NC STATE UNIVERSITY

Markdown and Git

Justin Post

Course Plan

- R Projects
- Markdown basics
- Git & Github
- Creating a blog & website with R and github
- Automating R Markdown
- Writing a Book with bookdown
- Creating interactive apps with R shiny

RStudio Project

- Often have many files associated with an analysis
- With multiple analyses things get cluttered

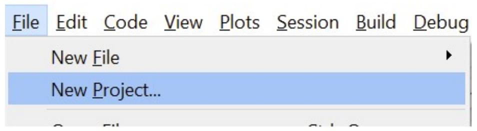
RStudio Project

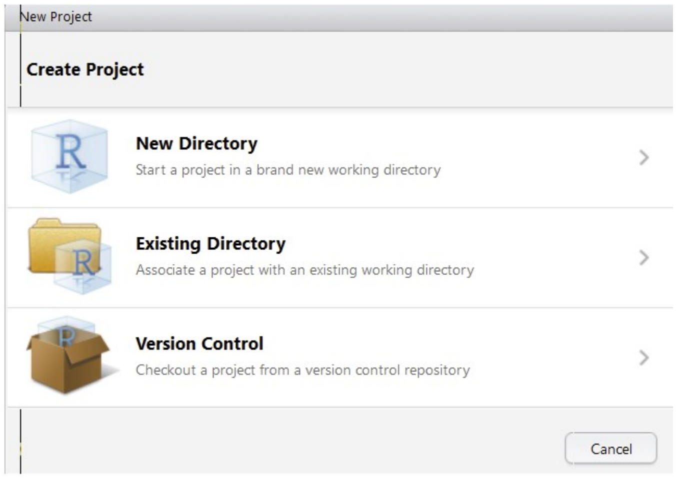
- Often have many files associated with an analysis
- With multiple analyses things get cluttered
- Want to associate different
  - environments
  - histories
  - working directories
  - source documents

  with each analysis
- Can use “Project” feature in R Studio

RStudio - Project

- Easy to create! Use an existing folder or create one:




New Project

Create Project

New Directory
Start a project in a brand new working directory

Existing Directory
Associate a project with an existing working directory

Version Control
Checkout a project from a version control repository

Cancel

- Place all files for that analysis in that directory

RStudio - Project

Create two new projects (with new empty folders):

- One called ‘github_website’
- One called ‘automation_of_markdown’

(We’ll also create one later from git.)

- On creation select the `renv` option

RStudio - Project

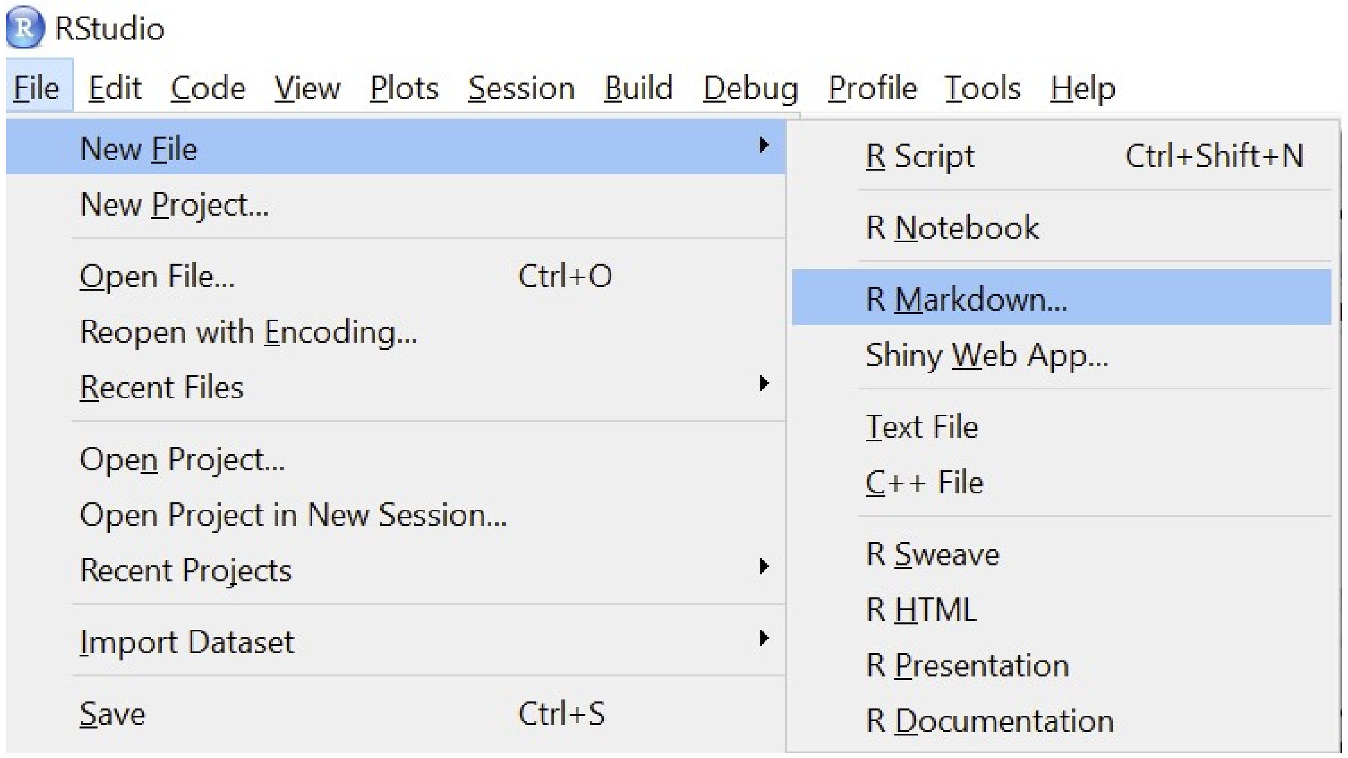
- Let's look at project options via `tools --> Project options`
- Switch between projects with the upper right menu
- Modify and save the project. Note the differing behavior of your R sessoin depending on your project options
- Open the .Rproj file in notepad

Markdown Basics

- Vital to make your work reproducible!
- Traditional documentation through comments (`#` in R) in script
- May have heard of JUPYTER notebooks
- R Markdown - built in notebook for R studio (a program that weaves word processing and code)

Creating an R Markdown Document

- Work in your 'github_website' project
- Create a new markdown doc via menus and let's explore it!



Markdown Verbage

- May have heard of HTML (HyperText Mark-up Language)
    - Write plain text that the browser interprets and renders

Markdown Verbage

- May have heard of HTML (HyperText Mark-up Language)
  - Write plain text that the browser interprets and renders
- Markdown is a specific markup language
  - Easier syntax
  - Not as powerful
- Any plain text file can be used (.Rmd extension associates it with R Studio)

R Markdown

R Markdown file contains three important types of content:

1. (Optional) YAML header surrounded by –––s
2. Chunks of R code
3. Text mixed with simple text formatting instructions

R Markdown - YAML Header

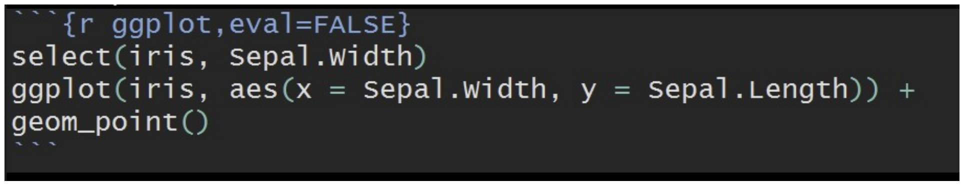
- Define settings for document

```
---
title: "Untitled"
author: "First Last"
date: "xxxx"
output: html_document
---
```

- CTRL/CMD + Shift + k **knits** (creates the output document) via this info
- Can also knit via the little arrow to knit to a different format
- Can knit via the `rmarkdown::render()` function

R Markdown - Code Chunks

- Below YAML header: ‘r chunk’



~~~
```{r ggplot,eval=FALSE}
select(iris, Sepal.Width)
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
geom_point()
```
~~~

- Start code chunk by typing ```{r} out or with CTRL/CMD + Alt/Option + I
- Code will be executed when document is created
- Can specify options on individual code chunks

R Markdown - Syntax

- Below code chunk is plain text with markdown sytnax

```
## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax
for authoring HTML, PDF, and MS Word documents. For more details on
using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that
includes both content as well as the output of any embedded R code
chunks within the document.
```

- When file created, “##” becomes a header, “<…>” a link, and `**Knit**` bold font

R Markdown - Syntax

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Markdown

Syntax is really easy to learn via the Cheat sheet

Quick look at the following:

- Markdown syntax
- Code chunks and their options
- Changing type of output

R Markdown syntax

- `# Header 1` becomes a large font header
- `## Header 2` becomes a slightly smaller font header
- Goes to 6 headers
  - Use of headers can automatically create a Table of Contents!
- `**bold**` **and** `__bold__`
- `` `code` `` becomes `code`

R Markdown syntax

- Can do lists: be sure to end each line with two spaces!
    - Indent sub lists four spaces

```
* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2
```

- unordered list
- item 2
    - sub-item 1
    - sub-item 2

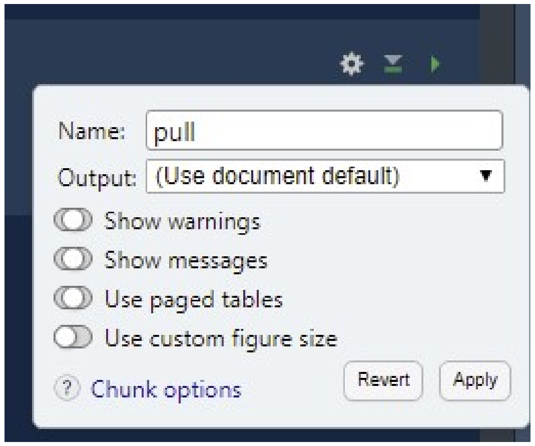
1. ordered list
2. item 2
    - sub-item 1
    - sub-item 2

Code chunks and their options

- Any R code can go into a chunk
- Usually a ‘set-up’ chunk at the top to read in packages, set global chunk options, etc.
- Chunks evaluate sequentially (can use output from prior chunk) that is saved as an R object - let’s check that!

Code chunks and their options

- Many options depending on chunk purpose!
- Can hide/show code with `echo = FALSE/TRUE`
- Can choose if code is evaluated with `eval = TRUE/FALSE`
- `message = TRUE/FALSE` and `warning = TRUE/FALSE` can turn on/off displaying messages/warnings



Name: pull

Output: (Use document default)

Show warnings

Show messages

Use paged tables

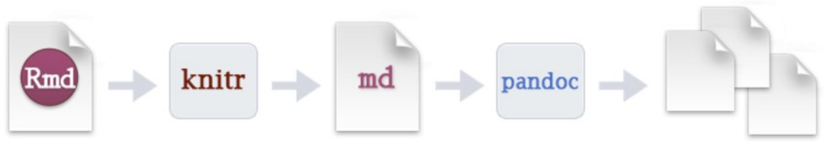
Use custom figure size

Documenting with Markdown

- R Markdown = Digital “Notebook”: Program that weaves word processing and code.
- Designed to be used in three ways (R for Data Science)
  - Communicating to decision makers (focus on conclusions not code)
  - Collaborating with other data scientists (including future you!)
  - As environment to do data science (can evaluate and edit/reevaluate code chunks and document what you did and what you were thinking)

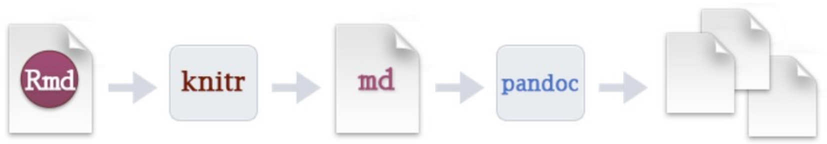
Changing type of output

R Markdown really flexible!



Changing type of output

R Markdown really flexible!



- Change output type in the YAML header and use CTRL/CMD + Shift + k
- knit via the menu
- Use code explicity: `rmarkdown::render("file.Rmd", output_format = "html_document")`
- Outputting to PDF requires a version of Tex. If you don't have one, install the `tinytex` package and run `tinytex::install_tinytex()`
- Check that you can output to HTML, PDF, and Word using one of the above methods

Other Options on Outputs

Check out the R Markdown definitive guide for cool options for each type of output!

- Try to implement code folding, tabsets, and a TOC in an HTML output doc. Can you do it via `render`?
- Try to implement the kable method of printing a data frame and a TOC in a PDF. Can you do it via `render`?

Output to multiple formats

Multiple outputs in one call:

- `rmarkdown::render("yourfile.Rmd", output_format = c("html_document", "pdf_document", "word_document")`
- You can also change the name of the output files via the `output_file` argument
- Can you get it to output three files called ‘my.html’, ‘your.pdf’, ‘their.docx’ with one function call?
- Let’s try it! (Note: You can’t specify options when rendering to multiple formats)

R Markdown

- Great for document your code/thoughts and for sharing your analyses easily!
- Next, we'll learn about git/github and how RStudio can work with them
- A similar Markdown language is used to render documents on github so we've already learned enough to create some basic webpages too :)

What are git and github?

Ideally we want to document our process, easily collaborate, and widely share our work

- To make our workflow for a project reproducible, ideally we would save different versions of our analysis, write-up, etc. along the way
- git is a version control software that easily allows multiple users to work on the **same** project. It simply tracks the changes that we `commit` to the files.
- No more finaldoc.pdf, finalfinaldoc.pdf, finaldoc08_11_21.pdf, …
- github is a hosting service that allows us to do Git-based projects on the internet and share them widely!

Tracking: Basic idea

- Create a repository on github.com (files saved remotely)
- When you (or collaborator) want to work on it, you `clone` the repo locally (or `pull` the files down to update it)
- You work, save things, etc.
- Your work was good! Nice. Now you want to `add` your modified/new files to the repo so other can use them.
- You then stage your `commit` (i.e. prepare everything to send back to the remote repo)
- You `push` your local committed changes up the the repo on github
- Everyone has access to it now!

Collaboration: Basic idea

Everyone can work on the same branch and update as needed (sometimes there will be merge conflicts, covered shortly)

- People often `fork` the repo or create their own `branch` to work on, rather than modify the main repository
- Perhaps the main repo is being used for something and you don't want to `push` up your changes until they are tested or the new 'feature' is debugged
- You work on your branch using the idea from the previous slide
- Once happy with everything, you do a `merge` request to combine your modified repo with the main repo branch

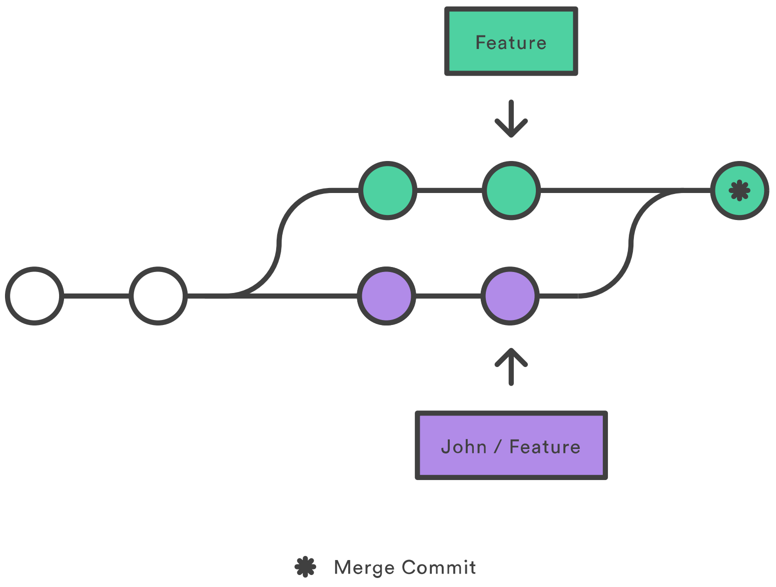
Visual

(from https://git.logikum.hu)

Each circle represents a 'commit' to that repository/branch (all version of files at each commit are kept!)

Merging vs. rebasing onto a remote branch

Merging



Rebasing

Wow - How do we get started??

Let’s look at an example repo and the commits done/how it is tracked!

- Start with basic workflow with yourself by creating a blog on github!

Blog Instructions

- Sign into github.com and then go to this repo
- Click `fork` in the top right corner. This will give you a copy of this repo under your account! Under settings, rename the repo to yourname.github.io
- Visit https://yourgithubusername.github.io to see the default blog page (i.e. you already have a blog :)

Blog Instructions

- Let’s make some changes! Click on the `_posts` folder. Edit the file you see there by clicking on the name. You can then click on the pencil in the right hand header for the file display.
    - Change the file name to today’s date.
    - Update the title in the YAML header.
    - Use the markdown syntax we learned to write a short paragraph or two (no R code chunks though!)
    - Write a commit message under ‘Commit changes’ and click the green button at the bottom
- It may take 2 minutes but visit https://yourgithubusername.github.io again to see the changes.

Cloning the repo

- To follow our previous idea (and to start doing this with R), we really don't want to use the web interface
- We can `clone` the repo (i.e. download the entire repo locally)
- Repo main page has a green button. Click on that.
    - Can download a zip and unzip it to an appropriate folder
    - Can clone via the URL and terminal or RStudio
- Open RStudio, go to new project, from Version Control, choose Git, and paste in the repo link. Select a directory to save this in and hit Create project!
- Now have the files locally!

Update check

We need to make sure RStudio and github can communicate. Do the following:

- Go to the `Git` tab in your `Environment` area
- You should see some files there. These are the ones that have changed from the remote repo (the one on github)
- Here you can add files that you'd like to commit up to the remote repo
- Click on all of the boxes (equivalent to `git add -A`) and click the `Commit` button

Update check

- This brings up a window that allows you to compare changes. If you are happy you can put a commit message in the box in the top right and click the commit button (equivalent to `git commit -m "message"`)
- Hit close on the window (you should see no errors, just a message about the commit)
- Now click the push button in the top right (equivalent to `git push`)
- You should be prompted to log-in in some way. Do so!
- Go to your repo on gitub.com and see the changes!

Alternative to the Menu

When working by myself on a repo, I’m not worried about merge conflicts with other people’s changes. As such, my workflow is as follows:

- Open the appropriate project in RStudio
- Go to the Terminal (switched to Bash) and type `git pull` (or use the git tab)
- Work… at a good spot for saving, back to the terminal
- Type `git add -A` to add all files that have been modified
- Type `git commit -m "Message"` to stage a commit
- Type `git push` to push the local changes to the remote repo

Try Out the Workflow

- Go to your blog repo and make a change via the web interface
- Your local repo is no longer up to date! (Type `git status` in the terminal to check). You'll need to pull down the changes.
- Now, let's update from RStudio!
- Update the about page of your blog by editing 'about.md'. Make changes like before and commit!
- Push up the changes. In 2-3 minutes you should see the About page updates :)

Creating a Repo

Let’s create a repo with our already created ‘github_website’ project

- Go to github.com and create a repo with that name (Use the + in the top right corner - don’t initiate a git ignore file)
- Open the github_website project we worked on earlier
- Go to Tools –> Project Options –> Git (restart R as requested)
- Now in the terminal do the following:
    - `git remote add origin [paste the clone link here]` (initiate the tracking of the project on github)
    - `git pull origin main` (download all remote files)
    - `git push -u origin main` (track changes on this machine)
- Add, commit, and push your files up!

Creating a website

Very easy! Two steps:

- Enable github pages in your repo settings (on github.com)
- Create a README.Rmd file. Change the output type to `github_document`
- Write whatever R code you want, add, commit, and push up your changes
- Now you'll have a site like https://jbpost2.github.io/github_website/

Automating R Markdown

Let's take a break from git!

- Other markdown functionality: Using `parameters`
- Can be added to the YAML header and can be used to automate reports!
- Suppose we are dealing with a football box score data set

```
NFLData <- read_csv("https://www4.stat.ncsu.edu/~online/datasets/scoresFull.csv")
NFLData
```

```
## # A tibble: 3,471 x 82
##   week  date  day   season awayTeam     AQ1   AQ2   AQ3   AQ4   AOT  AOT2 AFinal
##   <chr> <chr> <chr>  <dbl> <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 1     5-Sep Thu     2002 San Franc~     3     0     7     6    -1    -1     16
## 2 1     8-Sep Sun     2002 Minnesota~     3    17     0     3    -1    -1     23
## 3 1     8-Sep Sun     2002 New Orlea~     6     7     7     0     6    -1     26
## 4 1     8-Sep Sun     2002 New York ~     0    17     3    11     6    -1     37
## 5 1     8-Sep Sun     2002 Arizona C~    10     3     3     7    -1    -1     23
## # ... with 3,466 more rows, and 70 more variables: homeTeam <chr>, HQ1 <dbl>,
## #   HQ2 <dbl>, HQ3 <dbl>, HQ4 <dbl>, HOT <dbl>, HOT2 <dbl>, HFinal <dbl>,
## #   stadium <chr>, startTime <time>, toss <chr>, roof <chr>, surface <chr>,
## #   duration <dbl>, attendance <chr>, weather <chr>, vegasLine <chr>, OU <chr>,
## #   AfirstDowns <dbl>, AnetPassYds <dbl>, AtotalYds <dbl>, Aturnovers <dbl>,
```

Markdown Parameters

Parameters can be added to the YAML header

```
title: "NFL Reports"
author: "Justin Post"
output: html_document
params:
      team: "Pittsburgh Steelers"
```

- Can ‘Knit with parameters’
- In .Rmd, access via `params$team`
- Example: Let’s open up the NFL.Rmd document

Parameters

May want to create a similar document/output for all 32 teams

- Utilize code method for knitting the document

```
rmarkdown::render("NFL.Rmd", output_file = "Cleveland Browns.html",
                                    params = list(team = "Cleveland Browns"))
```

- Would create same document using the Cleveland Browns data

Parameters

Plan:

- Create data frame that has
    - file names to output to
    - list with each team name for using in `render()`

For one team the row would be (last column's value is really a list with one value in it)

```
##                  output_file                team
## 1 Pittsburgh Steelers.html Pittsburgh Steelers
```

Parameters

```r
#get unique teams
teamIDs <- unique(NFLData$awayTeam)
#create filenames
output_file <- paste0(teamIDs, ".html")
#create a list for each team with just the team name parameter
params = lapply(teamIDs, FUN = function(x){list(team = x)})

#put into a data frame
reports <- tibble(output_file, params)
```

Parameters

```
reports
```

```
## # A tibble: 32 x 2
##   output_file                params
##   <chr>                      <list>
## 1 San Francisco 49ers.html <named list [1]>
## 2 Minnesota Vikings.html   <named list [1]>
## 3 New Orleans Saints.html  <named list [1]>
## 4 New York Jets.html       <named list [1]>
## 5 Arizona Cardinals.html   <named list [1]>
## # ... with 27 more rows
```

Parameters

Now knit using `apply()` or via `purrr::pwalk()`

```
library(rmarkdown)
#need to use x[[1]] to get at elements since tibble doesn't simplify
apply(reports, MARGIN = 1,
            FUN = function(x){
                render(input = "files/NFL.Rmd", output_file = x[[1]], params = x[[2]])
                })

#or with pwalk (args are .l, .f, and ...)
#.l is a list of lists, .f is function, formula, or vector
pwalk(reports, render, input = "files/NFL.Rmd")
```

Parameters

This can be done with multiple parameters.

- Nice way to automate creation
- Could put into a nice pipeline
    - Create file to update NFL data each week (scrape new data and add to .csv file)
    - Create .Rmd file that you want for each team
    - Create file to submit creation of documents with params
    - Put all into one file (say with `source`)
    - Have that file automatically run!

Try it yourself!

Try to do a similar process where you use the `iris` data frame and create three separate analysis based off of the `Species` column.

- Create a parameter called `species`
- Run the following code in your markdown doc:

```
myIris <- filter(iris, params$species)
summary(myIris)
plot(myIris)
```

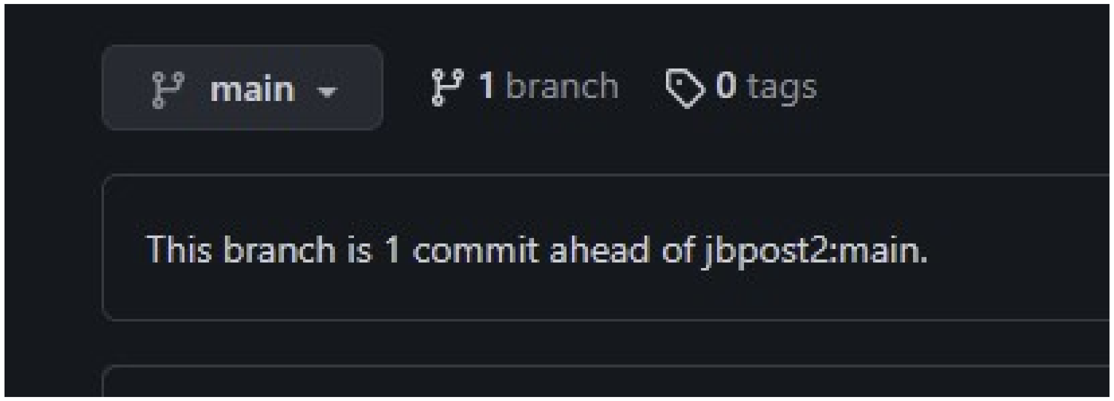
- Use `apply()` or `pwalk()` to generate the three reports.

Bookdown

- You can easily write a book using RStudio too!
- Can be done via File –> New Directory –> Book Project Using bookdown
- Instead we'll clone this repo and then create an R project from a git repository
- Let's see how to build the book!
- Chapters are just .Rmd files (named appropriately `01-intro.Rmd`) that start with a `#`

Merging Branches

- Seems like a good idea to talk about merging branches!
- If you do a commit on your branch, you may notice something like this

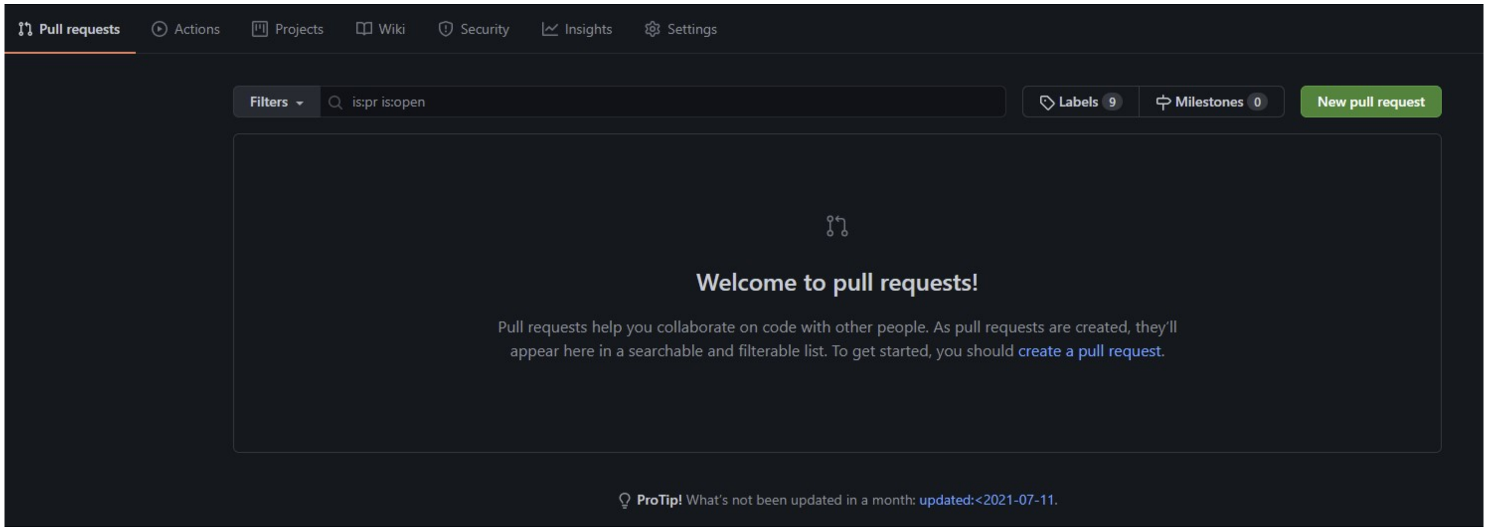


This branch is 1 commit ahead of jbpost2:main.

Merging Branches

Suppose you like your commit and you think I will too!

- You can do a `pull` request



Welcome to pull requests!

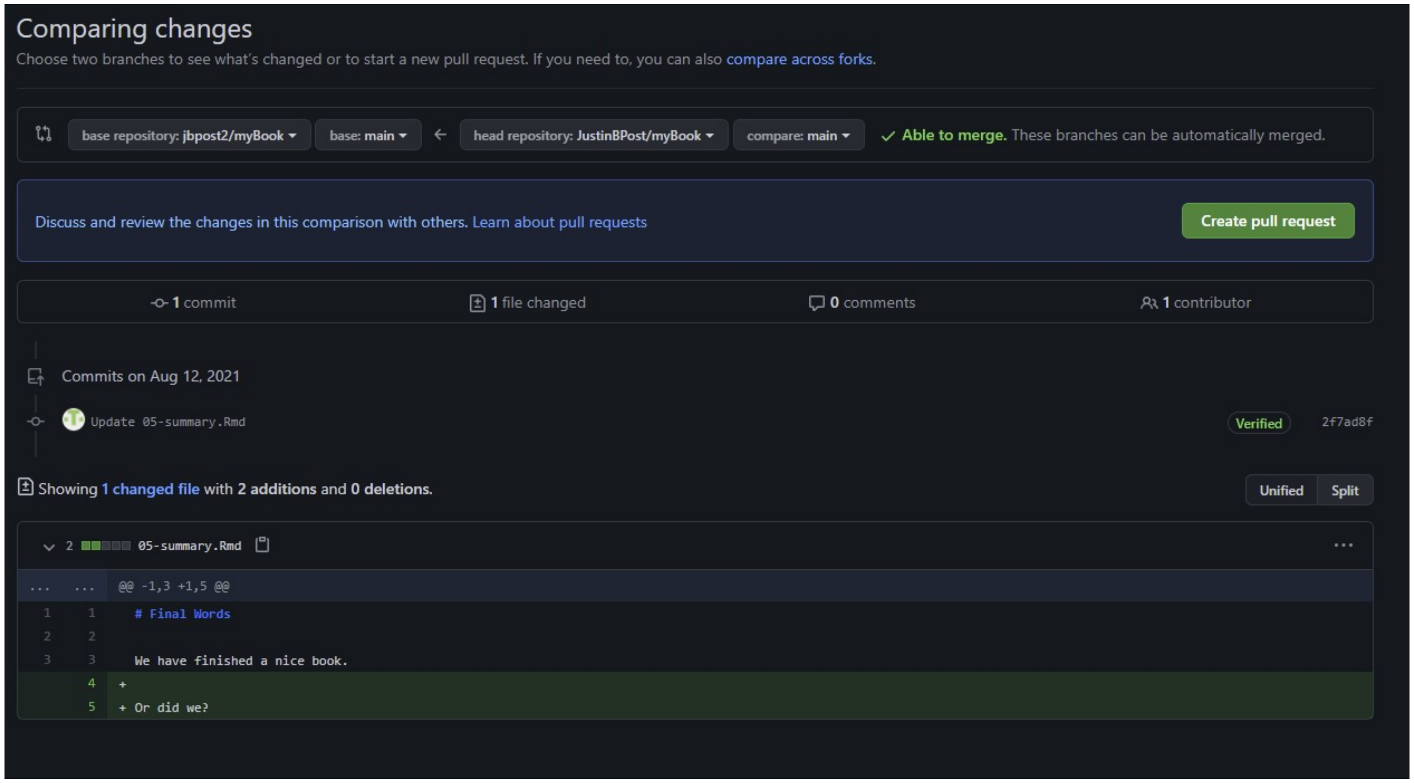
Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should create a pull request.

ProTip! What's not been updated in a month: updated:<2021-07-11.

Merging Branches

Suppose you like your commit and you think I will too!

- You can do a `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.

base repository: jbpost2/myBook ▾ | base: main ▾ | ← | head repository: JustinBPost/myBook ▾ | compare: main ▾ | ✓ **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 | **0** comments | **1** contributor

Commits on Aug 12, 2021

`Update 05-summary.Rmd` Verified 2f7ad8f

Showing 1 changed file with **2 additions** and **0 deletions**.

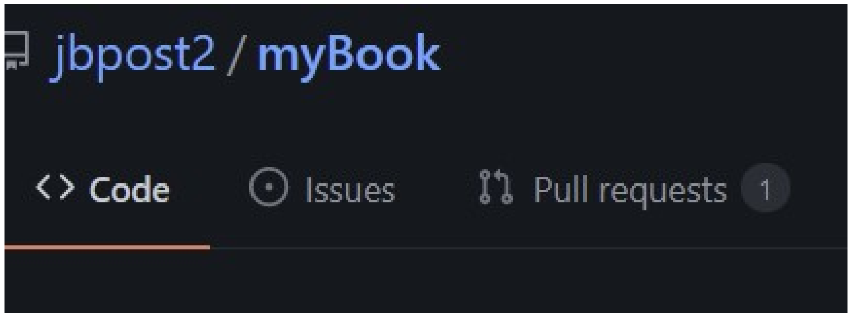
```
2 05-summary.Rmd
@@ -1,3 +1,5 @@
 # Final Words

 We have finished a nice book.
+
+ Or did we?
```

Merging Branches

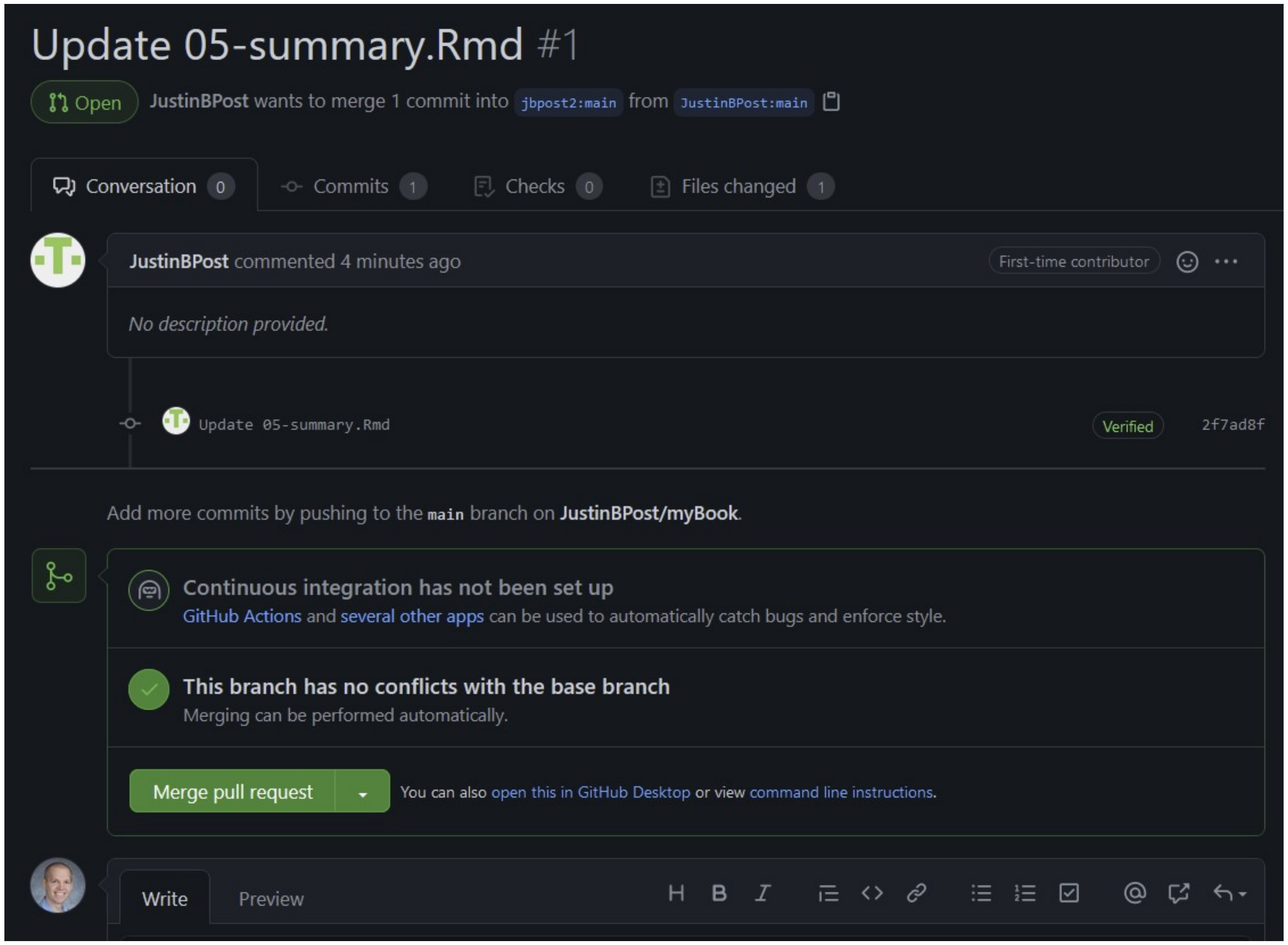
If you are lucky, there won't be any merge conflicts.

- Allows the owner of the original repo to accept the pull request without needing to modify things
- The owner will get a notification that a pull request has been made



Merging Branches

Owner can then investigate the request and choose whether or not to accept it or they can ask for more details


Update 05-summary.Rmd #1

Open JustinBPost wants to merge 1 commit into `jbpost2:main` from `JustinBPost:main`

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

JustinBPost commented 4 minutes ago — First-time contributor

No description provided.

`Update 05-summary.Rmd` — Verified — `2f7ad8f`

Add more commits by pushing to the **main** branch on **JustinBPost/myBook**.

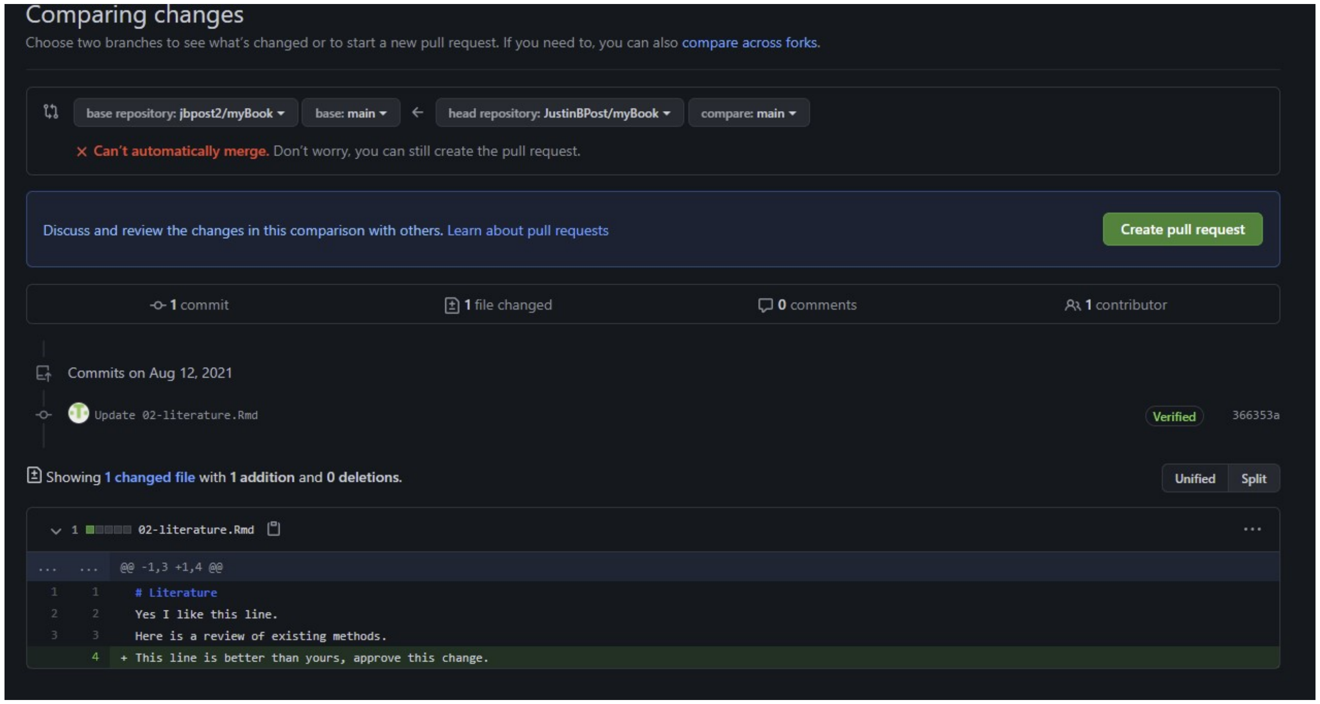
Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Dealing with conflicts

- Sometimes changes requested conflict with changes already made



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.

base repository: jbpost2/myBook · base: main ← head repository: JustinBPost/myBook · compare: main

× Can't automatically merge. Don't worry, you can still create the pull request.

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

1 commit · 1 file changed · 0 comments · 1 contributor

Commits on Aug 12, 2021

Update 02-literature.Rmd — Verified — 366353a

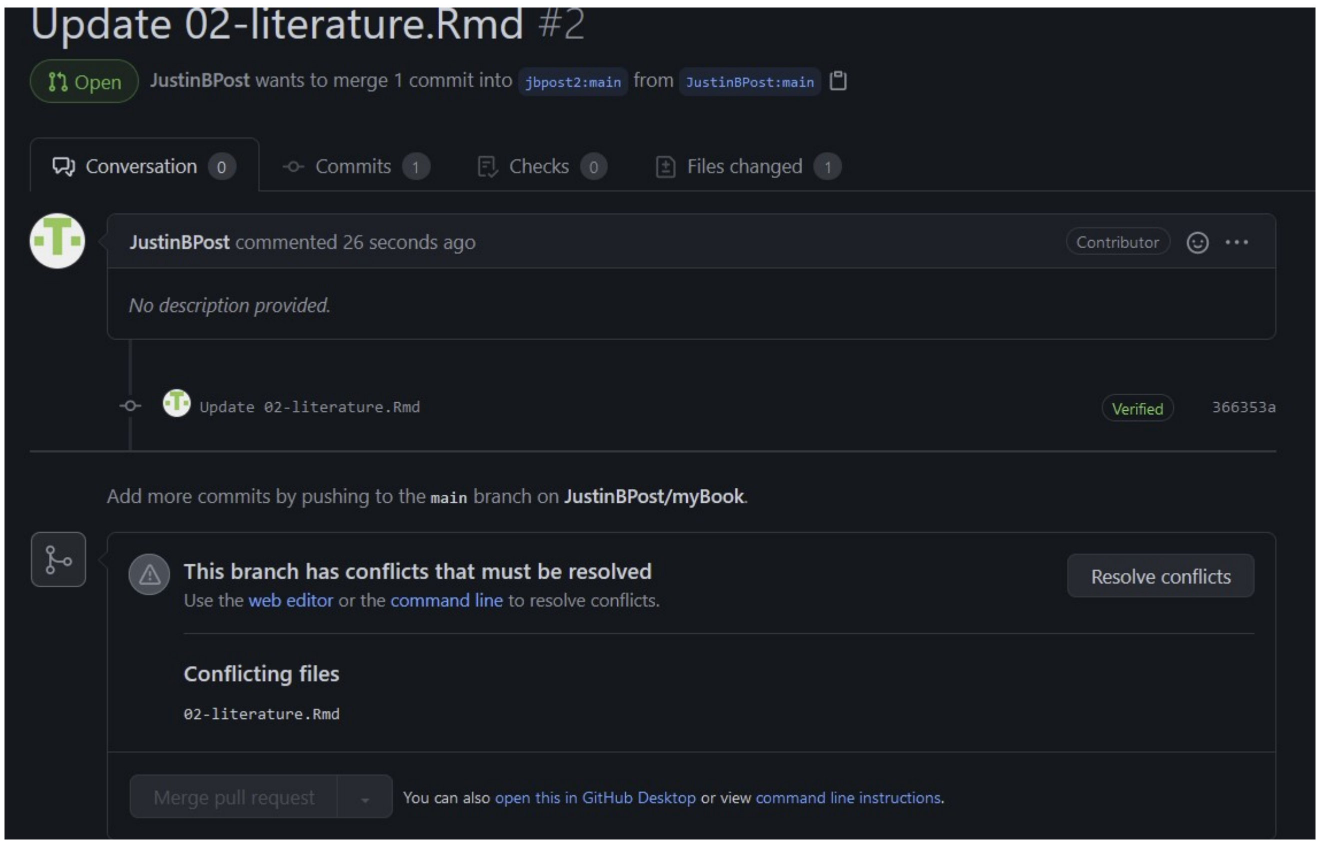
Showing 1 changed file with 1 addition and 0 deletions.

1 02-literature.Rmd

```
@@ -1,3 +1,4 @@
  # Literature
  Yes I like this line.
  Here is a review of existing methods.
+ This line is better than yours, approve this change.
```

Dealing with conflicts

Owner sees a notification about conflicts that must be resolved



Update 02-literature.Rmd #2

Open JustinBPost wants to merge 1 commit into jbpost2:main from JustinBPost:main

Conversation 0 Commits 1 Checks 0 Files changed 1

JustinBPost commented 26 seconds ago Contributor

No description provided.

Update 02-literature.Rmd Verified 366353a

Add more commits by pushing to the main branch on JustinBPost/myBook.

This branch has conflicts that must be resolved
Use the web editor or the command line to resolve conflicts.

Resolve conflicts

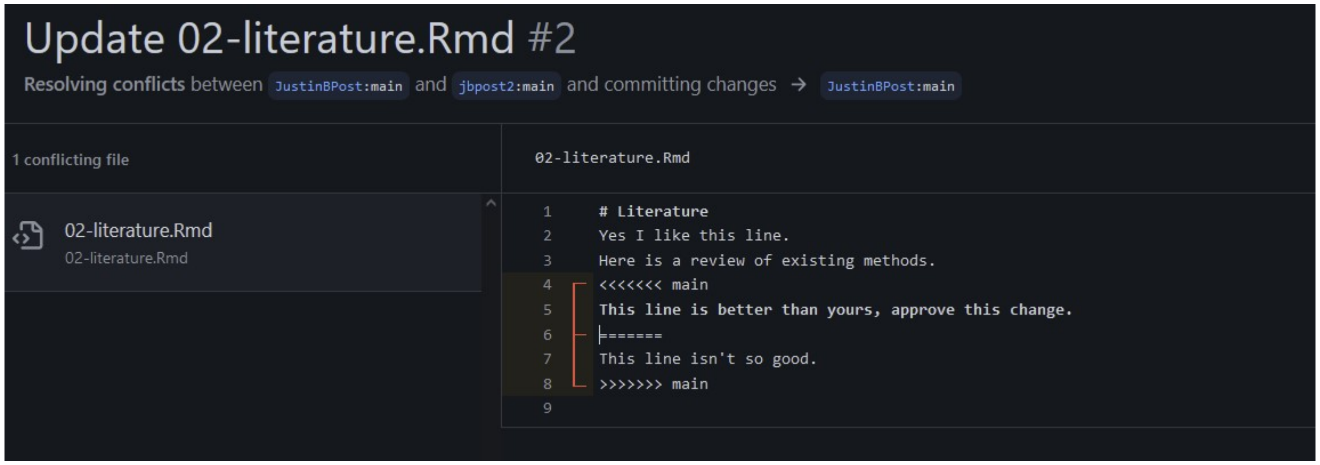
Conflicting files
02-literature.Rmd

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Dealing with conflicts

They can view the issues and pick which to include or to include both with a modification

<<<<<<< is a conflict marker


Update 02-literature.Rmd #2

Resolving conflicts between JustinBPost:main and jbpost2:main and committing changes → JustinBPost:main

1 conflicting file

02-literature.Rmd

```
# Literature
Yes I like this line.
Here is a review of existing methods.
<<<<<<< main
This line is better than yours, approve this change.
=======
This line isn't so good.
>>>>>>> main
```

- Figure out what to do and delete the <<< === >>> lines

Course Plan

- R Projects
- Markdown basics
- Git & Github
- Creating a blog & website with R and github
- Automating R Markdown
- Writing a Book with bookdown
- Creating interactive apps with R shiny