# GitHub

## Repositories – “Folders”

To demonstrate the concept of version control, you are going to create a new GitHub repository.

### Create a repository

1. Log in to GitHub (<https://github.com>). If you have not yet created a free account, and wish to follow along, do so.
2. Click the  button (upper left-hand area) to create a new repository.
3. On the **Create a new repository** page, name this something like “test-repo”.
   * It is a good habit to provide a repo *Description*, but this is not required.
   * You may have noticed the *Private* option. More on this later.
4. Check the box to “Initialize this repository with a README”.
5. Click on the green  button.

You have successfully made a GitHub repository! In this folder you have the README.md file that you told GitHub to create.

### Edit your markdown file

1. Click on the README.md filename on your repo main page.
2. Click on the Edit icon 
3. Add some content here. Some notes:
   * Text surrounded in single asterisks (e.g., \*example of text\*) will be made italic when you save your changes
   * Text surrounded in double asterisks (e.g., \*\*example of text\*\*) will be made bold when you save you changes.
   * A quick reference for markdown syntax can be found here: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

I’ve placed this document on the R: Drive

1. When you have added content to your hearts content (or until I’m ready to move on), scroll to the bottom of your README file and add a meaningful comment (where it currently says “Update README.md”).
   * A more detail comment is optional, but available.
2. Click on the button.

You have now edited a markdown document and begun to use a good workflow when working in Git (providing meaningful commit messages)!

### Add files to your repo

1. Double check that you are at your repo main page (not your README file).
2. On the R: Drive (STAT > Dykes > lets-git-together), I have put three documents (day1.html, day1.md, and day1.pdf). Drag-and-drop these files into your repo main page.
   * This might not work on your browser. There is a **Upload files** button near the upper/middle right-hand side.
3. Add a meaningful commit message and click on the green **Commit changes** button.
4. Explore each file by click on the file name from the main repo page. Discuss with your neighbor which files are easily viewable and which aren’t.

You’ve now done a couple of commits. Next we will compare commits with branching and merging. A good overview of branching and merging can be watched here: <https://www.youtube.com/watch?v=MJUJ4wbFm_A>

### Branching

1. On your repo main page, create a new branch:
   1. Find the “Branch: master” button in the upper/middle left-hand side and click on it.
   2. In the text box, call your new branch “test-branch”.
   3. Verify that you are in your new branch by view that same “Branch:…” button, but it should now say “Branch: test-branch”.
2. In your new branch, edit your README file with a relative link to the day1.md file by using the following syntax:

[a relative link](filename.md)

Comment and commit your changes.

1. Explore these two branches by switching between them. How do the repo structures differ?

### Merging

1. Merge your “test-branch” with your “master” branch by opening a **pull request**
   1. Click on the  button
   2. You should see that when you compare your master branch to your test-branch branch, you are **Able to merge**.
   3. Provide a meaningful comment (e.g., “merging test-branch to master”) and click on the  button.
   4. You shouldn’t see any issues here, so click on the  button and confirm.
   5. Delete your “test-branch”.
2. Make a new branch called “new-test-branch”.
3. Edit line 1 of the README files on **BOTH** branches to something different in each file. For example, in the master branch add “this is the master branch” and in the new-test-branch branch add “this is the new-test-branch”.
4. Try merging with a pull request. You should receive a *merge conflict* – resolve it (your choice), then merge.

Merge conflicts will more than likely come up as you collaborate in Git/GitHub. To resolve them, you (the repo owner) just needs to decide which is the correct version. To minimize merge conflicts, I encourage you to commit often!

# STA 418/518 demo

GitHub classroom + RStudio