Git Guide

Xingyu Wang

Goal No.1

Create a new RStudio Project. In this folder, create a simple qmd file called example.qmd that can be knitted into a HTML file. Show the result of the knitted file.

- Step 1: Create a New RProject: In RStudio: File → New Project → New directory → Fill in a new project name → Select location via Browser → Create Project
- Step 2: Create the example.qmd and show the result of the knitted file: File → New file → Quarto Document → Fill in the title and check HTML option → create → Remove example material → Save as "example"→ Render to show the result

Goal No.2

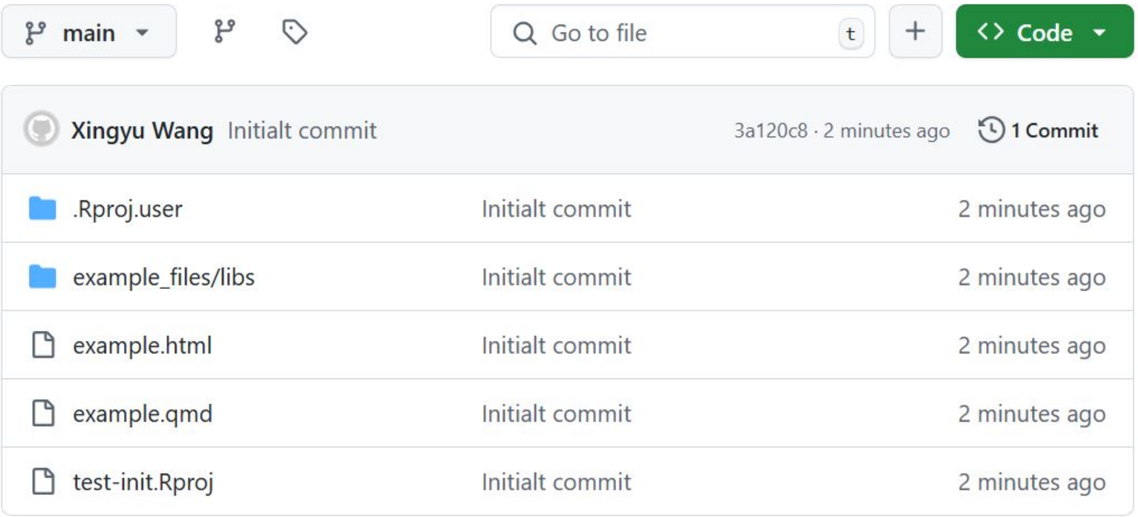
From the command line interface, initialize this folder as a git repository and push it to the GitHub classroom repository.

- Step 1: Create a new repo on GitHub. Please ensure the new repo is empty with no content(Do not select "Add README" or "Add license").
- Step 2: Navigate back to RStudio's terminal pane and run the following code.

```
git init
   # This will initializes a .git repository configuration directory.
git add .
git commit -m "Initialt commit"
git branch -M main
   # Change the current branch name from Master to main.
git remote add origin https://github.com/Your_user_name/Your_repo_name
   #Modify the content according to your repo.
git push -u origin main
```

Note: You can also find the above code on the created repo's homepage.

- Step 3: Refresh the GitHub page to view the synchronized files.(see Figure 1)



Xingyu Wang Initialt commit 3a120c8 · 2 minutes ago 1 Commit

.Rproj.user	Initialt commit	2 minutes ago
example_files/libs	Initialt commit	2 minutes ago
example.html	Initialt commit	2 minutes ago
example.qmd	Initialt commit	2 minutes ago
test-init.Rproj	Initialt commit	2 minutes ago

Figure 1: result of initialization

Goal No.3

Create a new branch called testbranch. Modify the file example.qmd and add the changes to both the local and remote repositories.

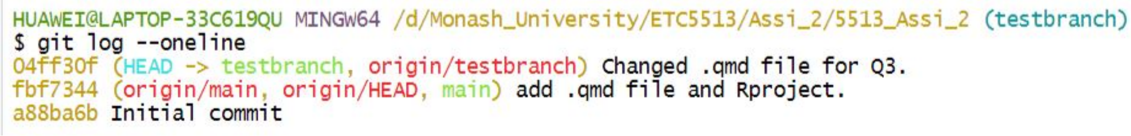
- Step 1: Create testbranch and make some changes in example.qmd and push to git repo

```
git branch testbranch
git checkout testbranch
```

- Step 2: Edit example.qmd: Add "Change from Q3" in the first line.

```
git add example.qmd
git commit -m "Changed .qmd file for Q3."
git push origin testbranch
```

(See the result in Figure 2)



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (testbranch)
$ git log --oneline
04ff30f (HEAD -> testbranch, origin/testbranch) Changed .qmd file for Q3.
fbf7344 (origin/main, origin/HEAD, main) add .qmd file and Rproject.
a88ba6b Initial commit
```

Figure 2: Result of editing example.qmd on testbranch.

Goal No.4

Create a folder called data, and add the data from Assignment 1 to that folder. Amend the previous commit to include the data folder. Push this amended commit to the remote.

- Step 1: Create data folder and move Assignment 1 dataset into it

```
mkdir data
```

- Step 2: Copy Assignment 1 dataset into data folder.

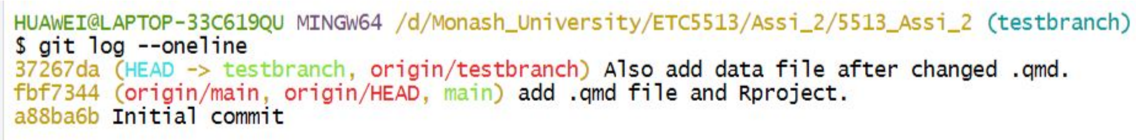
```
git add data
git commit --amend -m "Also add data file after changed .qmd."
```

Note: Using "-m" option after "–amend" avoids entering the text editor for commit messages.

- Step 3: Push the commit to the remote

```
git push -f origin testbranch
```

Note: Since commit history will be changed, –force is needed. Only I'm editing this branch, so it's safe - but could cause conflicts if others built on old commits.



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (testbranch)
$ git log --oneline
37267da (HEAD -> testbranch, origin/testbranch) Also add data file after changed .qmd.
fbf7344 (origin/main, origin/HEAD, main) add .qmd file and Rproject.
a88ba6b Initial commit
```

Figure 3: Result of amend commit.

Git log shows taht commit was modified (from Figure 2 to Figure 3).

Goal No.5

Switch back to the main branch and modify example.qmd in a different way such that it will conflict with testbranch. Commit and push these changes.

```
# switch back to main.
git checkout main

# Edit the first line of example.qmd file.
git add example.qmd
git commit -m "Changed .qmd from main."
git push origin main
```

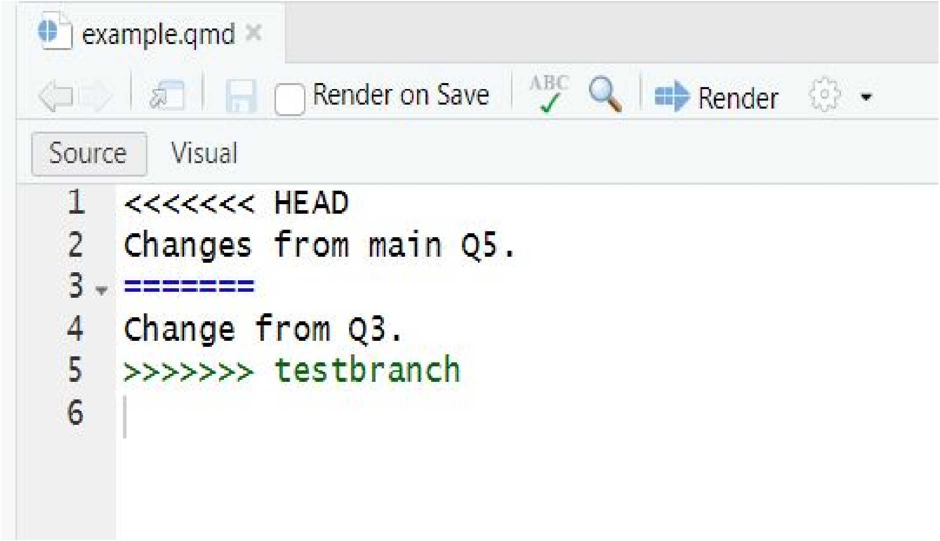
Goal No.6

Merge the changes in testbranch onto main. Show the conflict and fixing the merge conflict. Push the changes to GitHub, showing the status and changes in GitHub.

- Step 1: Merge the changes.

```
git merge testbranch
```

- Step 2: Merge conflict detected(See Figure 4).

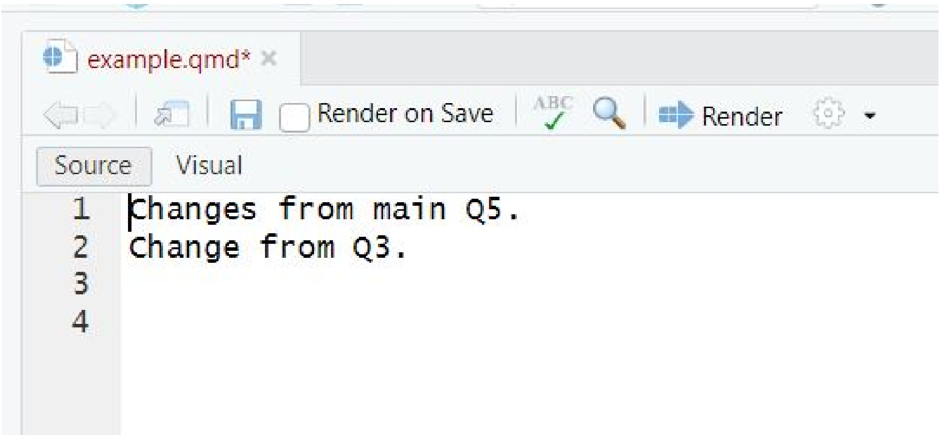


```
<<<<<<< HEAD
Changes from main Q5.
=======
Change from Q3.
>>>>>>> testbranch

```

Figure 4: Result of conlict.

- Step 3: Conflict resolution: preserve both modifications and delete the conflict indicators(See Figure 5).



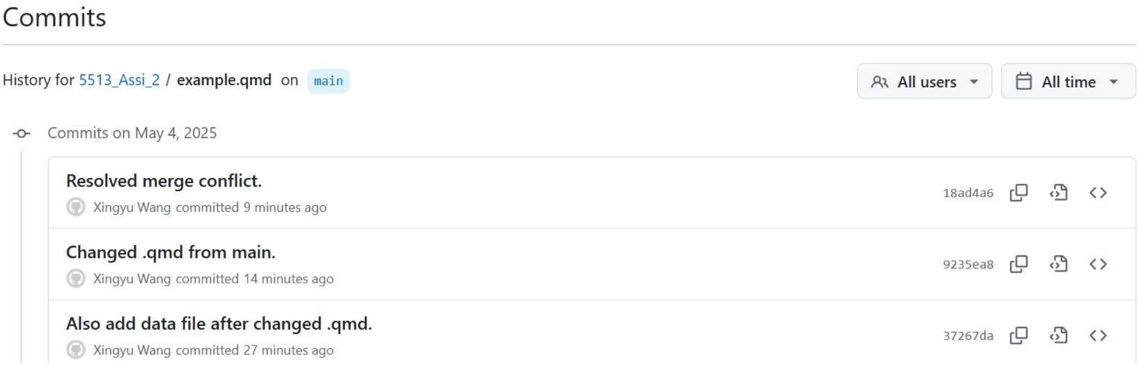
```
Changes from main Q5.
Change from Q3.


```

Figure 5: Result of conflict resolved.

```
git add example.qmd
git commit -m "Resolved merge conflict."
git push origin main
```

See the status Figure 6 and changes Figure 7 in GitHub



Commits

History for 5513_Assi_2 / example.qmd on main

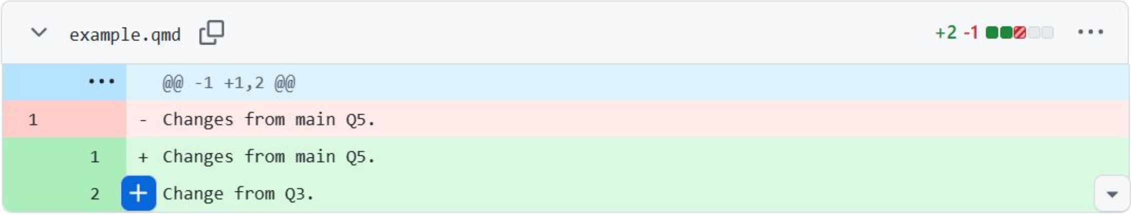
Commits on May 4, 2025

Resolved merge conflict.
Xingyu Wang committed 9 minutes ago
18ad4a6

Changed .qmd from main.
Xingyu Wang committed 14 minutes ago
9235ea8

Also add data file after changed .qmd.
Xingyu Wang committed 27 minutes ago
37267da

Figure 6: Status in GitHub



```
@@ -1 +1,2 @@
1       - Changes from main Q5.
    1   + Changes from main Q5.
    2   + Change from Q3.
```

Figure 7: Changes in GitHub

Goal No.7

Tag this commit v1.0 on main using an annotated tag.

```
git tag -a v1.0
```

Note: By default, git tag generates lightweight tags; the -a option creates annotated tags.

After running the code, Vim will open - enter a tag comment, for example: “Create tag v1.0”, then exit: Press Esc, type :wq.

Goal No.8

Delete branch testbranch locally and on the remote.

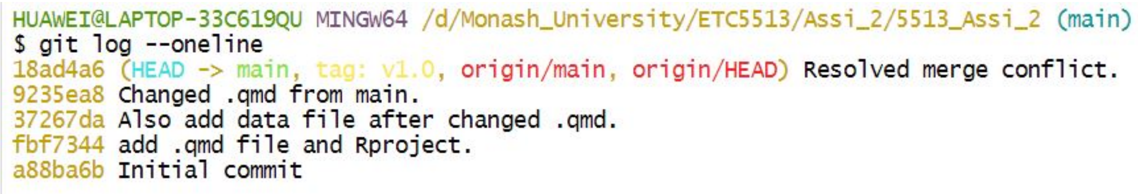
```
git branch -d testbranch
git push origin -d testbranch
```

Goal No.9

Show the commit log in condensed form in the terminal.

```
git log --oneline
```

Note: “–oneline” option shows each commit as one line - just hash and message. (See the result in Figure 8)



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git log --oneline
18ad4a6 (HEAD -> main, tag: v1.0, origin/main, origin/HEAD) Resolved merge conflict.
9235ea8 Changed .qmd from main.
37267da Also add data file after changed .qmd.
fbf7344 add .qmd file and Rproject.
a88ba6b Initial commit
```

Figure 8: Commit log in condensed form.

Goal No.10

On main, create a new section in example.qmd that includes an easy to make plot. Commit the changes, and demonstrate using the command line interface how to undo the commit without losing your local changes.

- Step 1: Add # and a space before section text to create a new section

```
# Easy Plot Example
```

- Step 2: Draw a simple plot. Load library.

```
library(ggplot2)
```

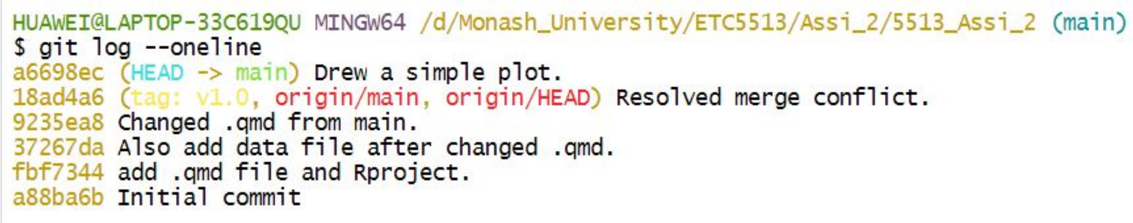
Draw a simple plot.

```
ggplot(iris, aes(Species)) +
  geom_bar()
```

- Step 3: Commit the changes

```
git add example.qmd
git commit -m "Drew a simple plot."
git log -oneline
```

(See the result in Figure 9)



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git log --oneline
a6698ec (HEAD -> main) Drew a simple plot.
18ad4a6 (tag: v1.0, origin/main, origin/HEAD) Resolved merge conflict.
9235ea8 Changed .qmd from main.
37267da Also add data file after changed .qmd.
fbf7344 add .qmd file and Rproject.
a88ba6b Initial commit
```

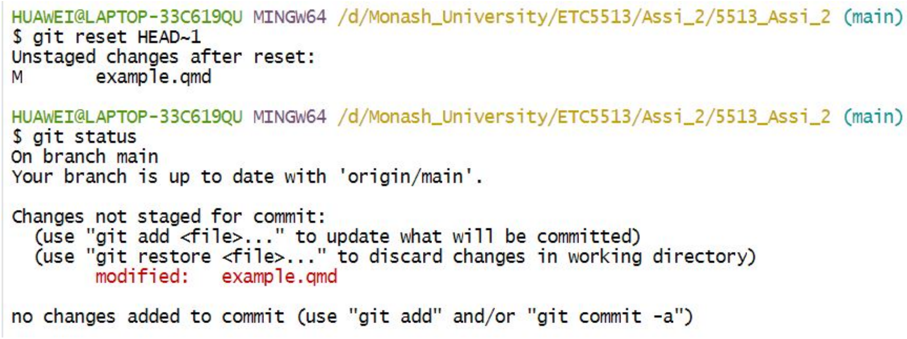
Figure 9: git log

- Step 4: 2 ways to undo the commit without losing local changes.

Approach 1: Use mixed reset

```
git reset HEAD~1
```

Running the code will undo the previous change to the working area, leaving it unstaged. (See Figure 10)



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git reset HEAD~1
Unstaged changes after reset:
M       example.qmd

HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

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

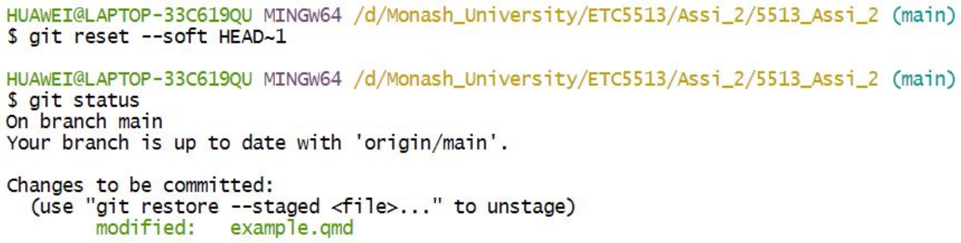
Figure 10: Result of using mixed reset.

Edit commit message(Test mixed reset) and recommit.

Approach 2: Use soft reset

```
git reset --soft HEAD~1
```

The code above will undo changes to the staging area.(See Figure 11)



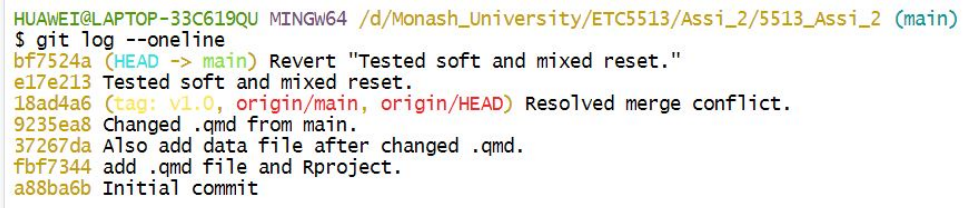
```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git reset --soft HEAD~1

HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (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:   example.qmd
```

Figure 11: Result of using soft reset.

Edit commit message(Tested soft and mixed reset.) and recommit. (See Figure 12)



```
HUAWEI@LAPTOP-33C619QU MINGW64 /d/Monash_University/ETC5513/Assi_2/5513_Assi_2 (main)
$ git log --oneline
bf7524a (HEAD -> main) Revert "Tested soft and mixed reset."
e17e213 Tested soft and mixed reset.
18ad4a6 (tag: v1.0, origin/main, origin/HEAD) Resolved merge conflict.
9235ea8 Changed .qmd from main.
37267da Also add data file after changed .qmd.
fbf7344 add .qmd file and Rproject.
a88ba6b Initial commit
```

Figure 12: Git log.