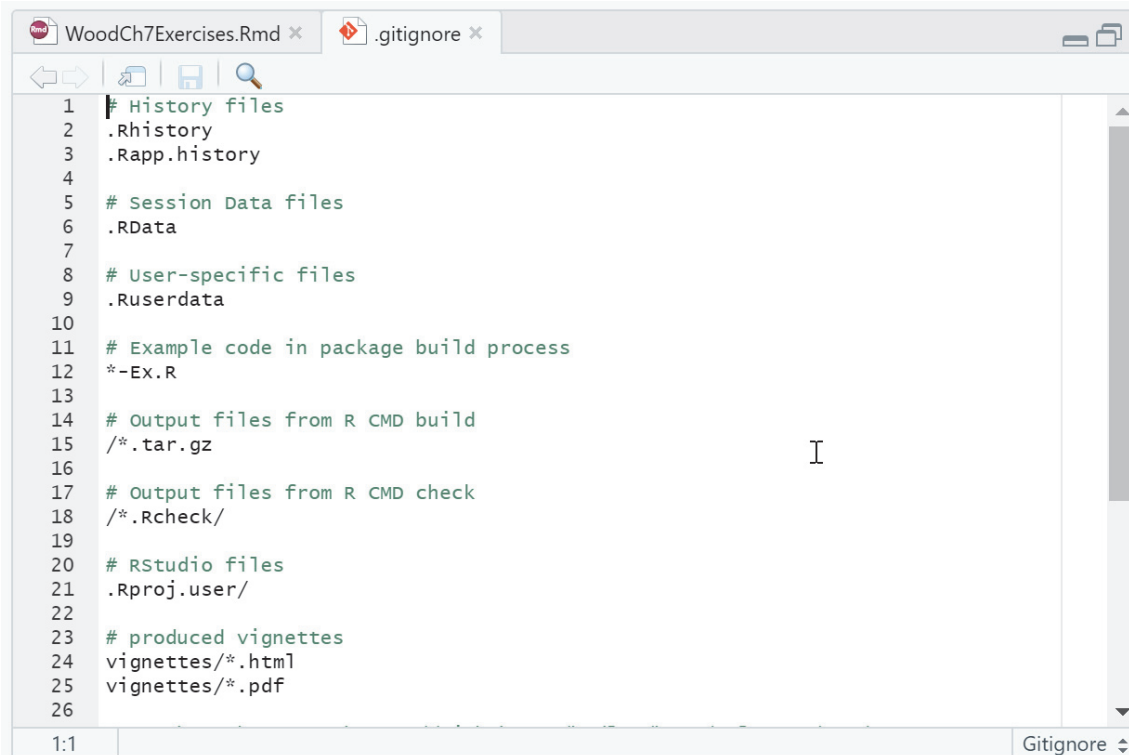
Files generated by other files



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



```
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 image shows a code editor window with a file named `.gitignore` open. The file contains several lines of text, each representing a pattern to ignore. Annotations in yellow boxes with arrows point to specific lines, explaining their meaning:

- Line 1: `# History files` (comment)
- Line 2: `.Rhistory`
- Line 3: `.Rapp.history`
- Line 4: `# Session Data files` (comment)
- Line 5: `.RData` (highlighted) → **Ignore any file ending in .Rdata anywhere in the repo**
- Line 6: `# User-specific files` (comment) → **Comment; understood by a reader, ignored by Git**
- Line 7: `.Ruserdata`
- Line 8: `# Example code files` (comment)
- Line 9: `*-Ex.R` (highlighted) → **Ignore file like "Test-Ex.R" or "my\_func\_Ex.R" anywhere**
- Line 10: `# Output files from R CMD SHLIB` (comment)
- Line 11: `/*.tar.gz` (highlighted) → **In the base directory, ignore files ending in .tar.gz**
- Line 12: `# Output files from R CMD SHLIB` (comment)
- Line 13: `/*.Rcheck/` (highlighted) → **Ignore any directories with .Rcheck in their name**
- Line 14: `# Rstudio files` (comment)
- Line 15: `.Rproj.user/` (highlighted) → **Ignore all files in this directory**
- Line 16: `# Produced vignettes` (comment)
- Line 17: `vignettes/*.html`
- Line 18: `vignettes/*.pdf` (highlighted) → **In the vignettes directory, ignore files ending in .html or .pdf**

The editor window also shows a status bar at the bottom with "1:1" and "Gitignore" with a dropdown arrow.

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
```

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

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

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
```

main ▾ DATAWorks22-Git-Intro / vignettes /

Go to file

Add file ▾

...

Curtis Miller Added knit support for Word files (generated files should be ignored)

0ce384f 2 minutes ago [History](#)

..

WoodCh7Exercises.Rmd

Added knit support for Word files (generated files should be ignored)

2 minutes ago

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

## ... But it did track the other changes

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

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

```
(main)  
s should be ignored  
should be ignored)  
(main)
```

main DATAWorks22-Git-Intro / v

Curtis Miller Added knit support for Word

..

WoodCh7Exercises.Rmd

```
4 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  
6 8 ---  
7 9  
8 10 ```{r setup, include=FALSE}
```

Go to file Add file ...

0ce384f 2 minutes ago History

2 minutes ago

```
a2e7bd9..0ce384f
```

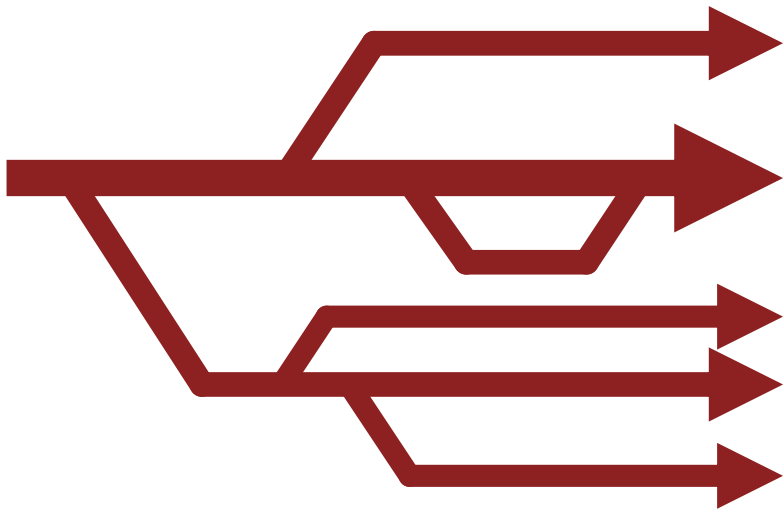
## Basic change tracking commands

Command	Description	Examples
<code>git add</code>	Add changes to (or brand new) files to a staged commit	<code>git add myfile.txt</code> <code>git add --all</code>
<code>git commit</code>	Commit changes	<code>git commit -m "Message"</code>
<code>git pull</code>	For repos hosted remotely (such as GitHub), update repo with changes from host	<code>git pull</code>
<code>git push</code>	For repos hosted remotely (such as GitHub), send changes to host	<code>git push</code>
<code>git rm</code>	Remove a file from tracking	<code>git rm myfile.txt</code>
<code>git reset</code>	Git's "undo" command	<code>git reset --hard</code> # CAUTION: Deletes changes
<code>git diff</code>	Compare commits and branches	<code>git diff HEAD~ HEAD</code> <code>git diff HEAD~ HEAD myfile.txt</code> <code>git diff onebranch otherbranch</code>
<code>git status</code>	Show the status of a repo (such as what will be changed in a commit)	<code>git status</code>
<code>git log</code>	See commit history	<code>git log</code>
<code>git checkout</code>	Retrieve data from commits	<code>git checkout identifier</code>



# Branches

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

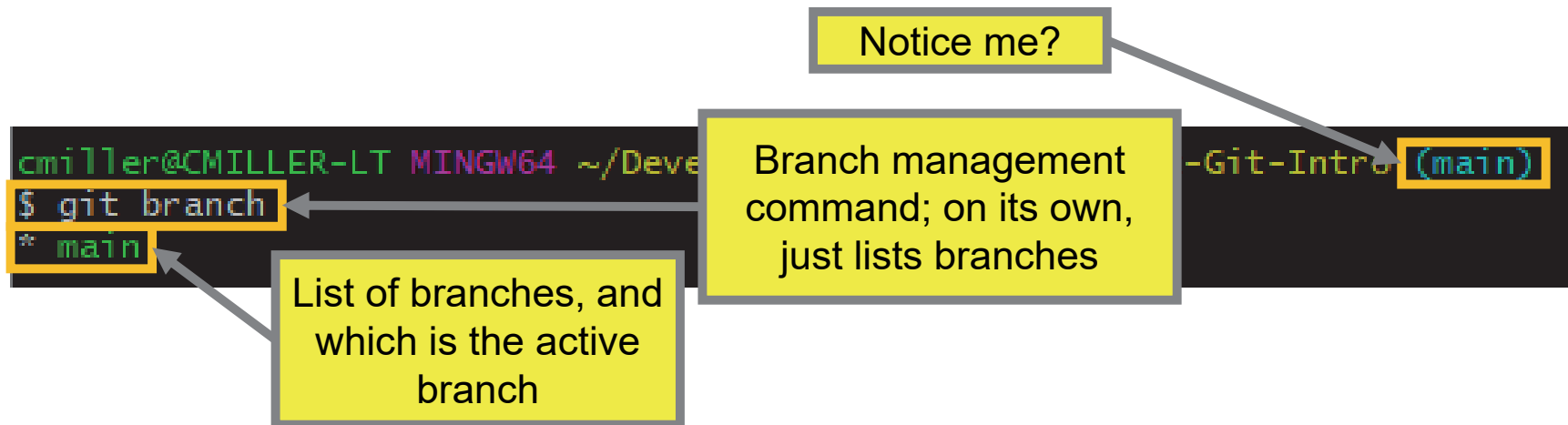


- Commits represent moving forward and backward in time while branches allow for differences between files to exist simultaneously
- Git can track the difference between branches and propagate a change that affects both branches while maintaining what makes them distinct
- Changes made in one branch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

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

Notice me?

List of branches; cmiller is the active branch

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

## Now work primarily with the personal branch

```
WoodCh7Exercises.Rmd
77
78 Let's consider the range of the response variable:
79
80 ```{r}
81 range(ch1_transform$ch1)
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 ```{r}
90 ch1_transform <- mutate(ch1_transform, ch1 = ifelse(ch1 == 0, 0.001, ch1))
91
92
93 Now fit a GAM.
94
95 ```{r}
96 ch1_fit <- gam(ch1 ~
97               s(tr.lon, k = 60, bs = 'cr') +
98               s(tr.lat, k = 60, bs = 'cr') +
99               s(tr.bath, k = 60, bs = 'cr') +
100              s(jul.day, k = 120, bs = 'cr') +
101              s(ch1.sw, k = 60, bs = 'cr'),
102              data = ch1_transform,
103              family = Gamma(link = "log"),
104              method = "REML"
105              )
106
107
108 Sadly the latitude and longitude are not interacted due to time to compute
109 solution being too long.
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
117
77:1 Part (b) ↕
```

```
MINGW64:/c/Users/cmiller/Development/git/DATAWorks22-Git-Intro
[cmiller ca00e1b] Attempting part (b) using log link/gamma family
1 file changed, 28 insertions(+), 3 deletions(-)

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

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

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAWorks22-Git-Intro (cmiller)
$ git checkout HEAD~ vignettes/WoodCh7Exercises.Rmd
Updated 1 path from fcl1a383

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

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

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

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

```
^ > ~/Development/DATAWorks22-Git-Intro > P main
git pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (??/??), done.
remote: Total 5 (de
Unpacking objects:
From https://github
a2e7bd9..0ce384f
Updating a2e7bd9..0
Fast-forward
.gitignore
vignettes/WoodCh7E
2 files changed, 4
  2 author: "Curtis Miller"
  3 date: "4/6/2022"
  4 output:
  5 word_document: default
  6 html_document: default
  7 ---
  8
  9 ```{r setup, include=FALSE}
10 knitr::opts_chunk$set(echo = TRU
11 ```
12
13 $$ Problem 6
14
15 The data frame ipo contains data
16 of 'Initial Public Offerings' (I
17 between 1960 and 2002. IPOs are
18 ownership of the company is sold
19 capital. Interest focuses on exp
20 have on numbers of IPOs per mont
21 variables:
22
23 ir: the average initial return f
24 percentage difference between th
25 after the first day of trading i
26 offer price undervalues the comp
27 attention to this when deciding
28
29 dp: is the average percentage di
30
^ > ~/Development/DATAWorks22-Git-Intro > P main
git checkout -b probs6n8
Switched to a new branch 'probs6n8'
^ > ~/Development/DATAWorks22-Git-Intro > P probs6n8
vim vignettes/WoodCh7Exercises.Rmd
^ > ~/Development/DATAWorks22-Git-Intro > P probs6n8 !1
git add --all
^ > ~/Development/DATAWorks22-Git-Intro > P probs6n8 +1
git commit -m "Adding problem text for probs 6 and 8"
[probs6n8 f3c3b46] Adding problem text for probs 6 and 8
1 file changed, 96 insertions(+)
^ > ~/Development/DATAWorks22-Git-Intro > 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
^ > ~/Development/DATAWorks22-Git-Intro > 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
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/
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 ps 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 ps 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

```
A > ~/Development/DATAworks22-Git-Intro > git ps 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 push --set-upstream origin probs6n8
Username for 'https://github.com': ntguardian
Password for 'https://ntguardian@github.com':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
```

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

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

## We should sync our local copy of the repo now

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

    git pull <remote> <branch>

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

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

## Read the resulting messages

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

Git is telling us about the branch that was created

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

We can use the following command to track our branch remotely

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

We can use the following command to track our branch remotely

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

Setting the location where the branch is stored

To see the new branch, use `git checkout`

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

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

To see the new branch, use `git checkout`

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

Name the branch to checkout

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

To compare branches, use `git diff`

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

To compare branches, use `git diff`

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

Other branch to  
be compared

One branch to  
be compared

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

## We can now compare what's in the branches

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

+## Problem 6
+
+*The data frame 'ipo'
+of 'Initial Public Offerings'
+between 1960 and 2000
```

```
### 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(ch1_transform$chl)
# Your code here
```

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

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

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

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (probs6n8)
$ git diff --name-only cmiller probs6n8
vignettes/WoodCh7Exercises.Rmd
```

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

```
cmiller@CMILLER-IT MINGW64 ~/Development/git/
$ git diff --name-only cmiller-probs6n8
vignettes/WoodCh7Exercises.Rmd
```

Option to show only filenames, not all differences

Which files differ between the two branches

Now go back to our personal branch

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

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

## A few more edits...

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

cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (cmiller)
$ git commit -m "Attempting part (c) but did not complete deviance computation on 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 ~/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 0ce584f..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 used
Compressing objects: 100%
Writing objects: 100%
Total 8 (delta 4), re
remote: Resolving del
To https://github.com
0ce384f..0569dd3
```



The screenshot shows a GitHub repository interface for 'DATAWorks 2022 Introduction to Git'. The commit history table is as follows:

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

The branch switcher shows the following branches:

- main (default)
- cmiller
- probs6n8

## Our log shows a more interesting structure now

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

For a more interesting visualization

Note the fork...

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

```
188 Now fit a GAM.
189
190 ▾ ````{r}
191 ✖ <<<<<<< 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
```

- 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(ch1 ~ s(lat) + s(lon), data = ch1, family = gaussian)
175 ^
176
```

```
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 <<<<<<<<< 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 <<<<<<< 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
207 ~~~~{r}
208
209 ~~~~ Part (c)
210
211 *In this sort of modeling the aim is to improve on simply using
```

Lines from this branch

Lines from incoming branch

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

... and we can see where

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

```
188 Now fit a GAM.
189
190 }
191 <<<<<<<<<< HEAD
192 nam(chl ~ s(lat) + s(lon), data = chl, family = gaussian)
193 =====
194 chl_model <- chl
195     s(lon.bath, k = 60,
196     s(jul.day, 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 >>>>>>>>>> main
207
208
209 ### Part (c)
210
211 *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 ```{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
204
205 ### Part (c)
206
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 chl_model <- chl ~
```

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (probs6n8|MERGING)  
$ git add --all
```

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (probs6n8|MERGING)  
$ git commit -m "Conflict resolved by using code from main (better solution)"  
[probs6n8 cf0ee86] Conflict resolved by using code from main (better solution)
```

```
cmiller@CMILLER-LT MINGW64 ~/Development/git/DATAworks22-Git-Intro (probs6n8)  
$ git push
```

```
Enumerating objects: 14, done.  
Counting objects: 100% (14/14), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (8/8), 1.33 KiB | 273.00 KiB/s, done.  
Total 8 (delta 4), reused 0 (delta 0), pack-reused 0  
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.  
To https://github.com/ntguardian/DATAworks22-Git-Intro.git  
f3c3b46..cf0ee86 probs6n8 -> probs6n8
```

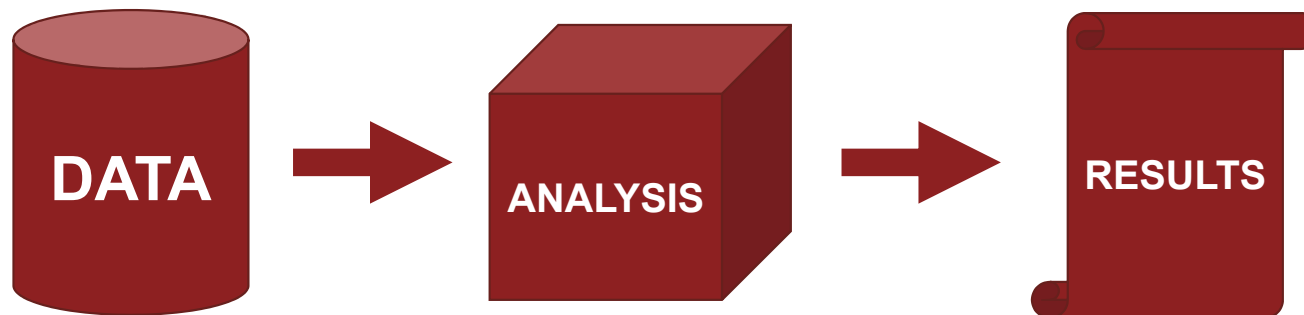
## Basic branch management commands

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

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

# BACKUP

# Navigating your computer using Bash

| Command   | Description  | Examples   |
|-----------|--|--|
| ls        | List information about files or directories                    | ls # List files<br>ls -lh # List files with description<br>ls -lha # List all files with description   |
| cd        | Change the current working directory                           | cd "My Documents" # Go to My Documents folder<br>cd .. # Go to parent directory<br>cd ~/ # Go to home directory<br>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"<br>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<br>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<br>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><br>04-2022 | <b>2. REPORT TYPE</b><br>Final | <b>3. DATES COVERED</b> |                             |
|                                  |                                | <b>START DATE</b>       | <b>END DATE</b><br>Apr 2022 |

**4. TITLE AND SUBTITLE**  
DATAWorks 2022 - Introduction to git

|  |                         |                                   |
|--|-------------------------|-----------------------------------|
| <b>5a. CONTRACT NUMBER</b><br>HQ0034-19-D-0001 | <b>5b. GRANT NUMBER</b> | <b>5c. PROGRAM ELEMENT NUMBER</b> |
|--|-------------------------|-----------------------------------|

|   |                                  |                             |
|---|----------------------------------|-----------------------------|
| <b>5d. PROJECT NUMBER</b><br>BD-09-2299 | <b>5e. TASK NUMBER</b><br>229990 | <b>5f. WORK UNIT NUMBER</b> |
|---|----------------------------------|-----------------------------|

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

|  |  |
|--|--|
| <b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b><br>Institute for Defense Analyses<br>730 East Glebe Road<br>Alexandria, Virginia 22305 | <b>8. PERFORMING ORGANIZATION REPORT NUMBER</b><br>NS D-33021<br>H 2022-000108 |
|--|--|

|  |  |  |
|--|--|--|
| <b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b><br>Director, Operational Test and Evaluation<br>1700 Defense Pentagon<br>Washington, DC 20301 | <b>10. SPONSOR/MONITOR'S ACRONYM(S)</b><br><br>DOT&E | <b>11. SPONSOR/MONITOR'S REPORT NUMBER</b> |
|--|--|--|

**12. DISTRIBUTION/AVAILABILITY STATEMENT**  
This publication has not been approved by the sponsor for distribution and release. Reproduction or use of this material is not authorized without prior permission from the responsible IDA Division Director.

**13. SUPPLEMENTARY NOTES**

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

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

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

|  |                                    |                                     |  |                                       |
|--|------------------------------------|-------------------------------------|--|---------------------------------------|
| <b>16. SECURITY CLASSIFICATION OF:</b> |                                    |                                     | <b>17. LIMITATION OF ABSTRACT</b><br>SAR | <b>18. NUMBER OF PAGES</b><br><br>143 |
| <b>a. REPORT</b><br>Unclassified       | <b>b. ABSTRACT</b><br>Unclassified | <b>c. THIS PAGE</b><br>Unclassified |  |                                       |

|   |  |
|---|--|
| <b>19a. NAME OF RESPONSIBLE PERSON</b><br>John T. Haman | <b>19b. PHONE NUMBER</b><br>703-845-2132 |
|---|--|