Last episode,

We went over the basic principals of version control with Git and GitHub within GUI's such as RStudio and GitKraken. Today, we will review some of those prinicipals and practice them "out of the box" within the command line (terminal).

I swear, this will be fun. And even if you hate it now, you'll thank me later because there are command you should know and understand in case you get stuck. GUI's are great but sometimes they have bugs. Bugs are gross. We shold be able to rely on our understanding of basic git commands to keep calm, and carry on (pest control).

Let us review:

- **No github without git**
- Git = source control system
- Distributed = full history of all changes (all work done on local then can sync with remote)
  - Free and open source
- Pros: fast, scalable, can work disconnected (bc it’s distributed), powerful and easy, branching (new feature, new branch), pull requests (enables collaborating - intrinsically brings discussion and reviews thus improving quality of code)
- Cons: different (mind switch if you’re used to more traditional control system), learning curve, tools(most powerful on the command line, but gui can be useful), large binary files
- Github = hosting service for git repositories – global – runs on top of git but offers a lot of other features – free and paid options

Materials:

- Git - available on any OS
- An editor
  - RStudio
- GH account

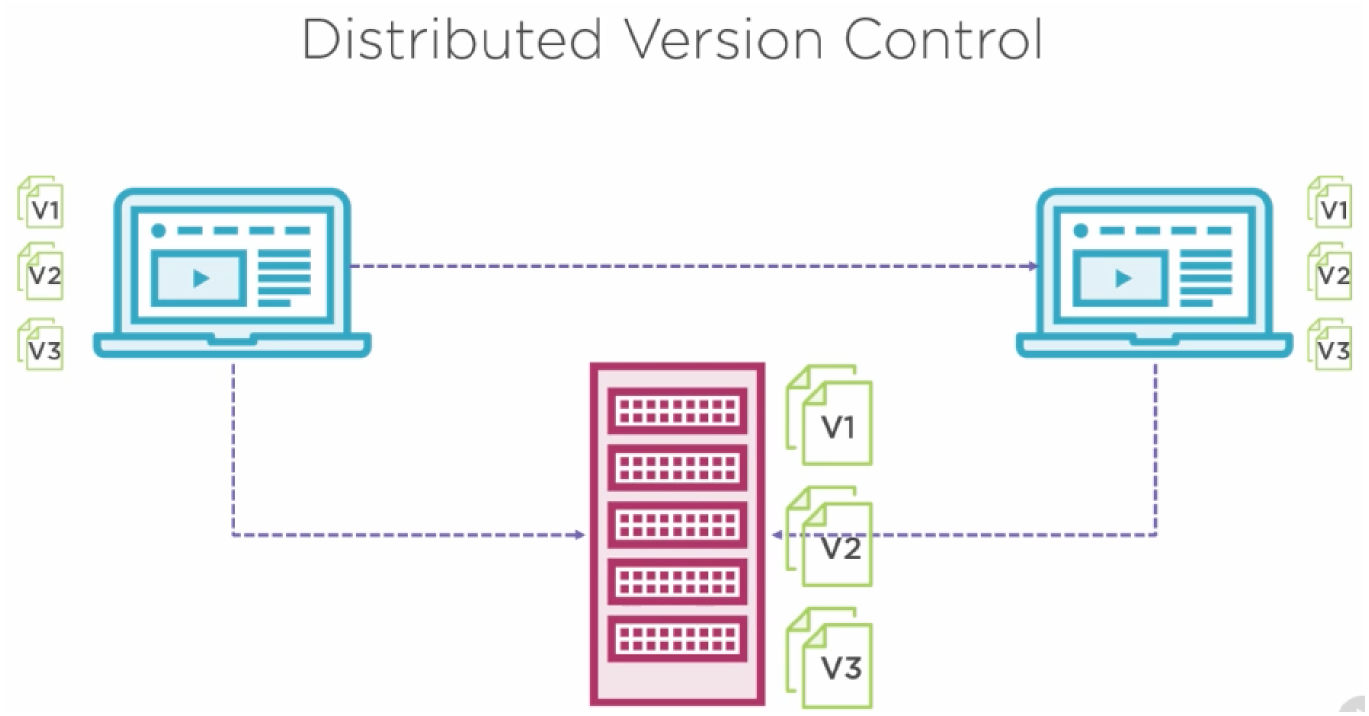
Demo: Setup Git

- Terminal/command line is a direct line for human to interact with computer, just like an application but better
- Let’s first make sure everything is set up
  - `git –version`
- Configure git locally – globally
  - Git bash

- Git config --global
  - User.name “lb”
  - User.email “email”
- Git config –edit –global
  - To see configurations
  - Will open on default editor

- Git config file in the local user directory

Foundations of Git:

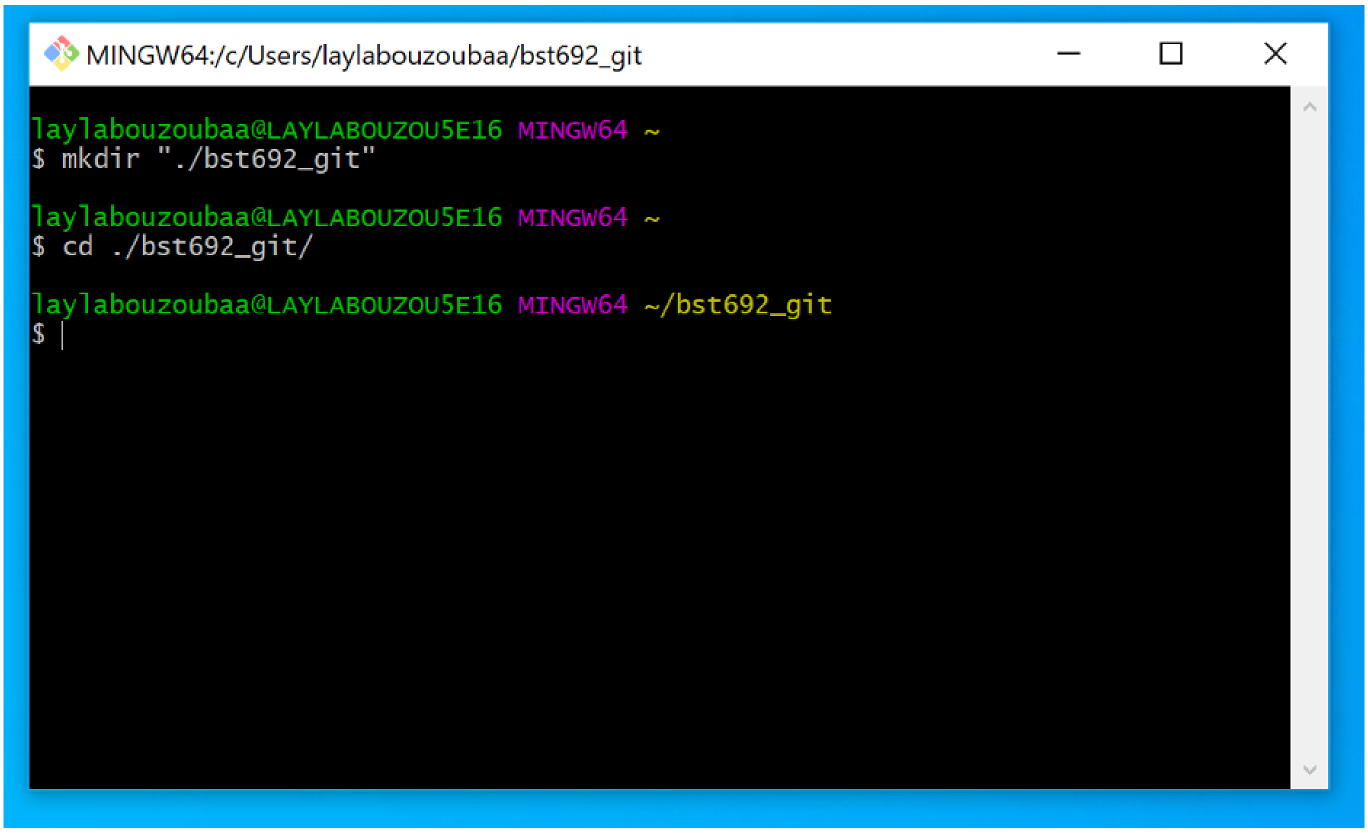


- Three states- promotion based system – ALL LOCAL
  - Committed
    - Data stored in local db
  - Modified
    - File has been changed but not committed
  - Staged
    - Modified file has been marked to be part of the next commit snapshot
- Three areas of git
  - Working directory
  - Staging area
    - Waiting to be committed
  - .git repo
    - Once a commit is made
    - When files are committed they are moved to this local directory
    - Entirely managed by git
    - Also created when we clone source from GitHub
  - 4th area – remote repo
    - Github
- Common commands:
  - All start with `git`

  - `git config`
  - `git init`
    - Create an empty repository
    - Initialize git
  - `git clone`
    - Download a project from a remote
  - `git add`
    - Prepare a file for staging
  - `git commit -m "yourcommitmessage"`
    - Commit changes to a local repository
    - Include a message

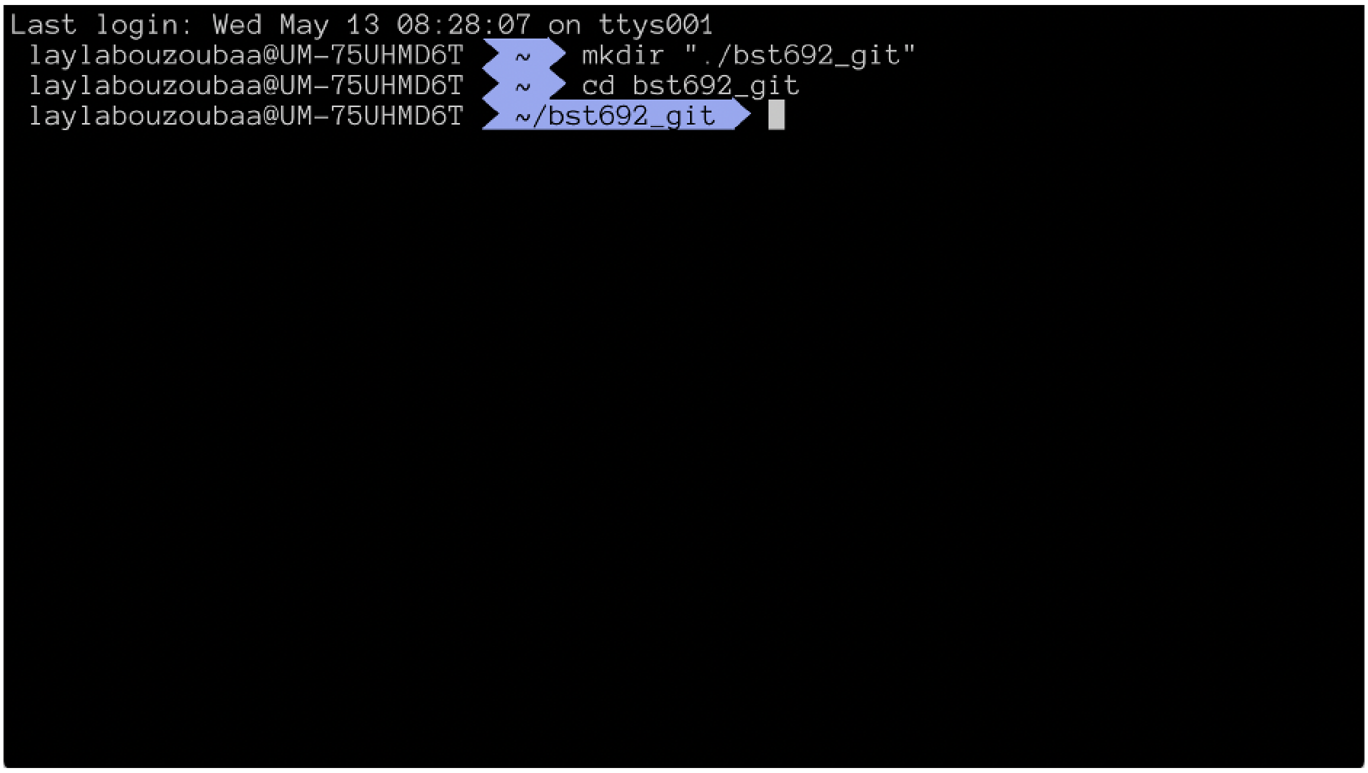
Demo: Create local repo

- Create a local repo
  - `mkdir "./bst692_git"`
  - Hint: use tab to autocomplete
  - Bash/terminal now knows you're in the directory
  - Note: not a git directory, yet

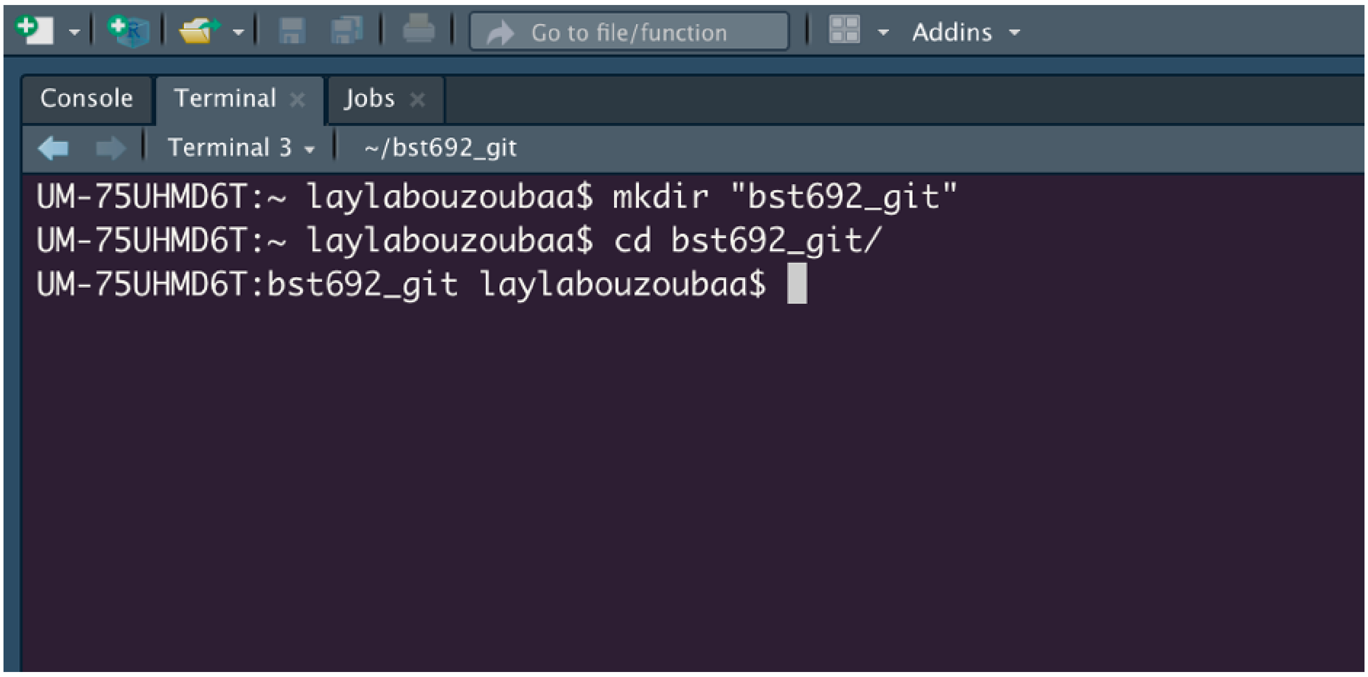


MINGW64:/c/Users/laylabouzoubaa/bst692_git

```
laylabouzoubaa@LAYLABOUZOU5E16 MINGW64 ~
$ mkdir "./bst692_git"

laylabouzoubaa@LAYLABOUZOU5E16 MINGW64 ~
$ cd ./bst692_git/

laylabouzoubaa@LAYLABOUZOU5E16 MINGW64 ~/bst692_git
$ |
```

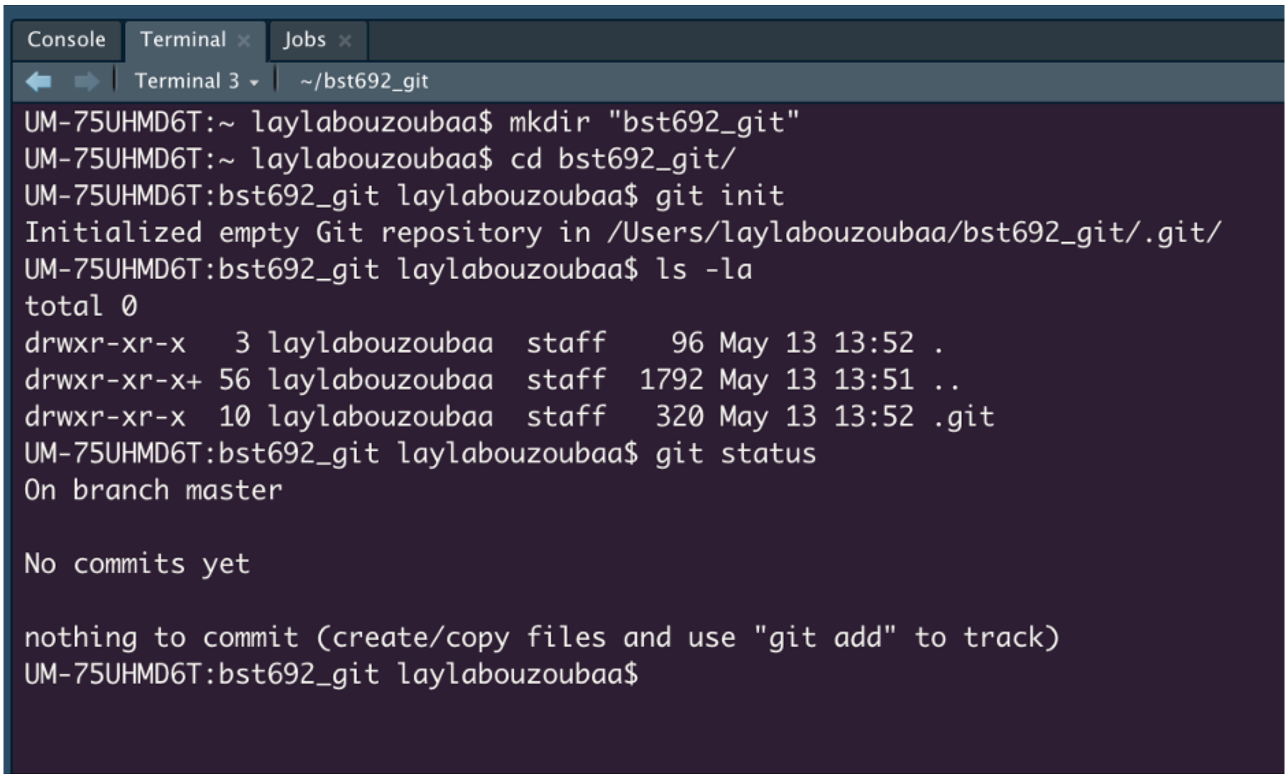


```
Last login: Wed May 13 08:28:07 on ttys001
 laylabouzoubaa@UM-75UHMD6T  ~  mkdir "./bst692_git"
 laylabouzoubaa@UM-75UHMD6T  ~  cd bst692_git
 laylabouzoubaa@UM-75UHMD6T  ~/bst692_git 
```



```
UM-75UHMD6T:~ laylabouzoubaa$ mkdir "bst692_git"
UM-75UHMD6T:~ laylabouzoubaa$ cd bst692_git/
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

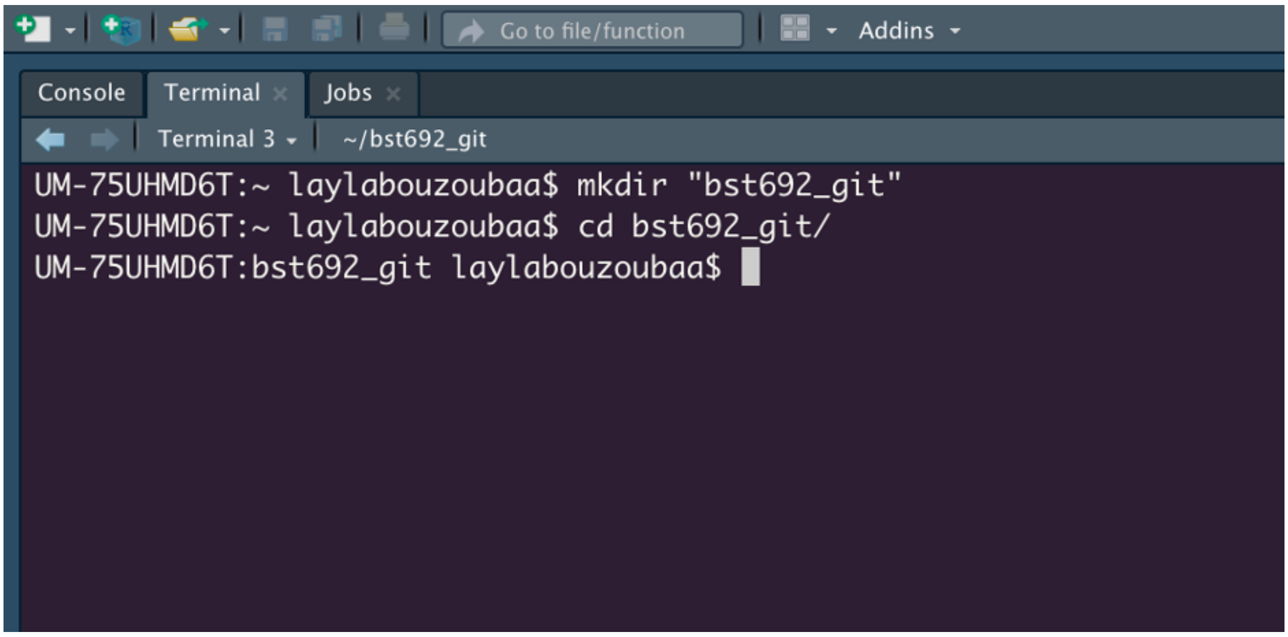
- Initiate git
  - `git init`
- See what is in the folder
  - `ls -la`
- Basic commands to work with git
  - `git status`
    - No commits yet obviously



```
UM-75UHMD6T:~ laylabouzoubaa$ mkdir "bst692_git"
UM-75UHMD6T:~ laylabouzoubaa$ cd bst692_git/
UM-75UHMD6T:bst692_git laylabouzoubaa$ git init
Initialized empty Git repository in /Users/laylabouzoubaa/bst692_git/.git/
UM-75UHMD6T:bst692_git laylabouzoubaa$ ls -la
total 0
drwxr-xr-x   3 laylabouzoubaa  staff    96 May 13 13:52 .
drwxr-xr-x+ 56 laylabouzoubaa  staff  1792 May 13 13:51 ..
drwxr-xr-x  10 laylabouzoubaa  staff   320 May 13 13:52 .git
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

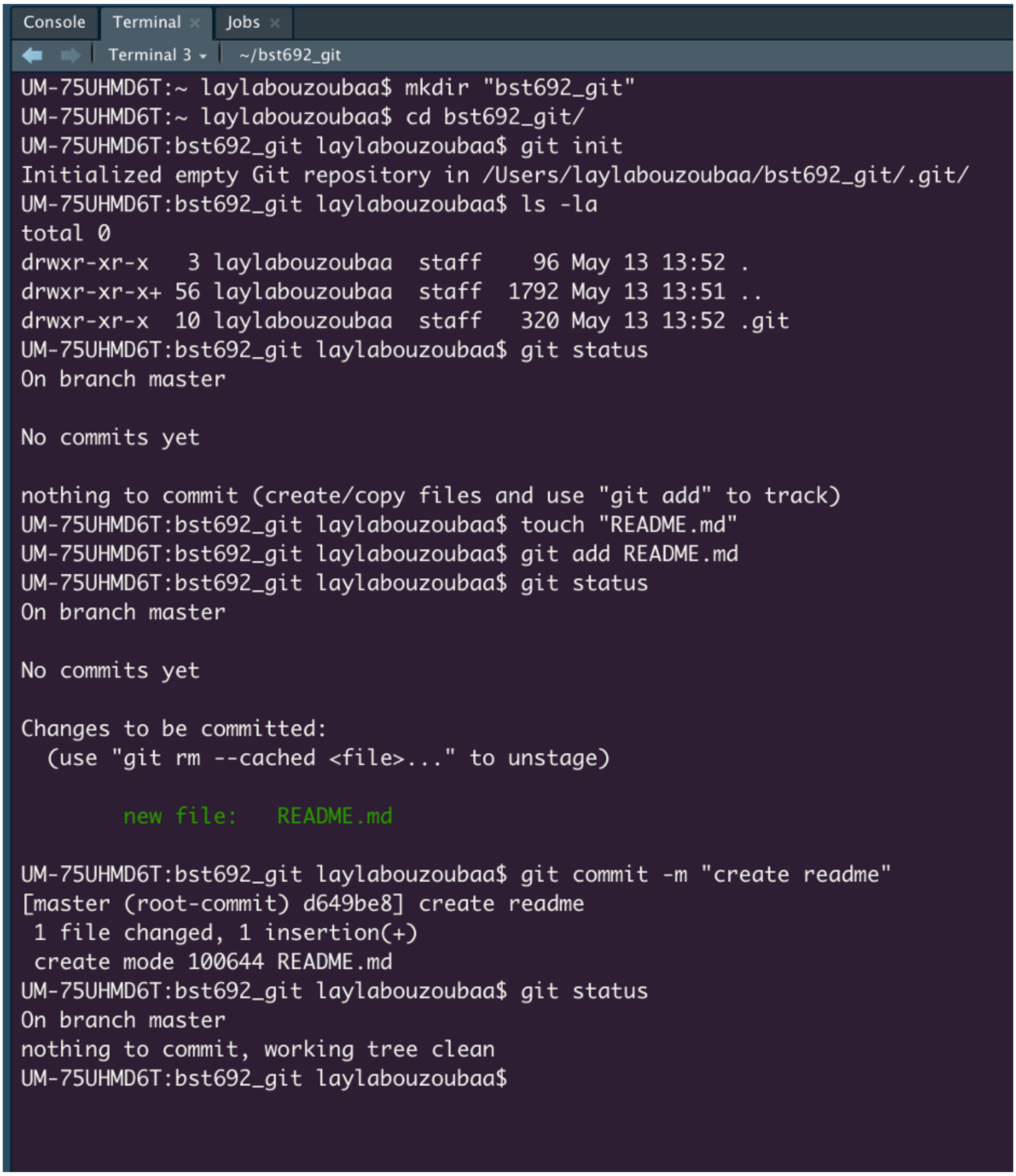
  - Create a new file

        - `touch README.md`



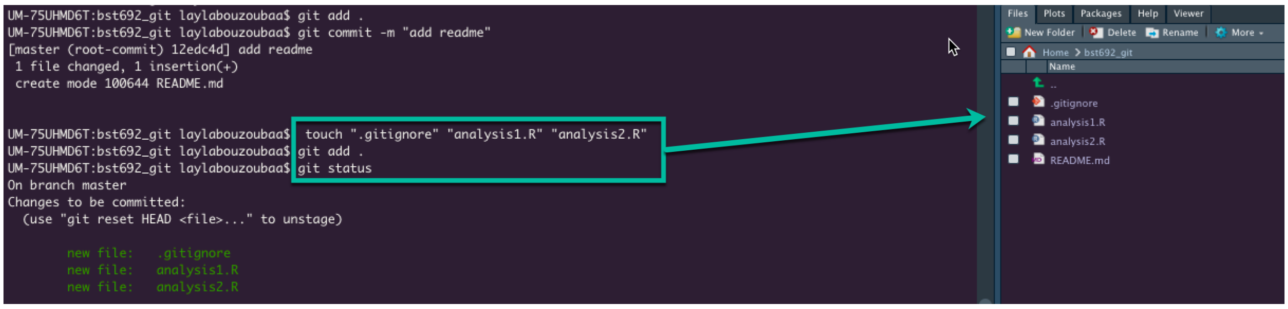
```
UM-75UHMD6T:~ laylabouzoubaa$ mkdir "bst692_git"
UM-75UHMD6T:~ laylabouzoubaa$ cd bst692_git/
UM-75UHMD6T:bst692_git laylabouzoubaa$ 
```

    - Stage it
        - `git add README.md`
    - `git status`
    - Commit
        - `git commit m "make readme"`
    - `git status`
        - No commits just yet but git is now tracking the readme file



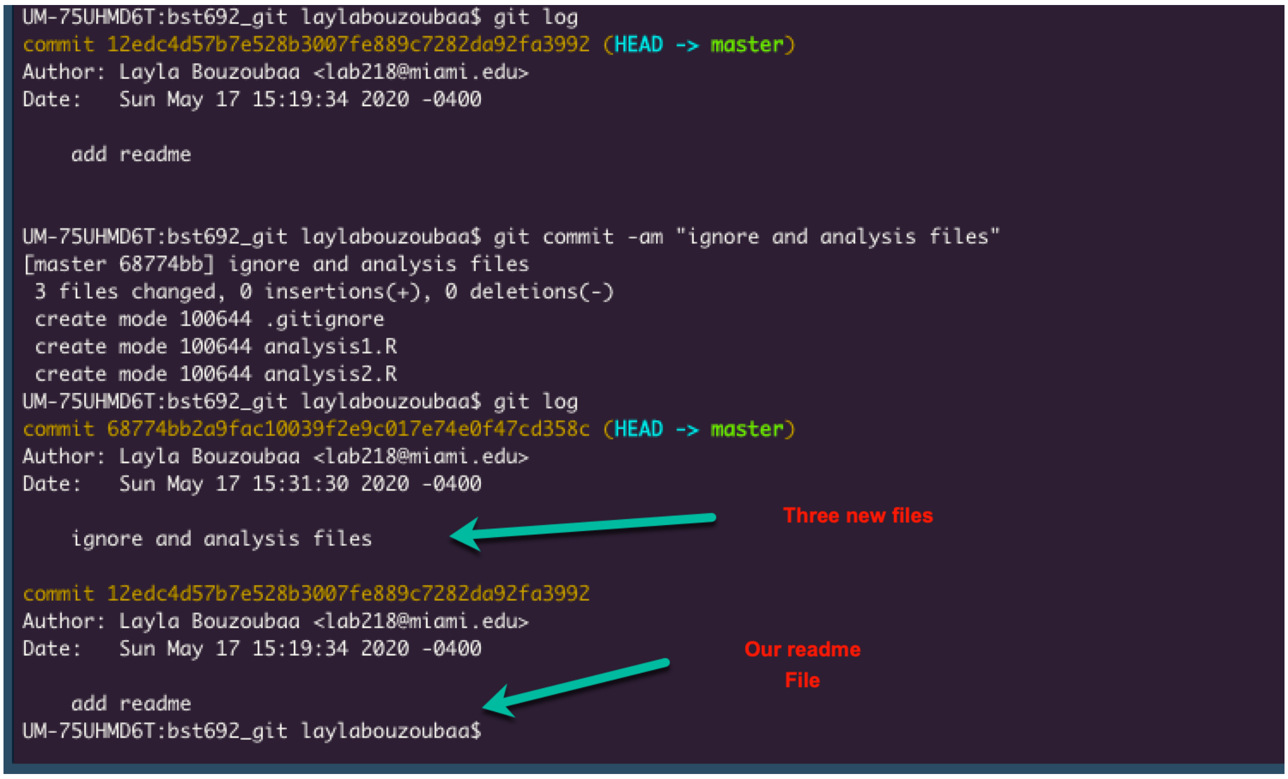
```
UM-75UHMD6T:~ laylabouzoubaa$ mkdir "bst692_git"
UM-75UHMD6T:~ laylabouzoubaa$ cd bst692_git/
UM-75UHMD6T:bst692_git laylabouzoubaa$ git init
Initialized empty Git repository in /Users/laylabouzoubaa/bst692_git/.git/
UM-75UHMD6T:bst692_git laylabouzoubaa$ ls -la
total 0
drwxr-xr-x   3 laylabouzoubaa  staff    96 May 13 13:52 .
drwxr-xr-x+ 56 laylabouzoubaa  staff  1792 May 13 13:51 ..
drwxr-xr-x  10 laylabouzoubaa  staff   320 May 13 13:52 .git
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
UM-75UHMD6T:bst692_git laylabouzoubaa$ touch "README.md"
UM-75UHMD6T:bst692_git laylabouzoubaa$ git add README.md
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.md

UM-75UHMD6T:bst692_git laylabouzoubaa$ git commit -m "create readme"
[master (root-commit) d649be8] create readme
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
nothing to commit, working tree clean
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

- You an track multiple files at once!
  - `git add .`
  - `.` will add all the new files created/modified but not added to git
  - Git will start tracking all three files
    - .gitignore, analysis1.R, analysis2.R
      - .gitignore starts with a "." and is therefore hidden
      - place files in the .gitignore and git will know to not track these files --> DATA files are a good thing to ignore



```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git add .
UM-75UHMD6T:bst692_git laylabouzoubaa$ git commit -m "add readme"
[master (root-commit) 12edc4d] add readme
 1 file changed, 1 insertion(+)
 create mode 100644 README.md


UM-75UHMD6T:bst692_git laylabouzoubaa$ touch ".gitignore" "analysis1.R" "analysis2.R"
UM-75UHMD6T:bst692_git laylabouzoubaa$ git add .
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   analysis1.R
        new file:   analysis2.R
```

- See our commit history
  - `git log`
  - OR `git log -1` to see just the last commit
- How can I do an add and a commit in one go?
  - `git commit -am "ignore and analysis files"`
- `git log`



```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git log
commit 12edc4d57b7e528b3007fe889c7282da92fa3992 (HEAD -> master)
Author: Layla Bouzoubaa <lab218@miami.edu>
Date:   Sun May 17 15:19:34 2020 -0400

    add readme


UM-75UHMD6T:bst692_git laylabouzoubaa$ git commit -am "ignore and analysis files"
[master 68774bb] ignore and analysis files
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 analysis1.R
 create mode 100644 analysis2.R
UM-75UHMD6T:bst692_git laylabouzoubaa$ git log
commit 68774bb2a9fac10039f2e9c017e74e0f47cd358c (HEAD -> master)
Author: Layla Bouzoubaa <lab218@miami.edu>
Date:   Sun May 17 15:31:30 2020 -0400

    ignore and analysis files

commit 12edc4d57b7e528b3007fe889c7282da92fa3992
Author: Layla Bouzoubaa <lab218@miami.edu>
Date:   Sun May 17 15:19:34 2020 -0400

    add readme
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

Scenario: After working on "analysis2.R", you realize that it is basically a better version of "analysis1.R". You only want to keep "analysis2.R" but like the idea of keeping "analysis1.R" as your code scratch pad (I do this all the time...). What do we do??

Well, at the end of the day, we want "analysis1.R" to be ignored by Git. But wait, we have already committed these files. How do we tell git to stop tracking "analysis1.R" so we can ignore it?

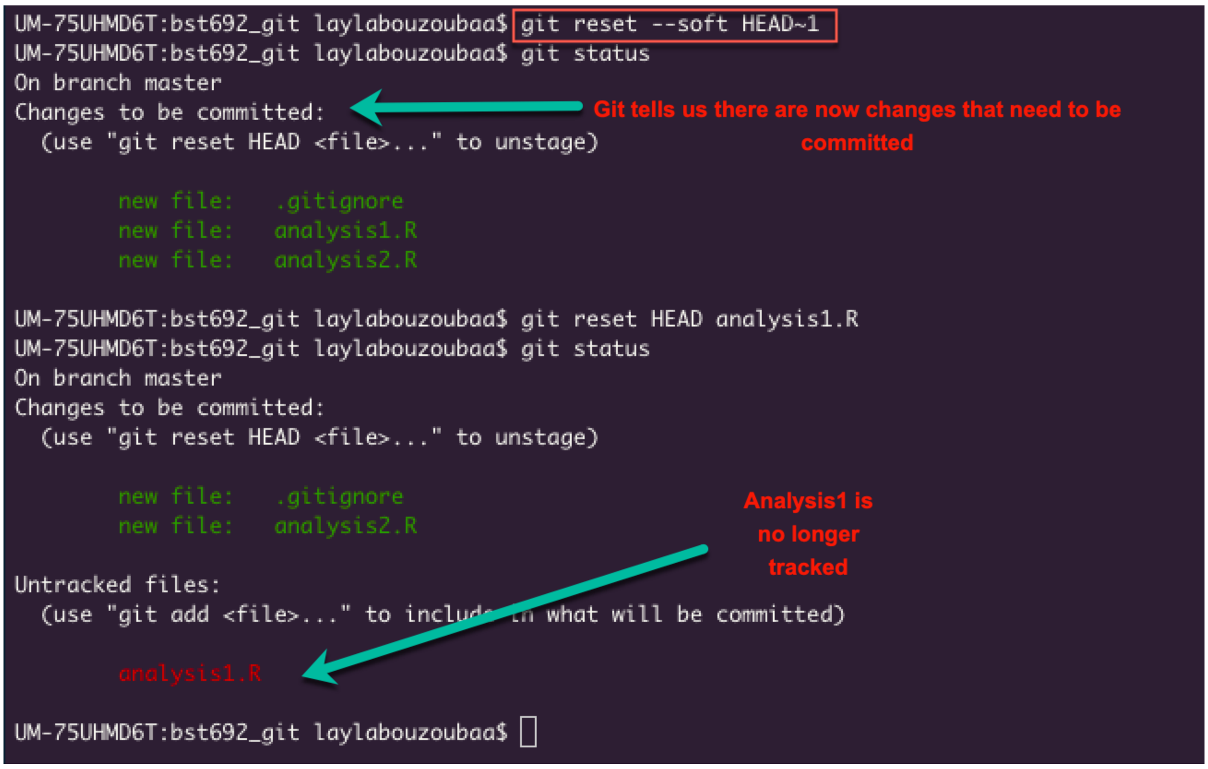
Like this!:

1. We need to undo the last commit which had "analysis1.R"
    1. `git reset --soft HEAD~1`
        1. `git reset` = undo commits
        2. `--soft` = preserves the changes made to the files

3. `HEAD~1` = only the LAST commit (remember in the log, the last commit appears at the top so the last commit will always be ~1)

2. Unstage "analysis.R"

   1. `git reset HEAD analysis1.R`

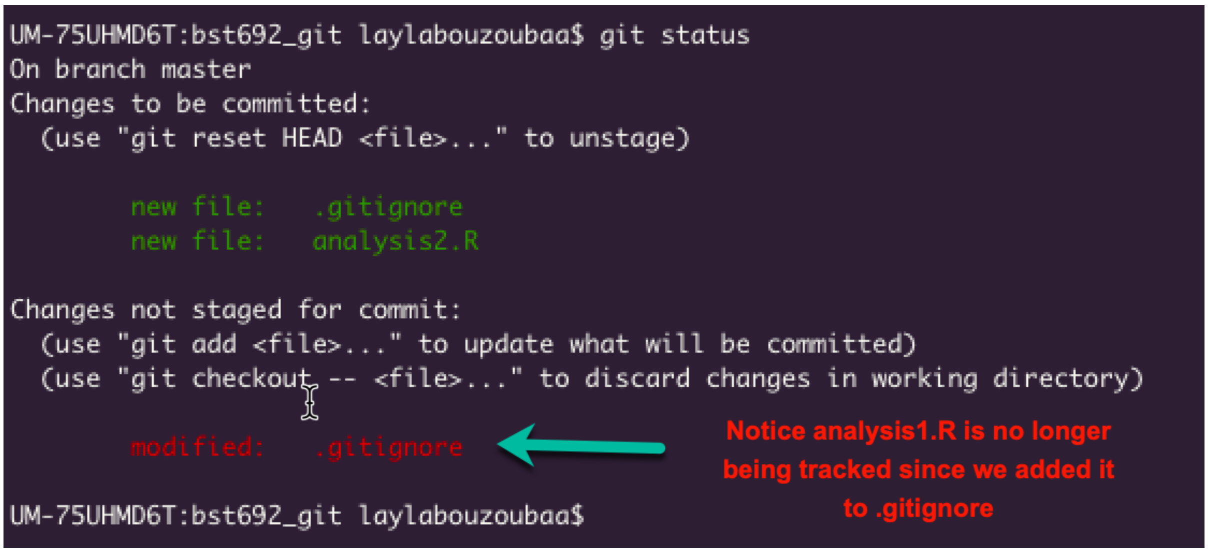


```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git reset --soft HEAD~1
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   analysis1.R
        new file:   analysis2.R

UM-75UHMD6T:bst692_git laylabouzoubaa$ git reset HEAD analysis1.R
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   analysis2.R

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        analysis1.R

UM-75UHMD6T:bst692_git laylabouzoubaa$
```

Git tells us there are now changes that need to be committed
Analysis1 is no longer tracked

3. Then we modify the .gitignore file

   1. add name of file, save
   2. `git status`



```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   analysis2.R

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

        modified:   .gitignore

UM-75UHMD6T:bst692_git laylabouzoubaa$
```

Notice analysis1.R is no longer being tracked since we added it to .gitignore

4. `git add .`

5. `git commit -m "move analysis1.R to gitignore"`

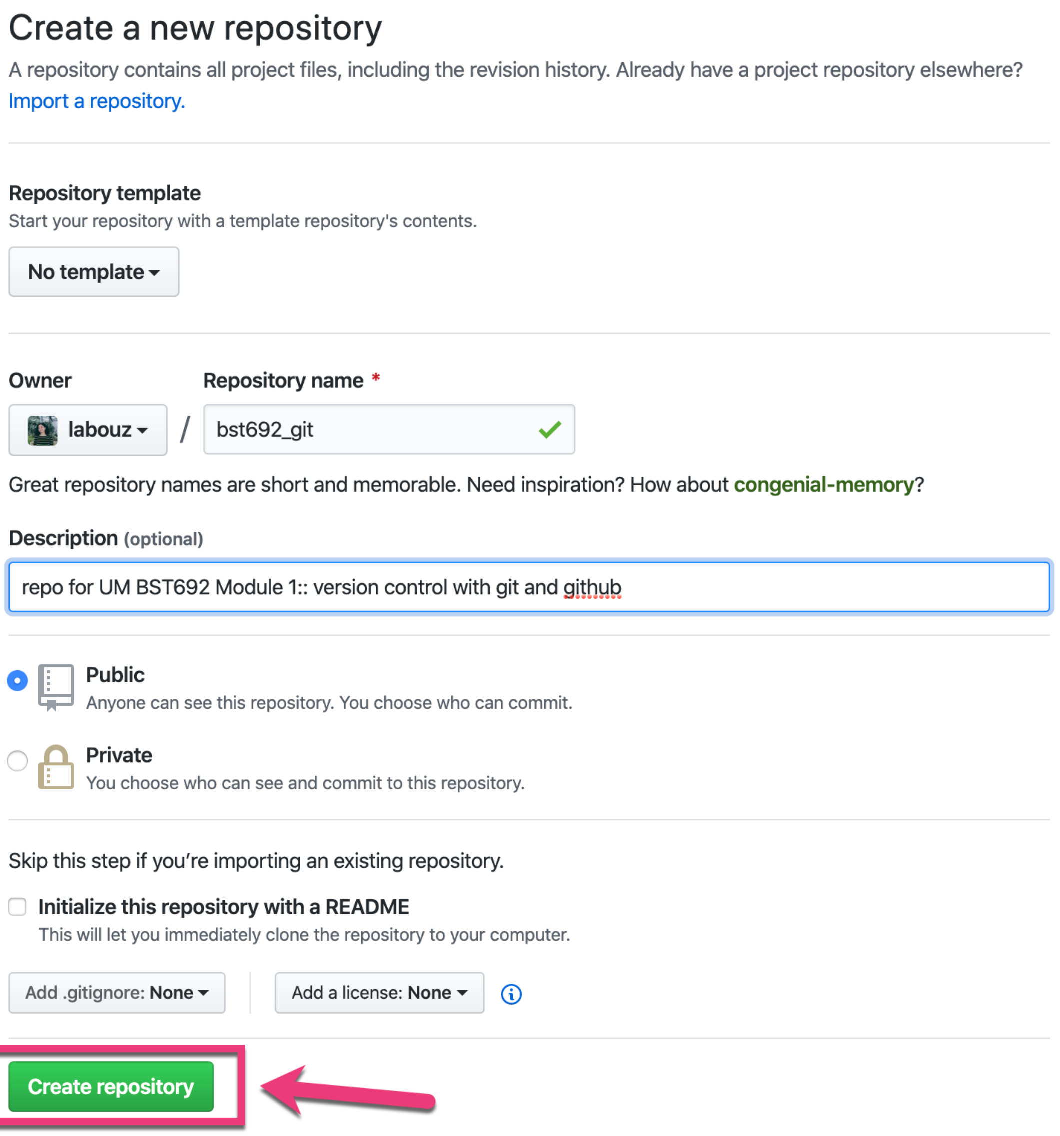
Remember, everything we have just done was done locally, on our computer. We still need to connect to a remote (online repository aka GitHub)!

Some notes on GitHub:

- Again, GH is a web-based hosting service for git
- Extends what we can do with git
- It's popular ~ 40 million developers worldwide
- Enables collaboration
- GH main features
  - Lots of code
  - Pull requests
    - Request their changes be merged with another branch, usually followed by a code review
  - Issues
    - Ways to mark bugs, feature enhancements, etc

Demo: Connect Repo

- Make a new repository
  - https://www.github.com



Create a new repository

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

Repository template
Start your repository with a template repository's contents.

No template

Owner: labouz / Repository name *: bst692_git

Great repository names are short and memorable. Need inspiration? How about congenial-memory?

Description (optional): repo for UM BST692 Module 1:: version control with git and github

[✓ Public] Anyone can see this repository. You choose who can commit.

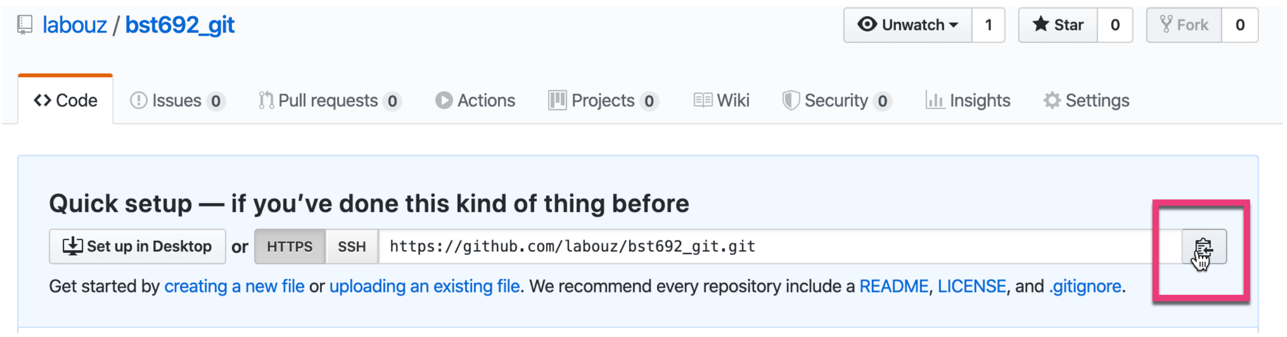
Private — You choose who can see and commit to this repository.

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

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

- Copy the git URL

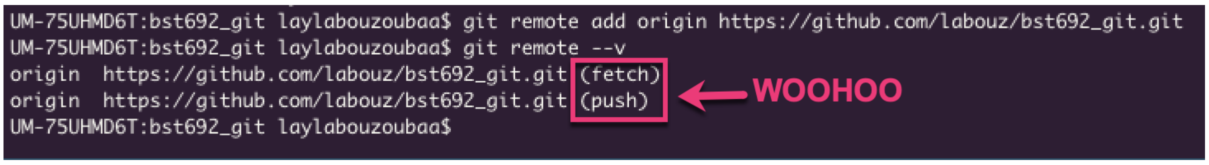

labouz / bst692_git

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/labouz/bst692_git.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

- Link it to local repository
    - `git remote add origin URL`
- Double check it's there
    - `git remote --v`



```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git remote add origin https://github.com/labouz/bst692_git.git
UM-75UHMD6T:bst692_git laylabouzoubaa$ git remote --v
origin  https://github.com/labouz/bst692_git.git (fetch)
origin  https://github.com/labouz/bst692_git.git (push)
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

- Let's push our work!
    - `git push origin master`
        - origin = remote
        - master = main tree

Scenario: Your colleague wants to join your project. You've been making all your changes to `master` and you're worried they may join and mess things up. What do you do?

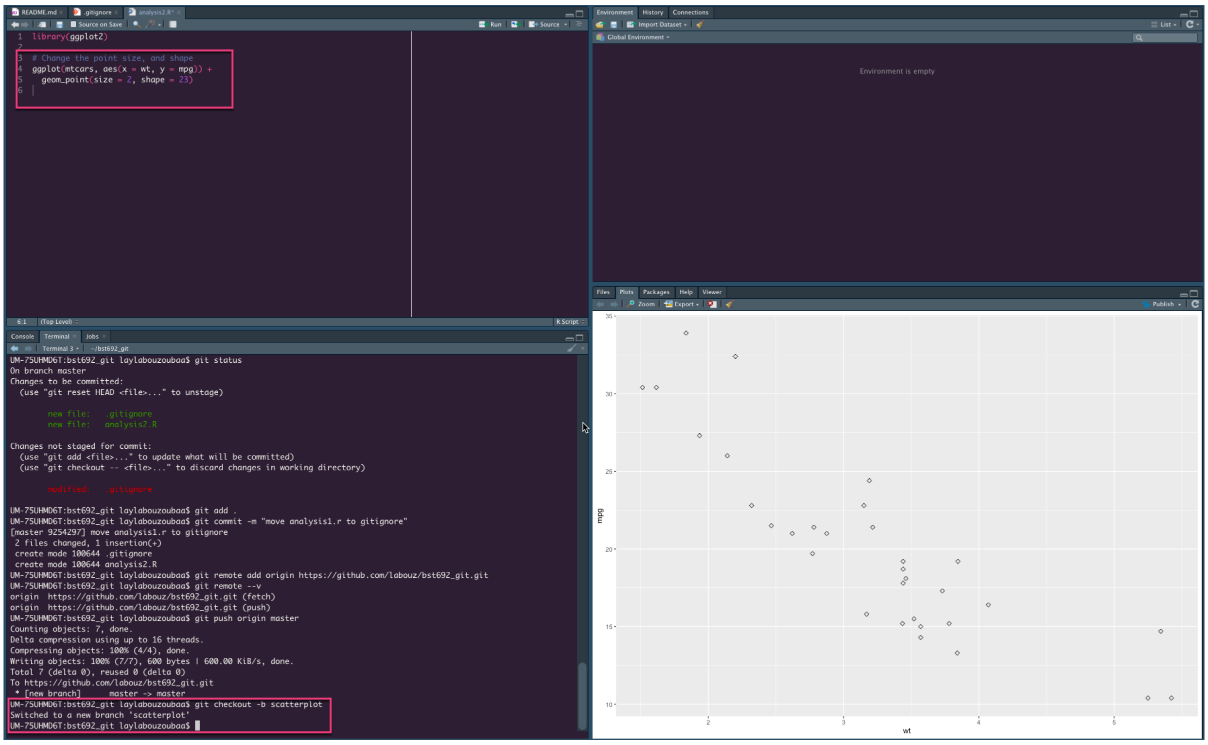
That's what branches were created for! You have already seen how to create a new branch in RStudio. Hopefully, you also remember why they are important. If not, let's review:

- Branches are important for code development
    - Wenever you need to create a new feature, fix a bug, or rewrite any of your code, it's a good idea to create a new branch so that none of your changes affect the "master" version of the code.
- How?
    - `git branch myNewFeature`
        - this command creates the new branch
        - it's good practie to create a new branch for each new feature and name your branch that feature
    - `git checkout myNewFeature`
        - this command switches you over from working on `master` to `myNewFeature`
    - OR simply adding `-b` will combine the two steps above (creating and switching over) into 1
        - `git checkout -b myNewFeature`

Demo: Branching

We are going to create a scatterplot in analysis2.R in our new branch `scatterplot`

- `git checkout -b scatterplot`
- modify analysis2.R
  - ggplot2
    - x = wt
    - y = mpg
  - geom_point



```r
library(ggplot2)

# Change the point size, and shape
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(size = 2, shape = 23)

```

```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   .gitignore
        new file:   analysis2.R

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

        modified:   .gitignore

UM-75UHMD6T:bst692_git laylabouzoubaa$ git add .
UM-75UHMD6T:bst692_git laylabouzoubaa$ git commit -m "move analysis1.r to gitignore"
[master 9254297] move analysis1.r to gitignore
 2 files changed, 1 insertion(+)
 create mode 100644 .gitignore
 create mode 100644 analysis2.R
UM-75UHMD6T:bst692_git laylabouzoubaa$ git remote add origin https://github.com/labouz/bst692_git.git
UM-75UHMD6T:bst692_git laylabouzoubaa$ git remote --v
origin  https://github.com/labouz/bst692_git.git (fetch)
origin  https://github.com/labouz/bst692_git.git (push)
UM-75UHMD6T:bst692_git laylabouzoubaa$ git push origin master
Counting objects: 7, done.
Delta compression using up to 16 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 600 bytes | 600.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/labouz/bst692_git.git
 * [new branch]      master -> master
UM-75UHMD6T:bst692_git laylabouzoubaa$ git checkout -b scatterplot
Switched to a new branch 'scatterplot'
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

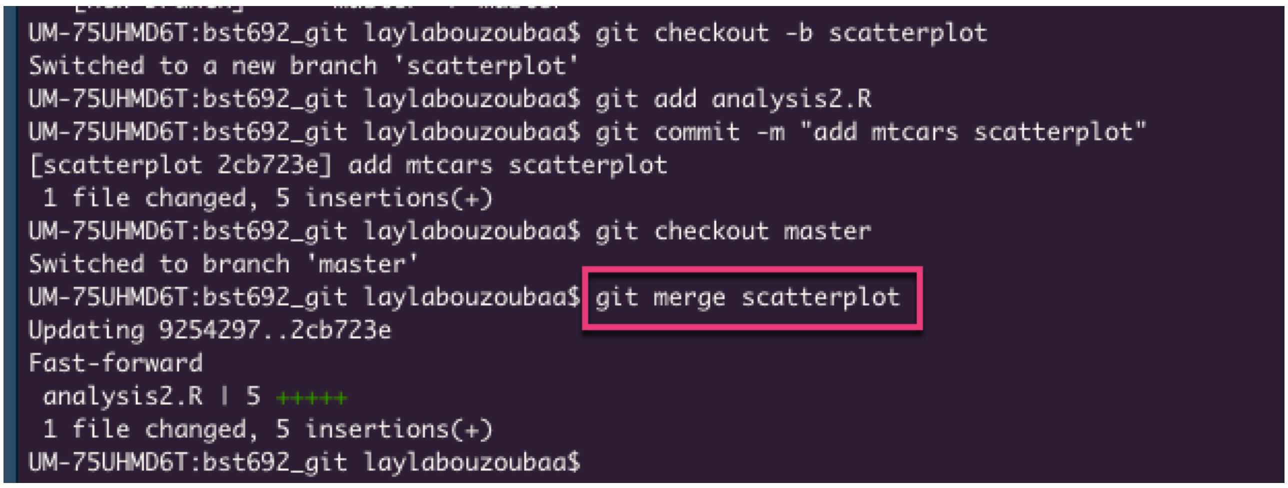
- Save
- `git add analysis2.R`
- `git commit -m "add mtcars scatterplot"`

Cool, but what if we want this version of "analysis2.R" to be the final version (for the moment)?

Demo: Merging

- switch back to `master`
  - `git checkout master`

- `git merge scatterplot`



```
UM-75UHMD6T:bst692_git laylabouzoubaa$ git checkout -b scatterplot
Switched to a new branch 'scatterplot'
UM-75UHMD6T:bst692_git laylabouzoubaa$ git add analysis2.R
UM-75UHMD6T:bst692_git laylabouzoubaa$ git commit -m "add mtcars scatterplot"
[scatterplot 2cb723e] add mtcars scatterplot
 1 file changed, 5 insertions(+)
UM-75UHMD6T:bst692_git laylabouzoubaa$ git checkout master
Switched to branch 'master'
UM-75UHMD6T:bst692_git laylabouzoubaa$ git merge scatterplot
Updating 9254297..2cb723e
Fast-forward
 analysis2.R | 5 +++++
 1 file changed, 5 insertions(+)
UM-75UHMD6T:bst692_git laylabouzoubaa$
```

Wait, aren't we missing something??

YES!

We still need to push our new changes to the remote (GitHub) so our colleague can see our changes too 😃

- `git push origin master`

That's it! BUT, we want to be tidy and not have our repos cluttered with so many unused branches so it's good practice to delete a branch once you're comfortable with your changes and want to move on.

Demo: Deleting a branch

- Local:
  - `git branch -d scatterplot`
- Remote:
  - `git push origin --delete scatterplot`
  - Note: if you noticed carefully, we never actually pushed our branch to the remote in this exmple. We did our merge to master LOCALLY and then pushed those changes to the remote so GitHub never saw our branch `scatterplot` 😃
    - IRL: you will likely make several commits and push to a branch before merging. Once you push to the branch then you will see that branch on GitHub!