Introduction to Version Control with git and github

Bryan Scott

LSSTC Data Science Fellowship Program Pre-Orientation

May 9, 2025

Some git Resources

Pro Git book available at git-scm.com/docs

and

Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer by Mariot Tsitoara

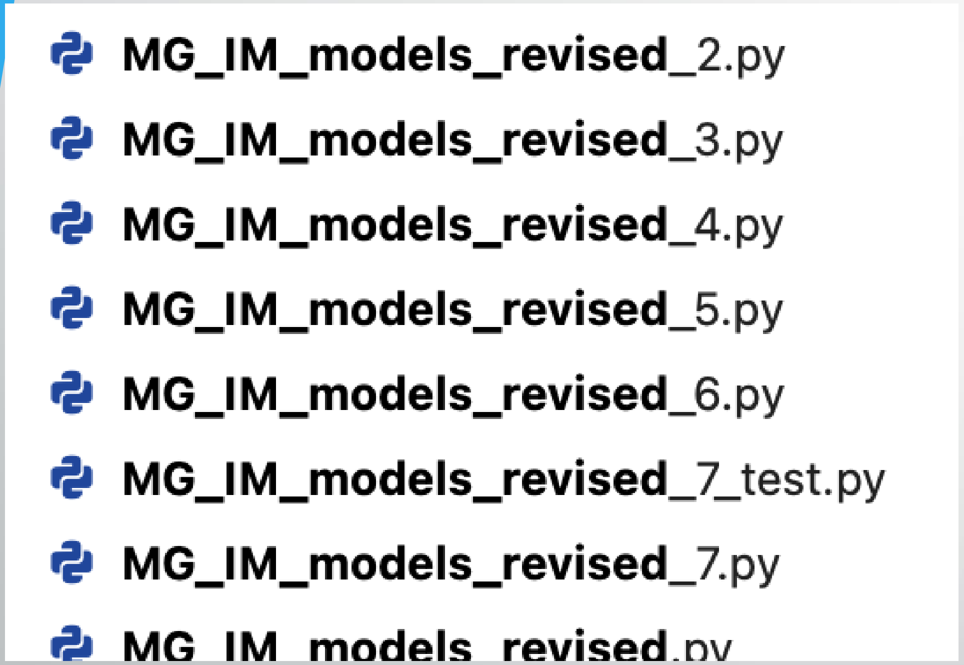
What makes learning git difficult?

What makes learning git difficult?

Let's be real – development workflows...

Brilliant idea! › Prototyping › Iterative Fixes › Oh no!

Problems in typical development workflows…


MG_IM_models_revised_2.py
MG_IM_models_revised_3.py
MG_IM_models_revised_4.py
MG_IM_models_revised_5.py
MG_IM_models_revised_6.py
MG_IM_models_revised_7_test.py
MG_IM_models_revised_7.py
MG_IM_models_revised.py

Paper_draft_final.tex
Paper_draft_final_revised.tex
Paper_draft_final_revised_final.tex
Paper_draft_final_revised_final_submission.tex
Paper_draft_final_revised_final_submission_with_comments.tex
Paper_draft_final_revised_final_submission_with_comments_and_responses_draft.tex
(and so on….)

What is version control?

Version control is a *system* for creating a *reproducible* record of changes to a *project*.

The key idea is that there is only **one** project. *No matter* how big or complicated that project is.

Git is a *distributed* version control system – there is no authoritative master repository and each copy is as valid as any other.

The usefulness of software repositories for different kinds of developers...

For "small projects", tracking changes with git (or another VCS) and making regular commits makes your work more reproducible and gives you a "global undo" if you introduce a catastrophic error into your code.

For "large projects", a distributed VCS can allow multiple developers to work independently and synchronously. Changes are merged later (and conflicts resolved appropriately).

Introduction to git

Distributed Version Control

The Distributed Version Control Model (Git)

A software repository is a place for storing software and metadata about it.

In a distributed version control system, each developer has a copy of the repository – while it may be convenient to choose a "central" copy as a way of aiding collaboration – there is no authoritative version of the repository.

Git is an example of a distributed version control system.

Some git terminology

Repository – a special kind of directory containing software and metadata about the software, in particular a record of files and changes to them.

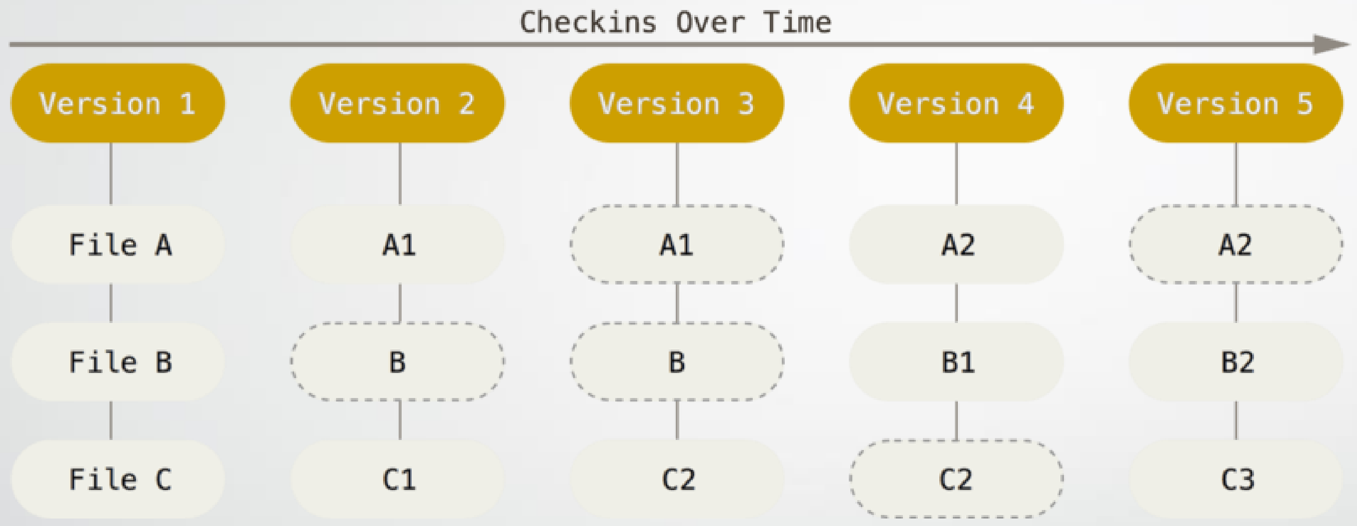
Commit – a "snapshot" of the state of a project at a specific time.

Branch – a set of commits ordered in a linear series

Merge – an action to take two branches and combine them into one

Push/pull - the action we take to move commits from one repository to another

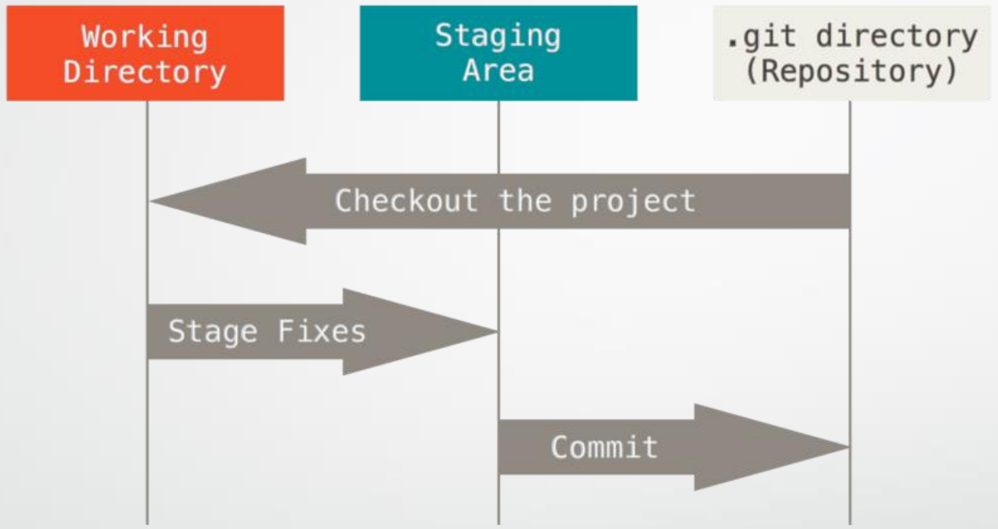
How does Git work?



Git stores snapshots of the project over time. If a new file is added or an old file changed, the new version will be stored in the next snapshot.

But if a file is unchanged, git simply links to the version stored in the previous snapshot.

Git States



Files can be in one of three states. *Modified, staged, and committed.* Files move between the three stages as you work. After editing a file it is *modified.* You then *stage* the file which tells git to include it in the next snapshot. You then *commit* the changes (save a snapshot of the project).

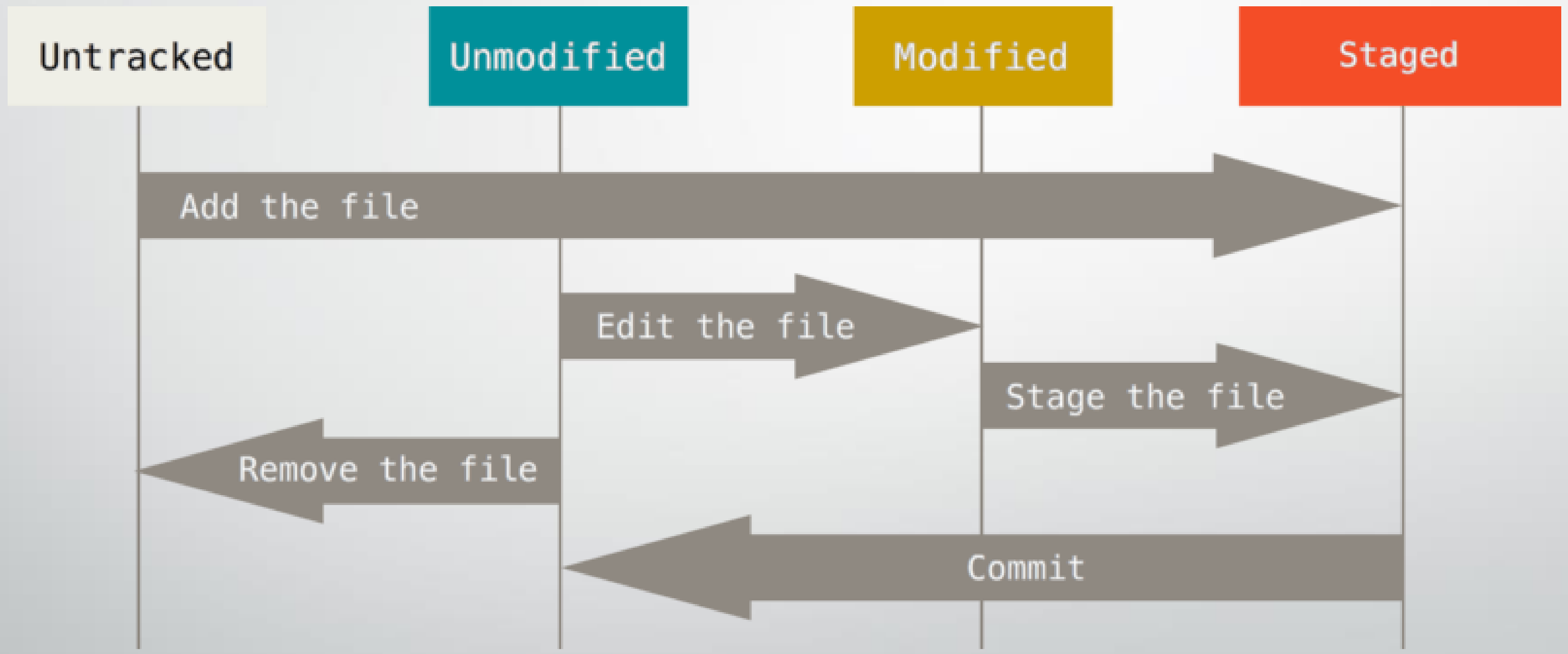
Git Usage – Changes to the Repository

Files in your project can be in one of two states – *tracked* or *untracked*

Git tracks all files that were in the last snapshot (*commit*) or that have been created and staged with the git *add* command.

To add changed files to the repository, you need to stage them with *add* and then snapshot with *commit*

Git File Cycle



Introduction to Github

+ branches, merges, and more

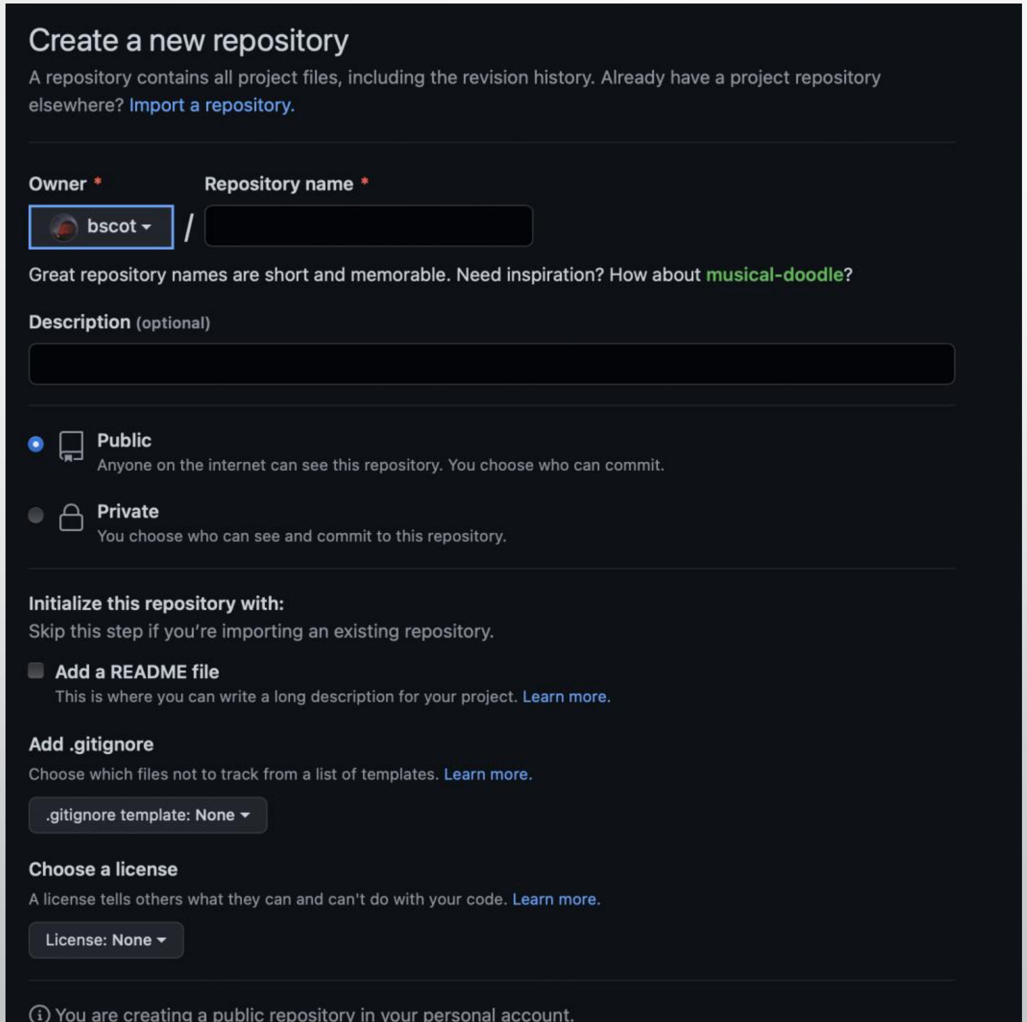
Git as a distributed version control system

Remember, git is a distributed version control system. Up until now, we have considered only a local copy of the software repository on our machine.

But there's nothing special about our copy – we can have n copies of our project that are shared across many developers. We'll need to figure out how to manage n copies (not easy!), but the power of being able to share and collaborate is obvious!

It would help if we had a convenient place to store a copy that we'll all work from – this is where github or gitlab come in.

Create a remote repository on github



Create a new repository

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

Owner * Repository name *

bscot /

Great repository names are short and memorable. Need inspiration? How about musical-doodle?

Description (optional)

[✓ 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: None

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

License: None

You are creating a public repository in your personal account.

Two ways to create a Repository

We can create a local repository by navigating to a directory and typing

```
git init
```

Or we can clone a repository from remote by typing

```
git clone [link to remote]
```

Git branches

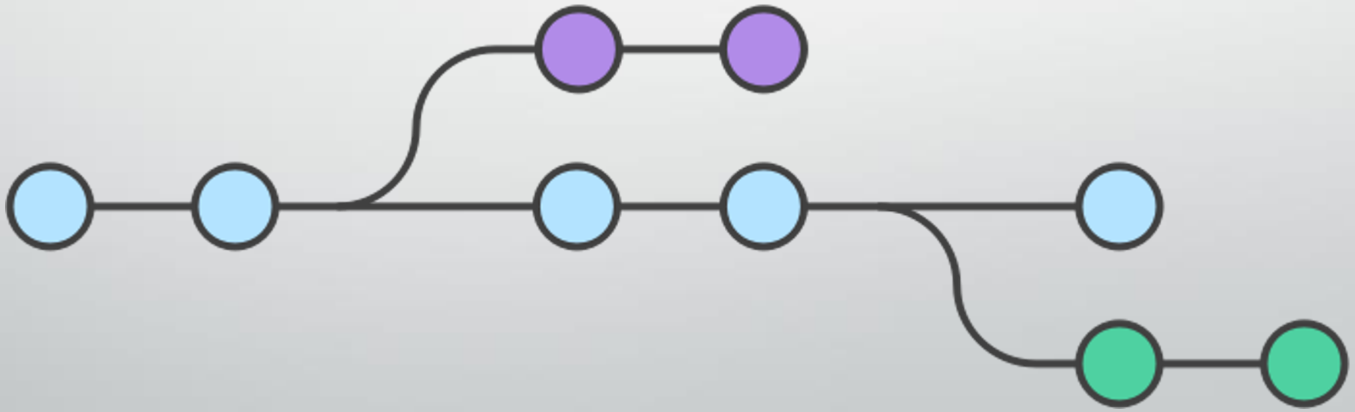
Let's say we want to add a new feature to our code. We don't want to impact our previous version with bugs that the new feature introduces.

We could copy the project and work on a whole new copy, perhaps, project_revised.py would be a new file in the copy. Or we could branch the project – recognizing we're still working on the same project, just adding something to it.

To branch our project

The branch command is

```
git branch [branch_name]
```



Giving branch the –d flag will delete the referenced branch

Linking the local and remote repositories

Once we have a remote repository, we can link the local to it with git *remote* add [name] [link]. By convention, origin or upstream are used as the name of the remote.

```
Git remote add origin [link to your fork of the DSFP repo]
```

You will sometimes see this command – this sends your code to the remote repository

```
git push origin main
```

Where origin is the [remote name] and [main] is the branch we want to push.

Caution: the command git push origin *master* was the default until recently. Main has replaced it as the preferred/default name for your production branch.

Pushing branches to remote – Pull Requests

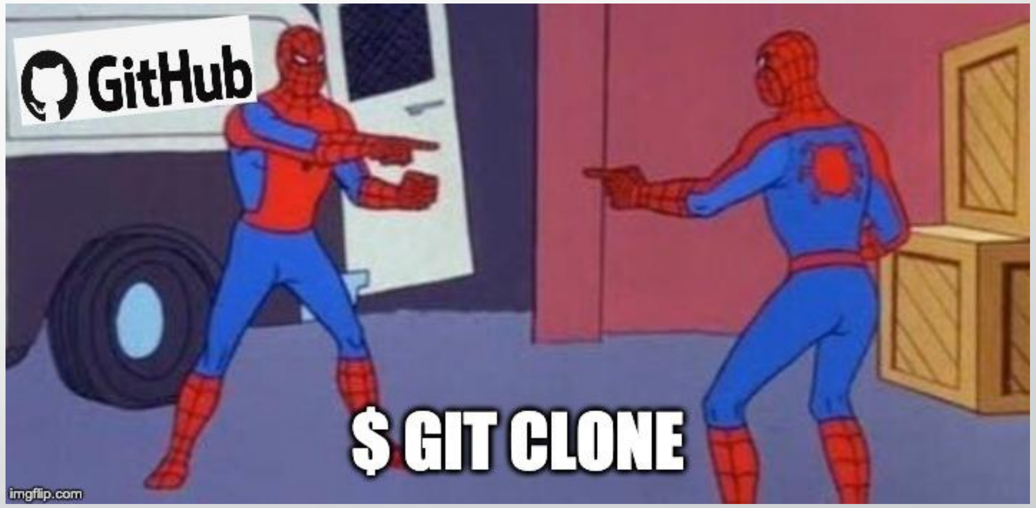
The output of git pushing a branch to the remote repository will either perform a merge automatically or you'll need to create a pull request.

If there are conflicts, it is sometimes possible to resolve the conflicts through the github GUI. Typically you will need to resolve it locally and push the new version.

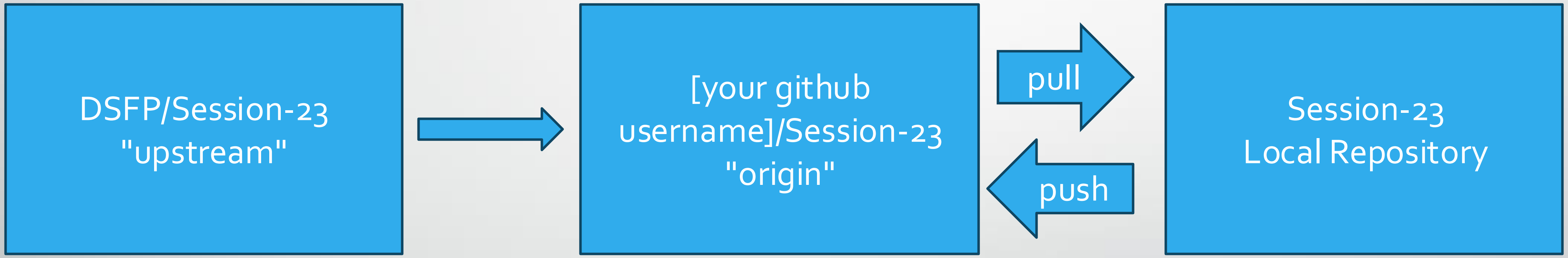
The terminology here is a little weird and confused me for a long time. Push and pull are just opposite actions based on my perspective – am I copying into my branch or copying out of my branch. Remember: git is distributed, so relationships between repos are symmetric.

Putting this into practice

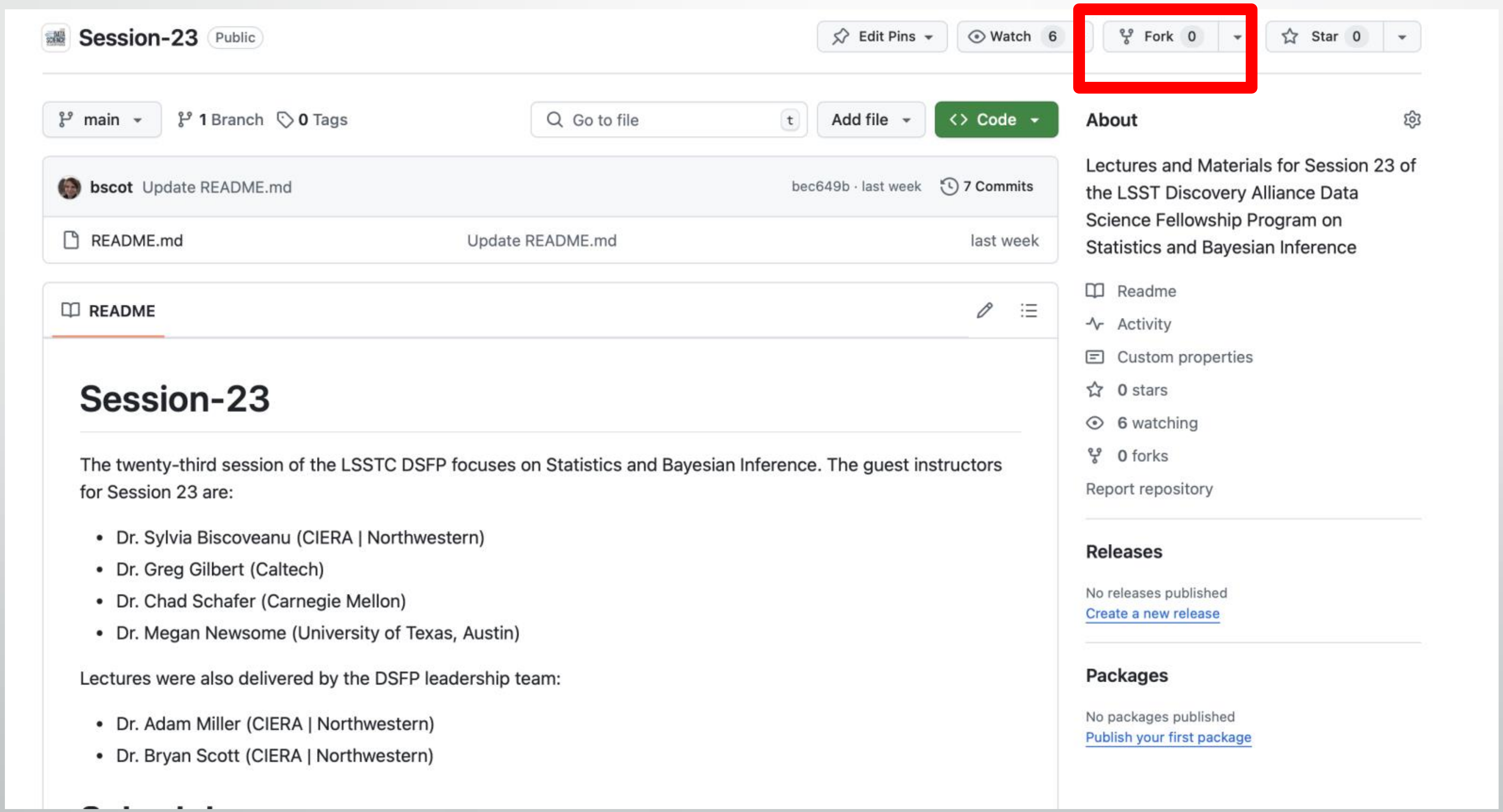
We will now go through forking and cloning the DSFP Repository to create a local version on your machine.



What we want to do



Fork the Session-23 Repo



bscot Update README.md bec649b · last week 7 Commits

README.md Update README.md last week

Session-23

The twenty-third session of the LSSTC DSFP focuses on Statistics and Bayesian Inference. The guest instructors for Session 23 are:

- Dr. Sylvia Biscoveanu (CIERA | Northwestern)
- Dr. Greg Gilbert (Caltech)
- Dr. Chad Schafer (Carnegie Mellon)
- Dr. Megan Newsome (University of Texas, Austin)

Lectures were also delivered by the DSFP leadership team:

- Dr. Adam Miller (CIERA | Northwestern)
- Dr. Bryan Scott (CIERA | Northwestern)

About

Lectures and Materials for Session 23 of the LSST Discovery Alliance Data Science Fellowship Program on Statistics and Bayesian Inference

Readme
Activity
Custom properties
0 stars
6 watching
0 forks
Report repository

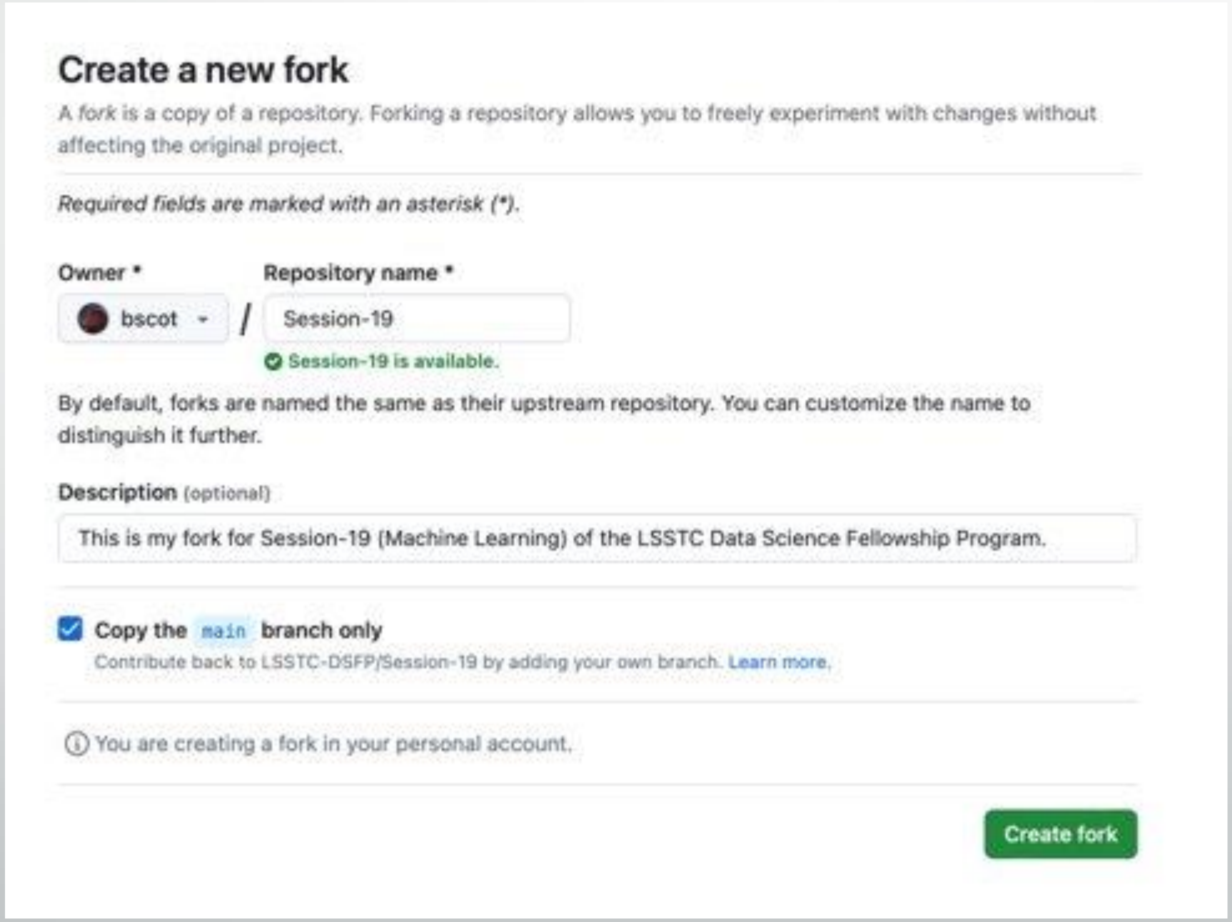
Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Fork the Session-23 Repo



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * bscot / **Repository name *** Session-19

Session-19 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

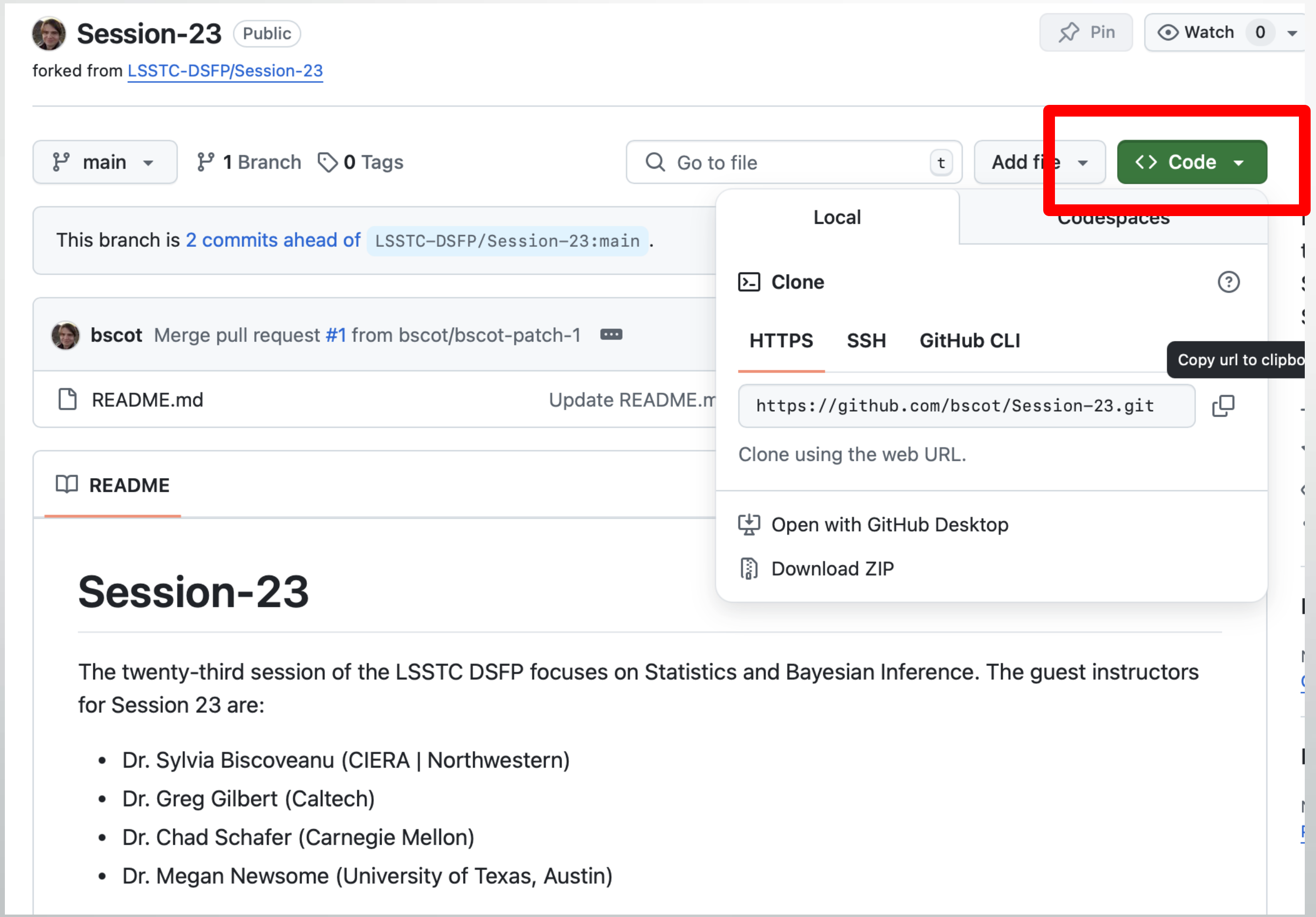
This is my fork for Session-19 (Machine Learning) of the LSSTC Data Science Fellowship Program.

[✓] **Copy the `main` branch only**

Contribute back to LSSTC-DSFP/Session-19 by adding your own branch. Learn more.

ⓘ You are creating a fork in your personal account.

Now clone the repository locally



forked from LSSTC-DSFP/Session-23

This branch is 2 commits ahead of `LSSTC-DSFP/Session-23:main` .

bscot Merge pull request #1 from bscot/bscot-patch-1

README.md Update README.m

```
https://github.com/bscot/Session-23.git
```

Clone using the web URL.

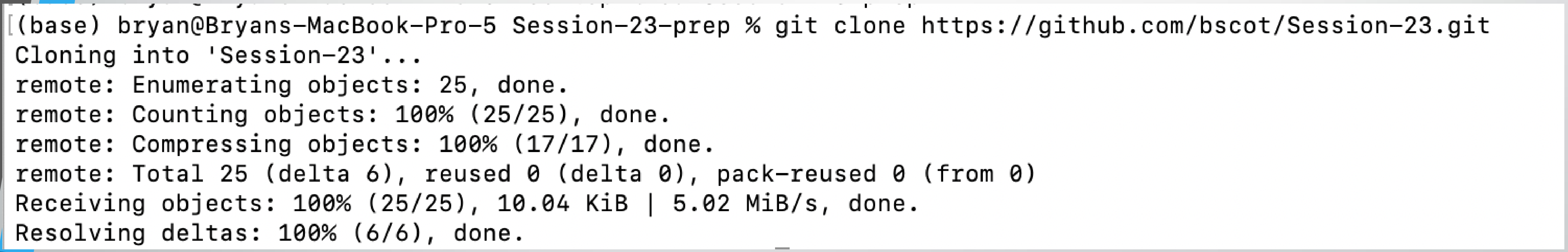
Session-23

The twenty-third session of the LSSTC DSFP focuses on Statistics and Bayesian Inference. The guest instructors for Session 23 are:

- Dr. Sylvia Biscoveanu (CIERA | Northwestern)
- Dr. Greg Gilbert (Caltech)
- Dr. Chad Schafer (Carnegie Mellon)
- Dr. Megan Newsome (University of Texas, Austin)

Now clone the repository locally

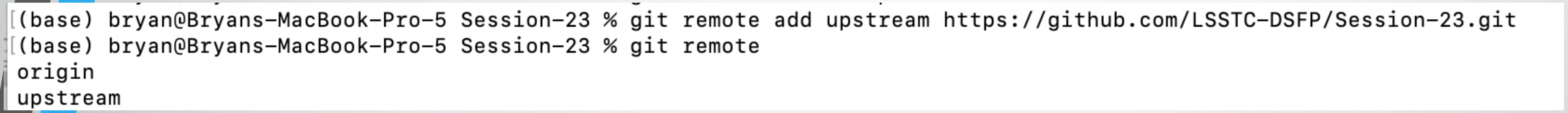
```
git clone [link you copied]
```



```
(base) bryan@Bryans-MacBook-Pro-5 Session-23-prep % git clone https://github.com/bscot/Session-23.git
Cloning into 'Session-23'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 25 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (25/25), 10.04 KiB | 5.02 MiB/s, done.
Resolving deltas: 100% (6/6), done.
```

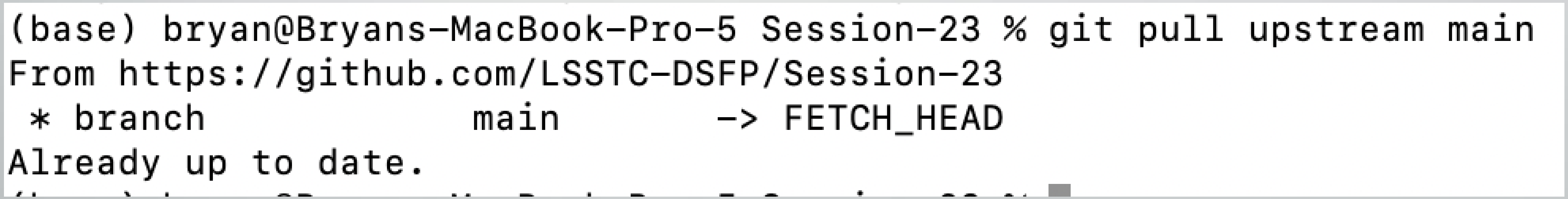
Finally, let's link this to the 'official' Session-23 Repo

```
git remote add upstream https://github.com/LSSTC-DSFP/Session-23.git
```



```
(base) bryan@Bryans-MacBook-Pro-5 Session-23 % git remote add upstream https://github.com/LSSTC-DSFP/Session-23.git
(base) bryan@Bryans-MacBook-Pro-5 Session-23 % git remote
origin
upstream
```

```
git pull upstream main
```



```
(base) bryan@Bryans-MacBook-Pro-5 Session-23 % git pull upstream main
From https://github.com/LSSTC-DSFP/Session-23
 * branch            main       -> FETCH_HEAD
Already up to date.
```

Using git in the DSFP

Before each day, pull from the upstream repository to get the day's materials (and any other new material from the previous day)

- You can either pull from upstream directly ("git pull upstream main") [you'll then need to push to origin ("git push origin main") to bring everything up to date.
- Or use the github GUI to bring your origin repository up to date, then pull origin. [preferred]

At the end of each day (or more frequently), push your changes/saved work to origin

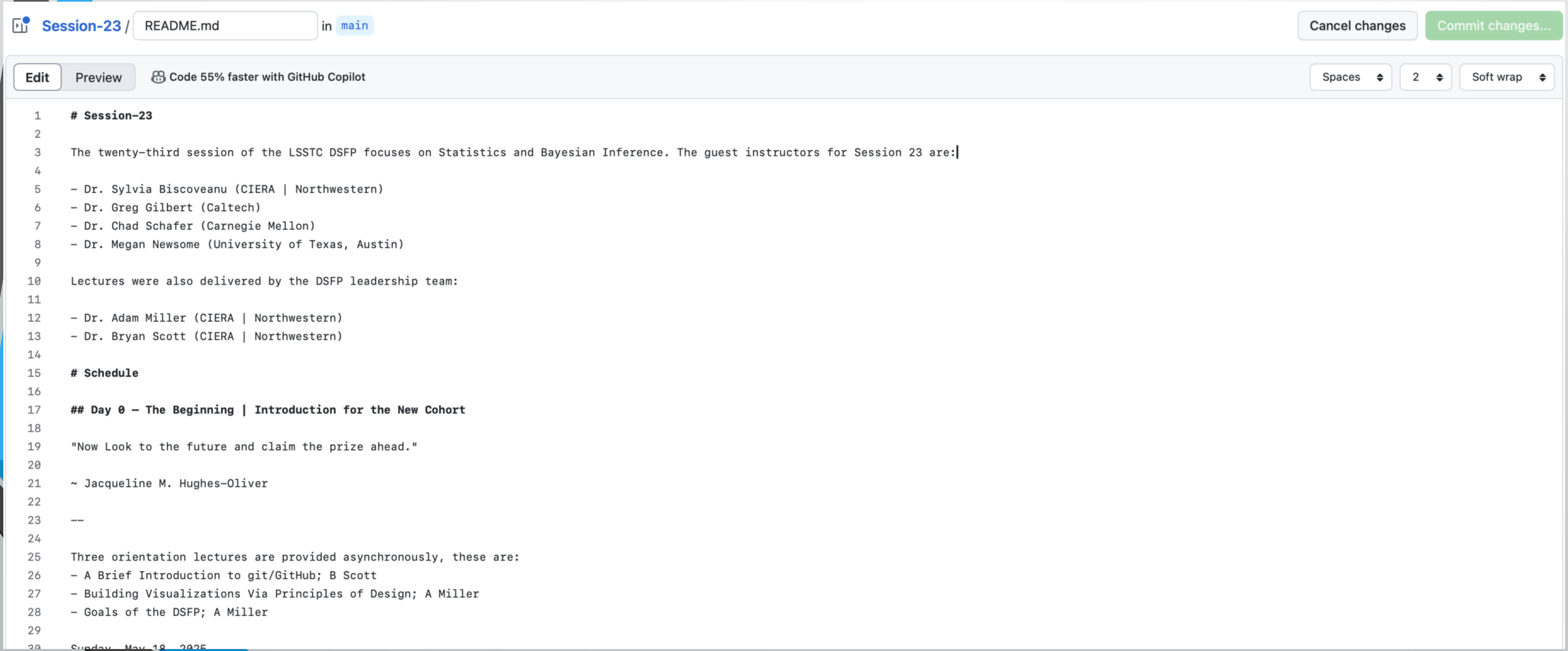
- "git push origin main"
- You may have merge conflicts – if two commits differ on their version of a line of a code – we'll discuss how to handle merge conflicts in detail at a future session.

Bonus: Pull Requests

For this next part, we'll put you pairs in breakout rooms.

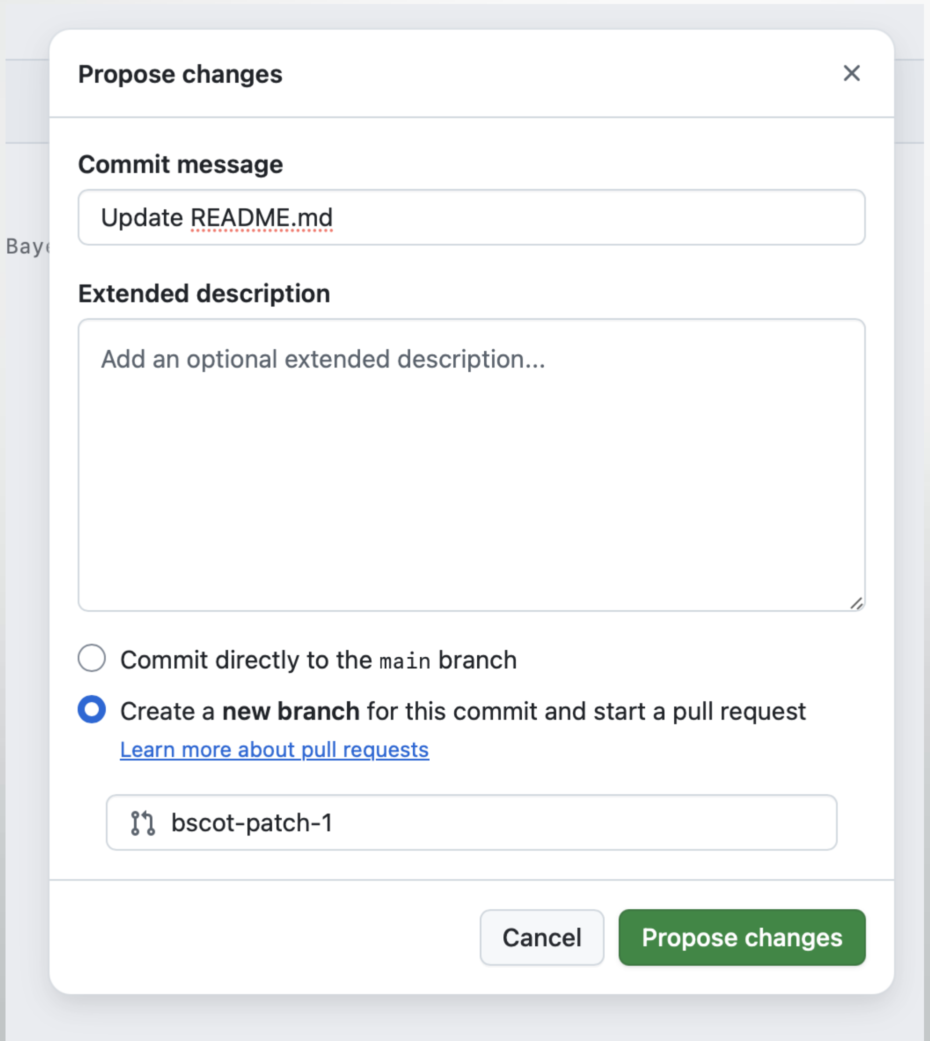
- Make an edit to the README.md file to add your name as an author of the repository
- Once you've done that, hit commit, open a pull request in the original repository to merge in the changed README.md
- The other partner should then review the pull request to confirm they agree with the changes and approve or deny the pull.

Edit the Readme (in your fork)


Session-23 / README.md in main
Cancel changes
Commit changes...
Edit
Preview
Code 55% faster with GitHub Copilot
Spaces
2
Soft wrap

```
# Session-23

The twenty-third session of the LSSTC DSFP focuses on Statistics and Bayesian Inference. The guest instructors for Session 23 are:

- Dr. Sylvia Biscoveanu (CIERA | Northwestern)
- Dr. Greg Gilbert (Caltech)
- Dr. Chad Schafer (Carnegie Mellon)
- Dr. Megan Newsome (University of Texas, Austin)

Lectures were also delivered by the DSFP leadership team:

- Dr. Adam Miller (CIERA | Northwestern)
- Dr. Bryan Scott (CIERA | Northwestern)

# Schedule

## Day 0 – The Beginning | Introduction for the New Cohort

"Now Look to the future and claim the prize ahead."

~ Jacqueline M. Hughes-Oliver

--

Three orientation lectures are provided asynchronously, these are:
- A Brief Introduction to git/GitHub; B Scott
- Building Visualizations Via Principles of Design; A Miller
- Goals of the DSFP; A Miller

Sunday, May 18, 2025
```

Pull Requests



Bay

Propose changes

Commit message

Update README.md

Extended description

Add an optional extended description...

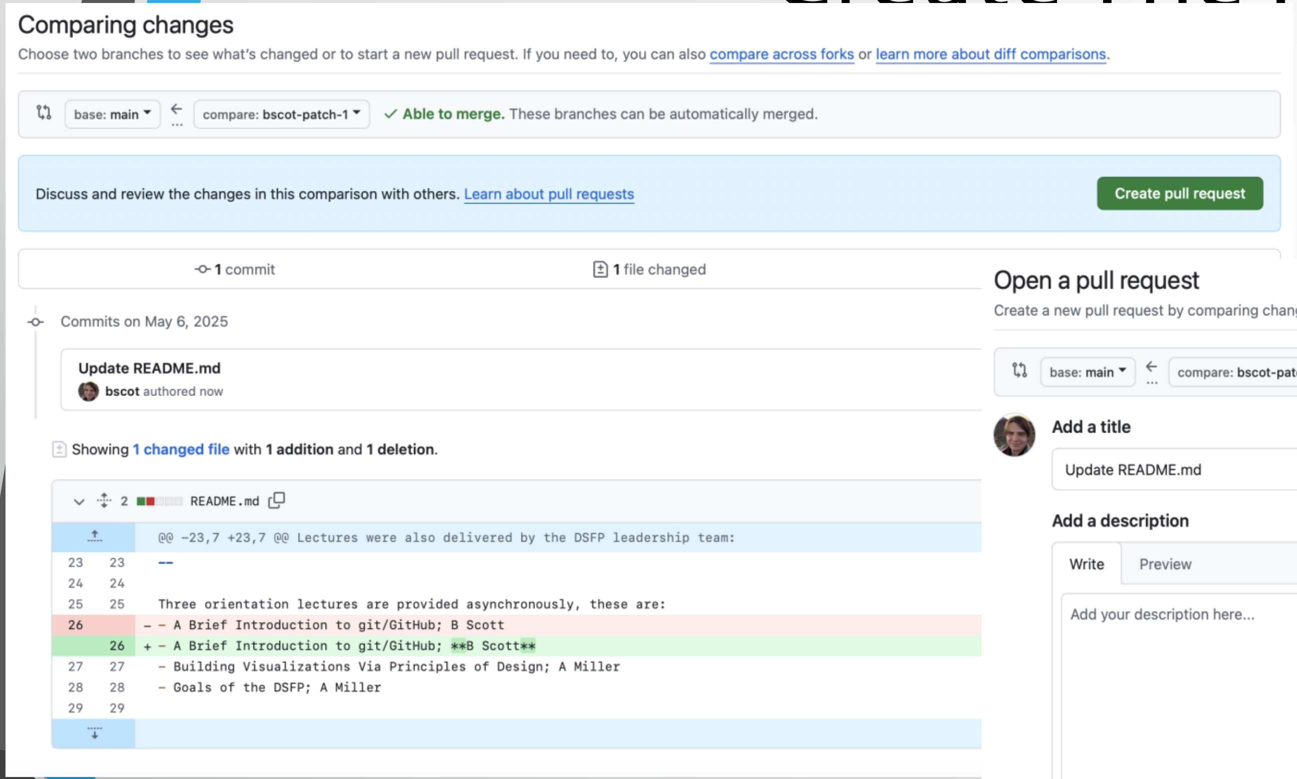
( ) Commit directly to the `main` branch

[✓ Create a **new branch** for this commit and start a pull request]

Learn more about pull requests

bscot-patch-1

Create The Pull Request



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base: main ← compare: bscot-patch-1 ✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. Learn about pull requests

1 commit | 1 file changed

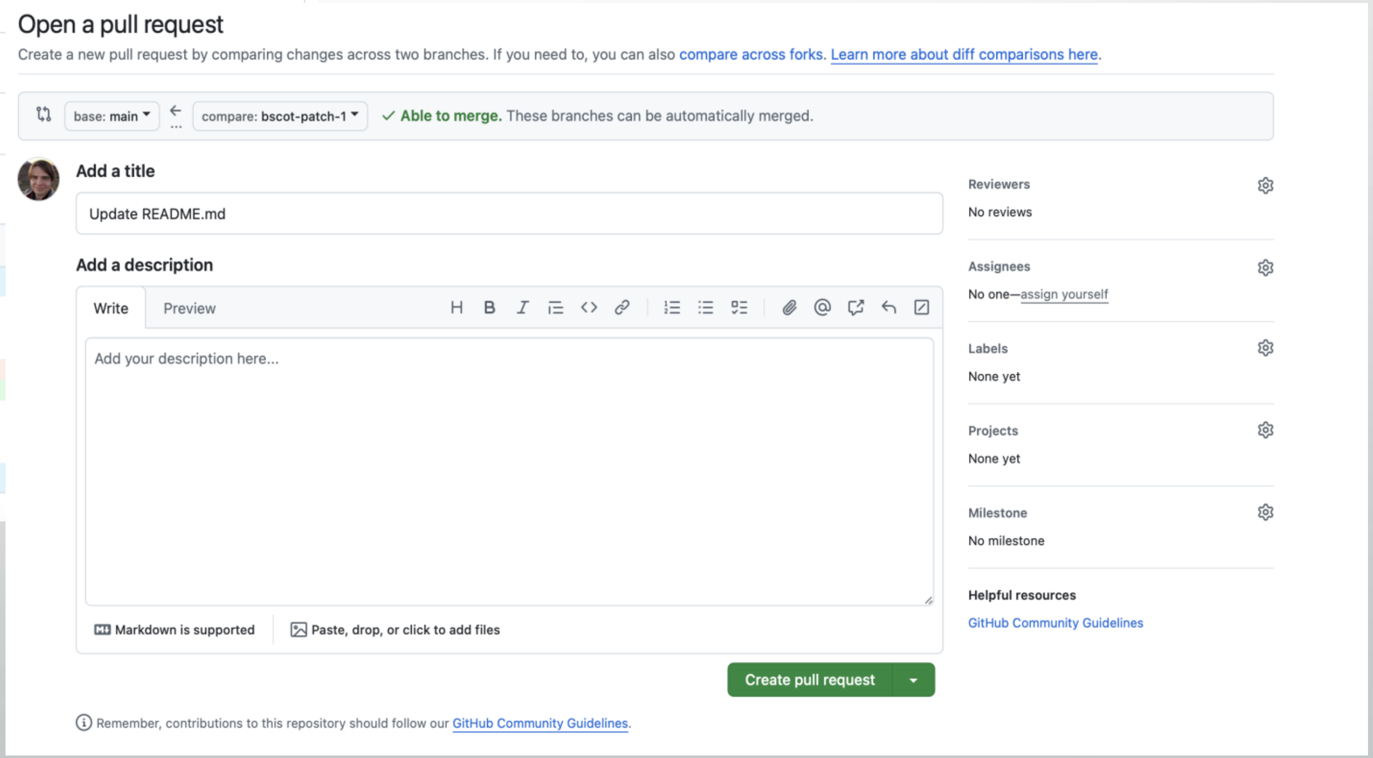
Commits on May 6, 2025

Update README.md
bscot authored now

Showing 1 changed file with 1 addition and 1 deletion.

2 README.md

```
@@ -23,7 +23,7 @@ Lectures were also delivered by the DSFP leadership team:
 --

 Three orientation lectures are provided asynchronously, these are:
- - A Brief Introduction to git/GitHub; B Scott
+ - A Brief Introduction to git/GitHub; **B Scott**
 - Building Visualizations Via Principles of Design; A Miller
 - Goals of the DSFP; A Miller

```



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

base: main ← compare: bscot-patch-1 ✓ Able to merge. These branches can be automatically merged.

Add a title

Update README.md

Add a description

Add your description here...

Markdown is supported | Paste, drop, or click to add files

Remember, contributions to this repository should follow our GitHub Community Guidelines.

Reviewers: No reviews

Assignees: No one—assign yourself

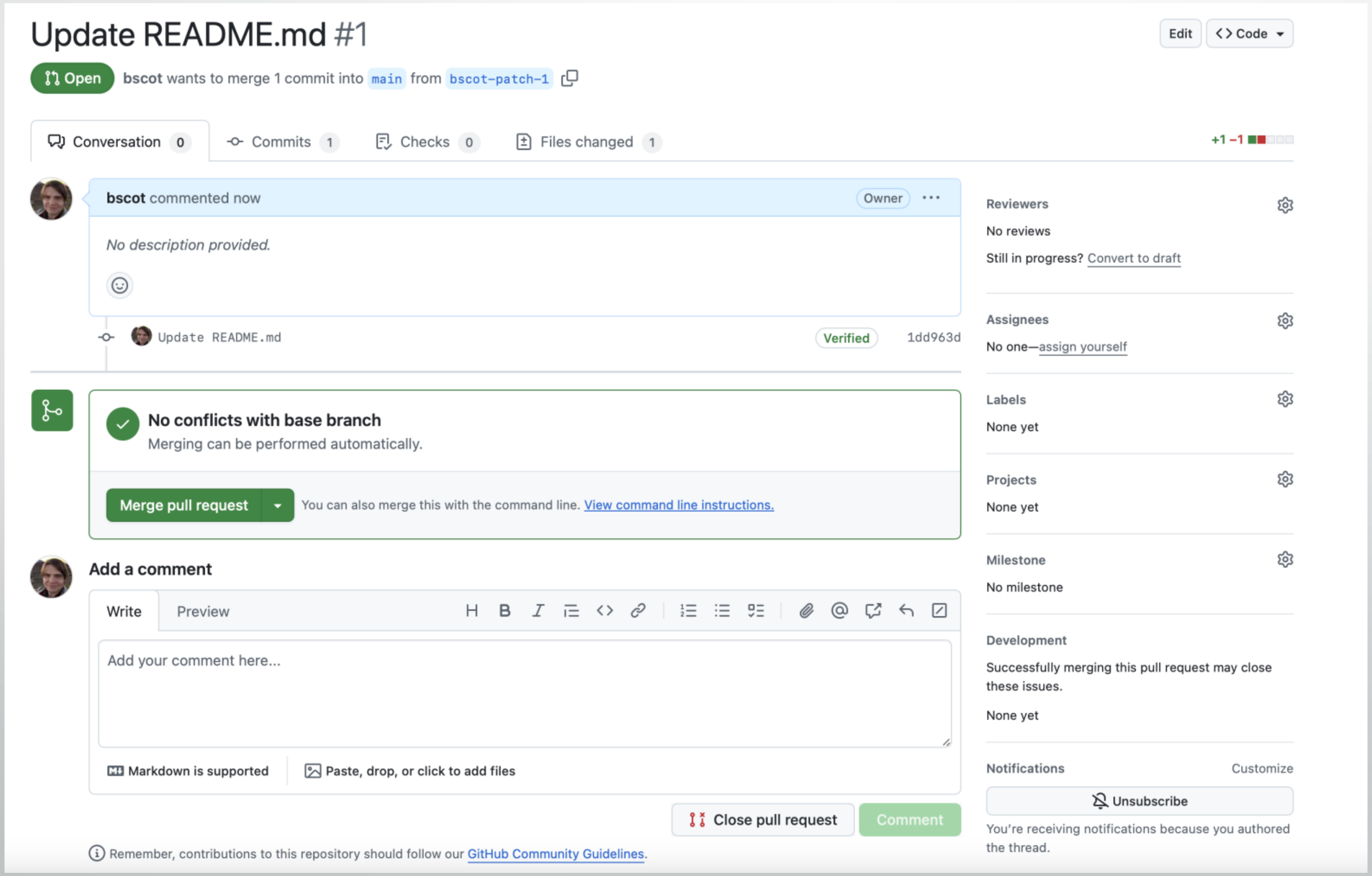
Labels: None yet

Projects: None yet

Milestone: No milestone

Helpful resources: GitHub Community Guidelines

Approving the Pull Request And Merging



Update README.md #1

bscot wants to merge 1 commit into main from bscot-patch-1

bscot commented now

No description provided.

Update README.md Verified 1dd963d

No conflicts with base branch

Merging can be performed automatically.

You can also merge this with the command line. View command line instructions.

Reviewers: No reviews. Still in progress? Convert to draft

Assignees: No one—assign yourself

Labels: None yet

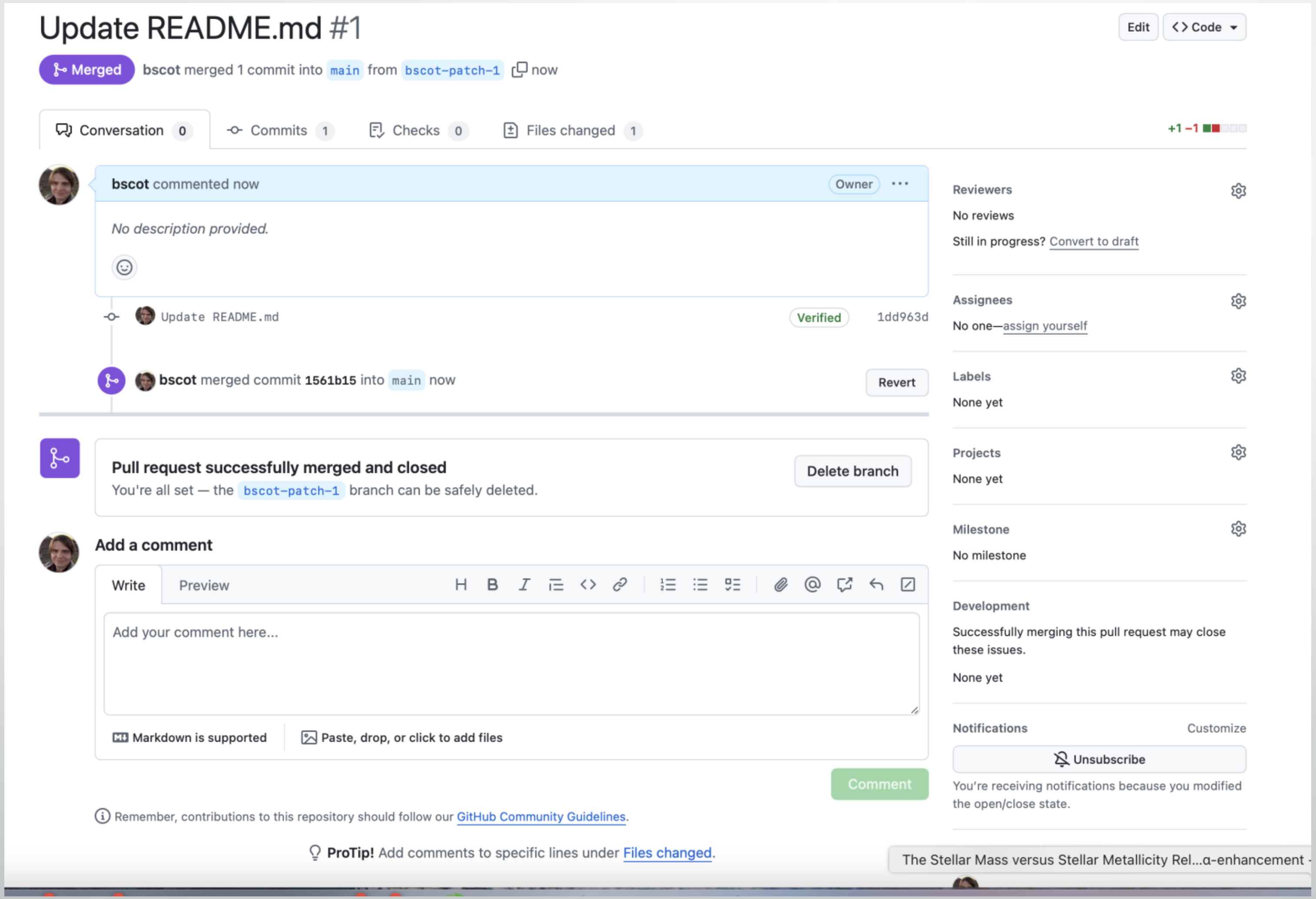
Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. None yet

You're receiving notifications because you authored the thread.

Remember, contributions to this repository should follow our GitHub Community Guidelines.



Update README.md #1

Edit | Code

Merged bscot merged 1 commit into `main` from `bscot-patch-1` now

Conversation 0 | Commits 1 | Checks 0 | Files changed 1 | +1 −1

bscot commented now — Owner

No description provided.

Update README.md — Verified — 1dd963d

bscot merged commit 1561b15 into `main` now — Revert

Pull request successfully merged and closed

You're all set — the `bscot-patch-1` branch can be safely deleted. — Delete branch

Add a comment

Write | Preview

Add your comment here...

Markdown is supported | Paste, drop, or click to add files

Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

ProTip! Add comments to specific lines under Files changed.

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Notifications — Customize
Unsubscribe
You're receiving notifications because you modified the open/close state.

The Stellar Mass versus Stellar Metallicity Rel...α-enhancement

Git "muscle memory'

```
git add [filename] # adds a file so that git tracks it
```

```
git status # shows what files have been added & modified + more
```

```
git commit –m [brief description] # commits the file to your repo
```

```
git push origin main # sends your repo to the remote repository
```

Git "muscle memory'

```
git add [filename] # adds a file so that git tracks it
```

```
git status # shows what files have been added & modified + more
```

```
git commit –m [brief description] # commits the file to your repo
```

```
git push origin main # sends your repo to the remote repository
```

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Summary: Git and Version Control

Version control gives you a 'global undo' button to revert changes in your code. It also helps make your code more open and reproducible, which makes your (and everyone else's) science better!

We've walked through cloning and forking the DSFP Session 23 repository, as well as how to push and pull to/from your fork/branch of it. For each session, we'll ask you to repeat this.

Summary: Git and Version Control

Version control gives you a 'global undo' button to revert changes in your code. It also helps make your code more open and reproducible, which makes your (and everyone else's) science better!

We've walked through cloning and forking the DSFP Session 23 repository, as well as how to push and pull to/from your fork/branch of it. For each session, we'll ask you to repeat this.