

Reproducible Research Practices for Economists

Mindy L. Mallory

October 24, 2017

Questions for the Audience

How many of your research folders look like this?

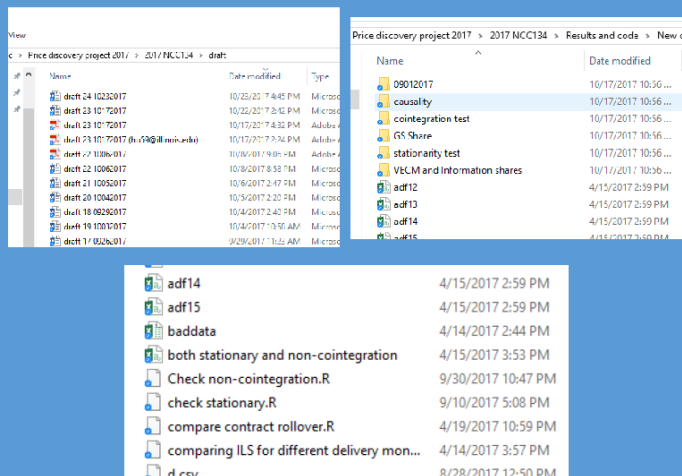
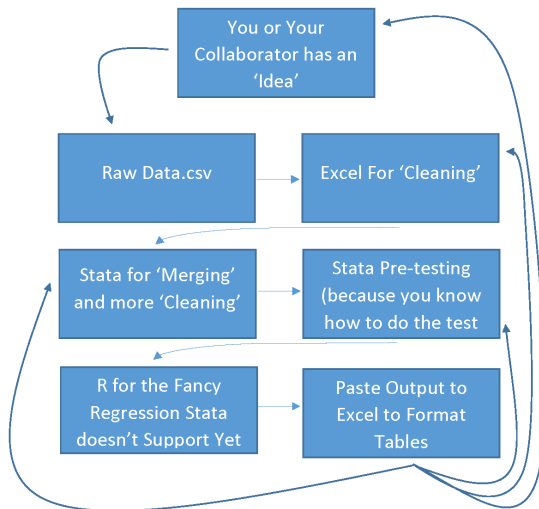


Figure 1: Picking on Zhepeng

How many of you have a research work flow that looks like this?



Questions for the Audience

How many of you would rather die than have to reproduce a table from a paper you published 2 years ago?

Questions for the Audience

Do you wake up in a cold sweat dreaming that Reviewer number 2 asked you to update your data-set (perform robustness test, etc) and you couldn't even reproduce your original results?

Questions for the Audience

Students, have you ever purposely obfuscated your code figuring if your professor can't follow it they can't criticize it?

Questions for the Audience

Have you ever lost data between submission and being asked to revise and resubmit and then you had to go and REPURCHASE!!! said data?

Questions for the Audience

Have you ever lost an entire paper due to the Word file becoming corrupted then you thought you salvaged the paper through document recovery but then it got rejected because you missed some weird characters from the file corruption and reviewer number 2 recommended rejecting your paper because the authors were 'careless' to allow the weird characters to remain the document?

I can say yes to all of these questions!

But I got tired of being nervous all the time!

There is a better way!

Reproducible research with R, RStudio, RMarkdown, Knitr, and Github

- R - is awesome statistical computing software (open source and free!)
- Rstudio - is an awesome integrated development environment (program making it convenient to work with R); also open source and free

Reproducible research with R, RStudio, RMarkdown, Knitr, and Github

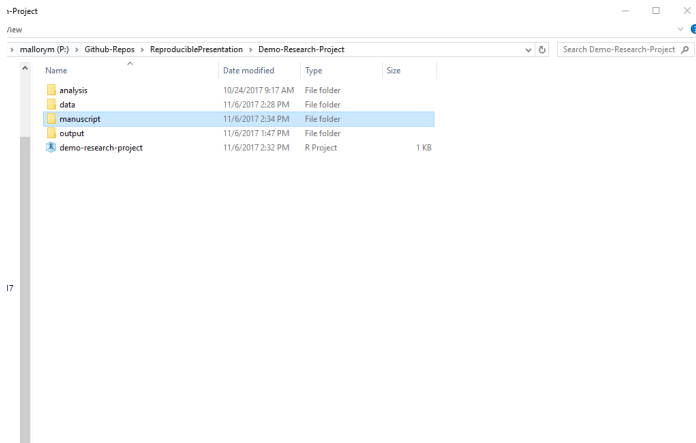
- RMarkdown is a kind of markup language supported by RStudio that uses **Knitr** to weave statistical analysis and results into beautifully formatted documents.
 - ▶ Written in plaintext, it understands latex code and documents can be rendered into many different output formats
 - ★ PDF
 - ★ Beamer
 - ★ HTML
 - ★ Word*

Reproducible research with R, RStudio, RMarkdown, Knitr, and Github

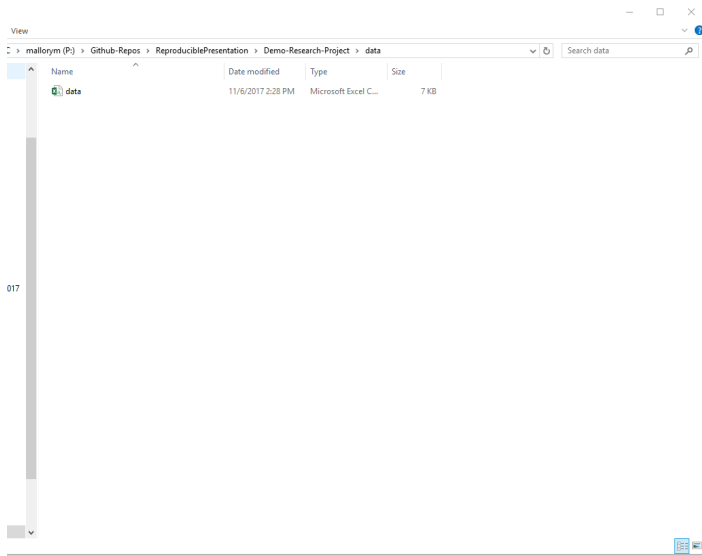
- Github - is a cloud-based repository that is great at versioning (it was designed by and for software developers)

The Basics - Set up a clean, reproducible project repository

- RStudio Rule #1 - use projects!
- Never change the working directory
- Once you have created a project, the working directory is automatically set to this file path

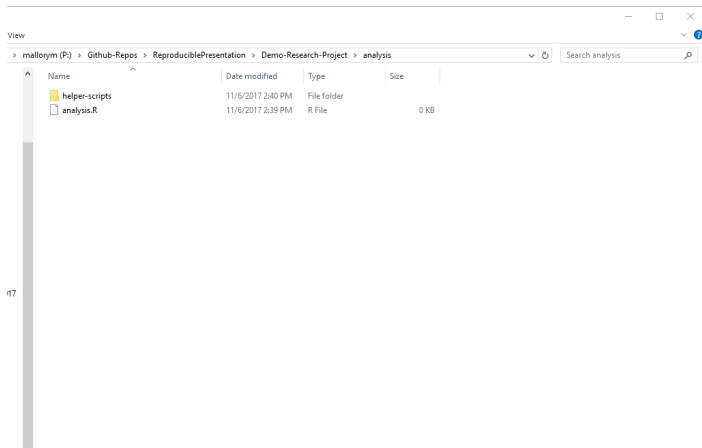


The Basics - Put your raw data in the 'data' folder and never touch again



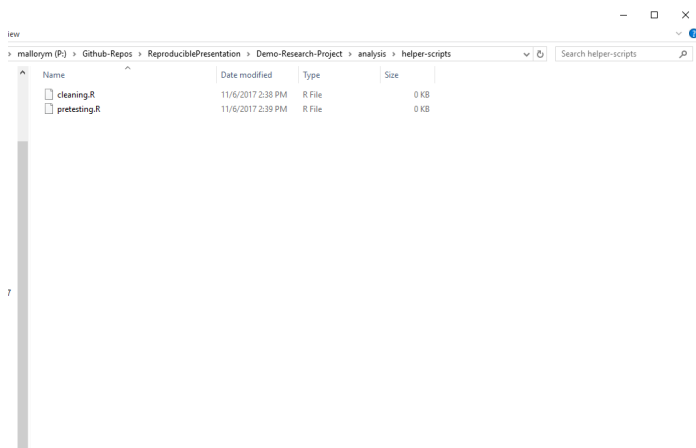
The Basics - Organize Scripts

- Document what each script does
- If your project requires an elaborate 'readme.txt' with instructions about which scripts to run and in what order, your work is not reproducible.



The Basics - Organize Scripts

- Document what each script does
- If your project requires an elaborate 'readme.txt' with instructions about which scripts to run and in what order, your work is not reproducible.



Data Analysis - Cleaning

Your analysis may involve 'cleaning' raw data.

- May be aggregating many individual files
- Dealing with missing data
- Merging two or many large datasets

This type of activity should be done by the `cleaning.R` script that takes raw data files and makes them useful.

If at all possible, do not save intermediate cleaned data. Run scripts that build from raw data everytime so you know it is reproducible.

Look at `cleaning.R`

Data Analysis - Pretesting

Similarly, you may need to check for stationarity or do other common diagnostic tests that inform model choice.

This file will take cleaned data from `cleaning.R` and perform diagnostics. The tests will create R objects that can be called and inserted into manuscript results.

Look at `pretesting.R`

Data Analysis - Fit Main Model

Then, your main analysis can be performed in `analysis.R`. This script will fit model and the output will be R objects that can be inserted to display results directly into tables and text of your manuscript.

Look at `analysis.R`

Write Paper in RMarkdown

RMarkdown is an easy to use way to create reproducible reports that can be rendered to many formats.

- Accepts Latex commands for math equations and other formatting
- Supports reference management with bibtex
- Execute R scripts right in the document and incorporate the results into your document

Look at manuscript.Rmd

Commit and Push to Github.com

Unlike Dropbox and Box that automatically watch for changes and upload new file versions to cloud storage, you have to manually commit changes and send them to the remote repository.

Can be tricky, until you get in the habit of committing and pushing, similar to how we automatically have the reflex to save a file every so often.

Advantage - If your file gets corrupted, it won't overwrite all your copies with the corrupted version (this happened to me).

Github is a time machine, you can go back and recover your files at any state of the repository.

Commit and Push to Github.com

Git Basic Steps:

- Stage - means get changes ready to be committed to the repository
- Commit - means they are 'permanately' part of the repository record
- Push - sends you committed changes to the remote repository for safe keeping forever.

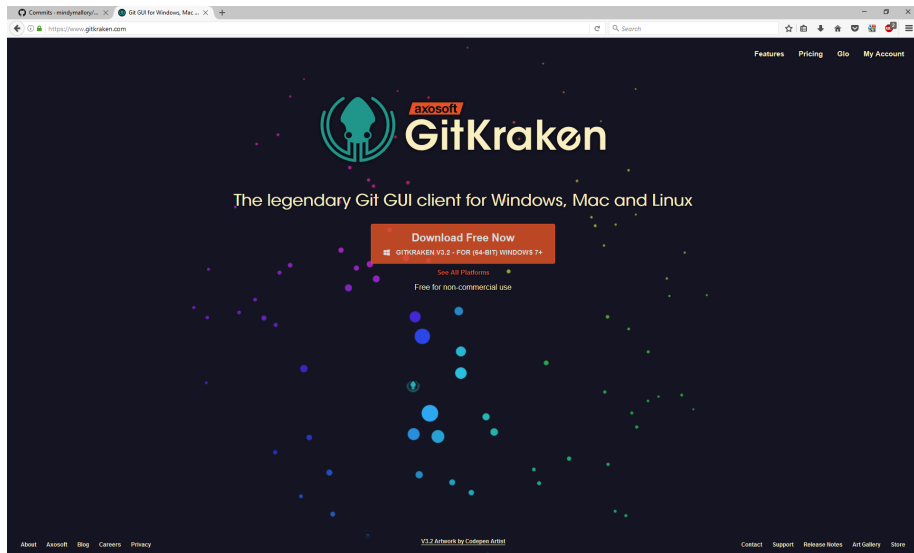
Git Clients

Git can be run in a git command line interface (no idea how this works)

Git is integrated in RStudio, and for simple changes it often works ok; however, it can be buggy.

Git Clients

Gitkraken is a nice GUI that I find intuitive and easy to use.



Gitkraken - Stage

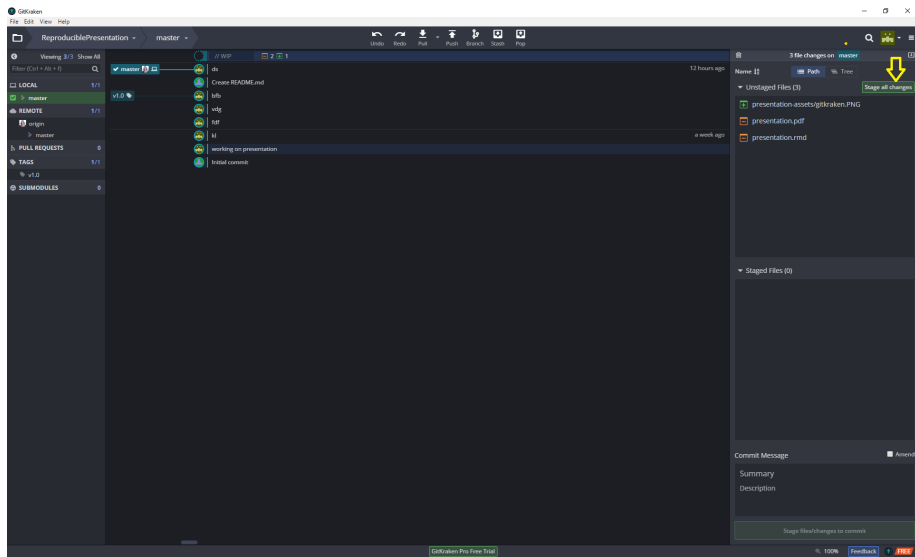


Figure 8: Stage

Gitkraken - Commit

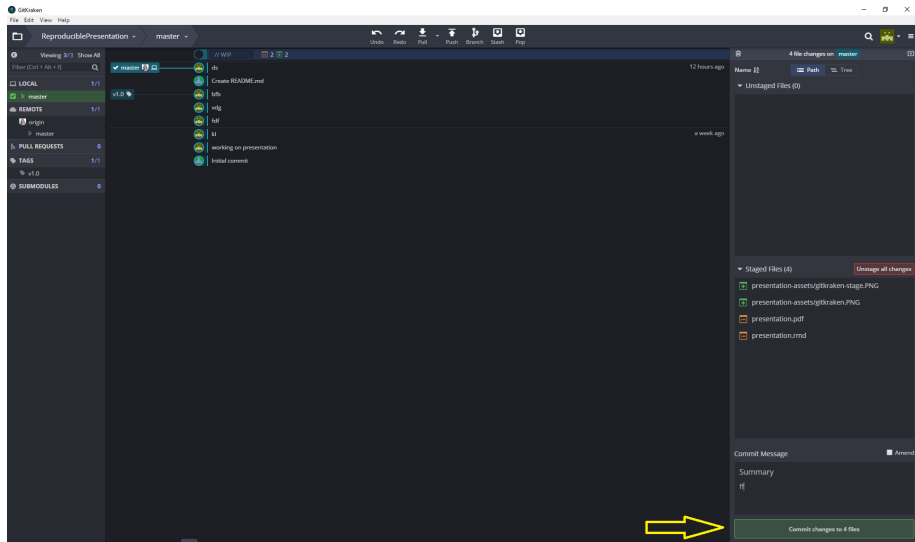


Figure 9: Commit

Gitkraken - Push

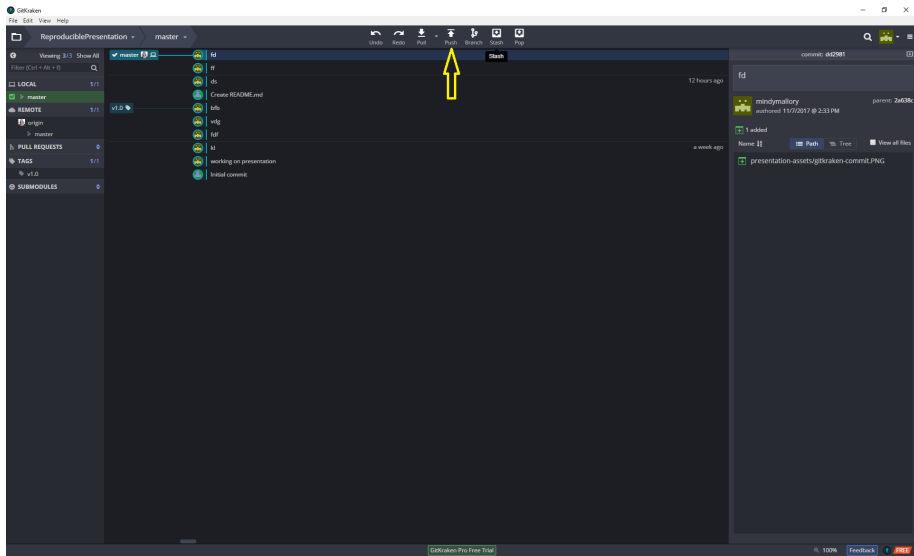
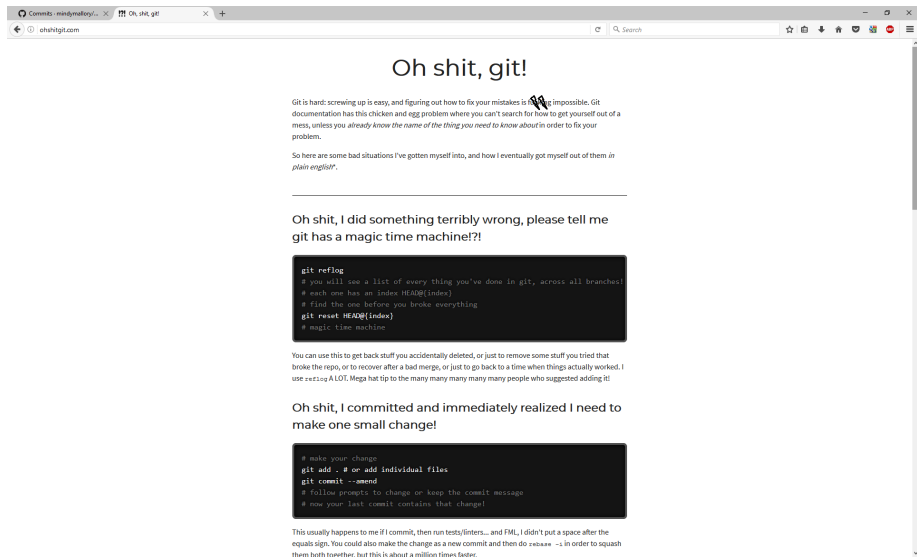


Figure 10: Push

If you mess up, there is help on the internet



The screenshot shows a web browser window with the address bar displaying 'ohshitgit.com'. The page title is 'Oh shit, git!'. The main content area has a heading 'Oh shit, git!' followed by a paragraph explaining that Git is hard and that documentation has a chicken and egg problem. Below this is a section titled 'Oh shit, I did something terribly wrong, please tell me git has a magic time machine?!' which contains a code block for 'git reflog' and 'git reset HEAD@{index}'. Another section titled 'Oh shit, I committed and immediately realized I need to make one small change!' contains a code block for 'git add', 'git commit --amend', and 'git commit'. The page ends with a paragraph about using 'git revert' or 'git reset' to undo changes.

Commits - mindymallory/... x Oh, shit, git! x +

ohshitgit.com

Oh shit, git!

Git is hard: screwing up is easy, and figuring out how to fix your mistakes is ~~fixing~~ impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, unless you *already know the name of the thing you need to know about* in order to fix your problem.

So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*!.

Oh shit, I did something terribly wrong, please tell me git has a magic time machine?!

```
git reflog
# you will see a list of every thing you've done in git, across all branches!
# each one has an index HEAD@{index}
# find the one before you broke everything
git reset HEAD@{index}
# magic time machine
```

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried that broke the repo, or to recover after a bad merge, or just to go back to a time when things actually worked. I use `reflog` A LOT. Mega hat tip to the many many many many people who suggested adding it!

Oh shit, I committed and immediately realized I need to make one small change!

```
# make your change
git add . # or add individual files
git commit --amend
# follow prompts to change or keep the commit message
# now your last commit contains that change!
```

This usually happens to me if I commit, then run tests/linters... and FML, I didn't put a space after the equals sign. You could also make the change as a new commit and then do `rebase -i` in order to squash them both together, but this is about a million times faster.

Commit and Push to Github.com

Show the Github time machine

Use 'Releases' to Mark Important Milestones in Paper's Progress

Since Github was developed by and for software developers, 'Releases' are built in.

- Releases signify specific points in the repository's commit history
- i.e. v2.3.0 of your software
- Convenient for important versions of your paper
- AAEA invited paper
- AJAE submission
- JARE submission
- etc. . .

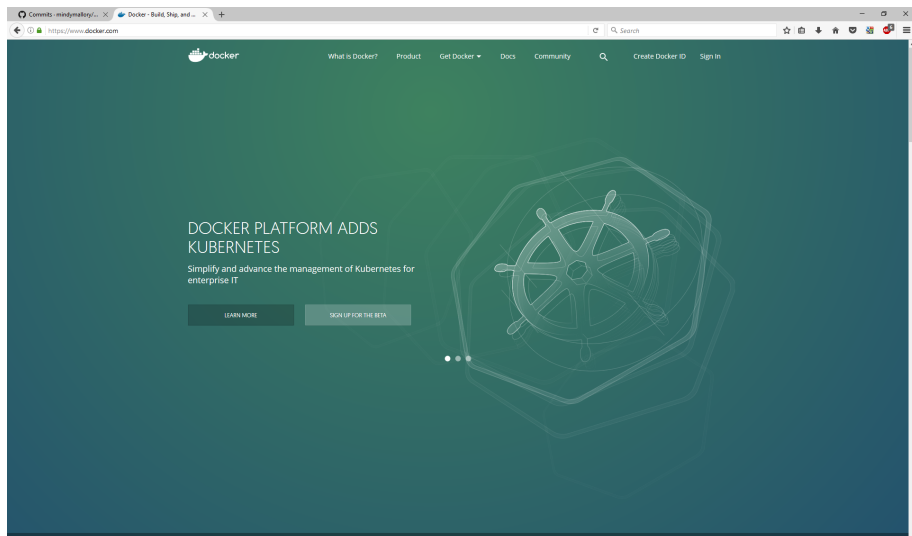
Use 'Releases' to Mark Important Milestones in Paper's Progress

This is useful.

- Say you have a table in your conference paper.
- You cut it for the AJAE submission
- It gets rejected and you send it to JARE
- Reviewer #2 asks you do exactly this and R&R back to JARE
- You can go to the AAEA 'release' and recover exactly the state of your repository where you have working code that generates this table
- Easily incorporate it back into the more recent version

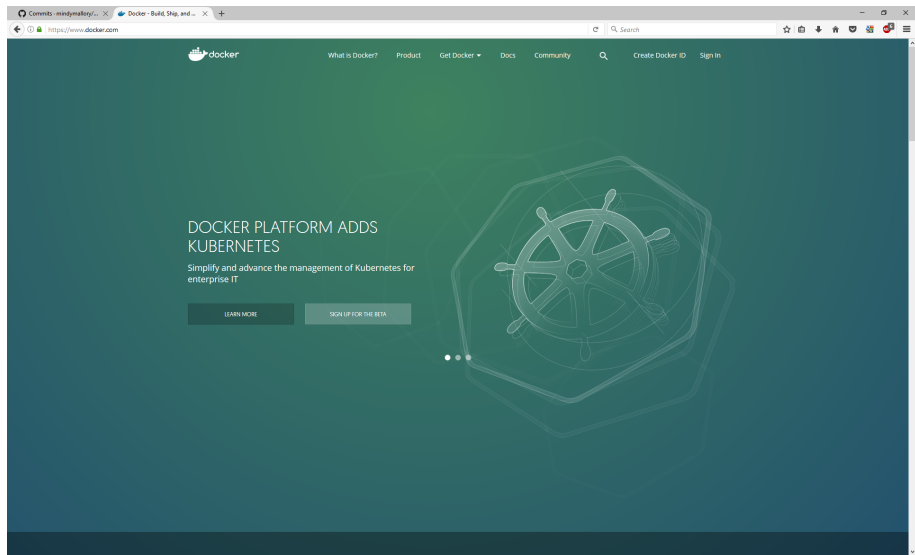
How to Prevent Your Code from Breaking

Sometimes, even if you follow these practices, R package updates will break your code!



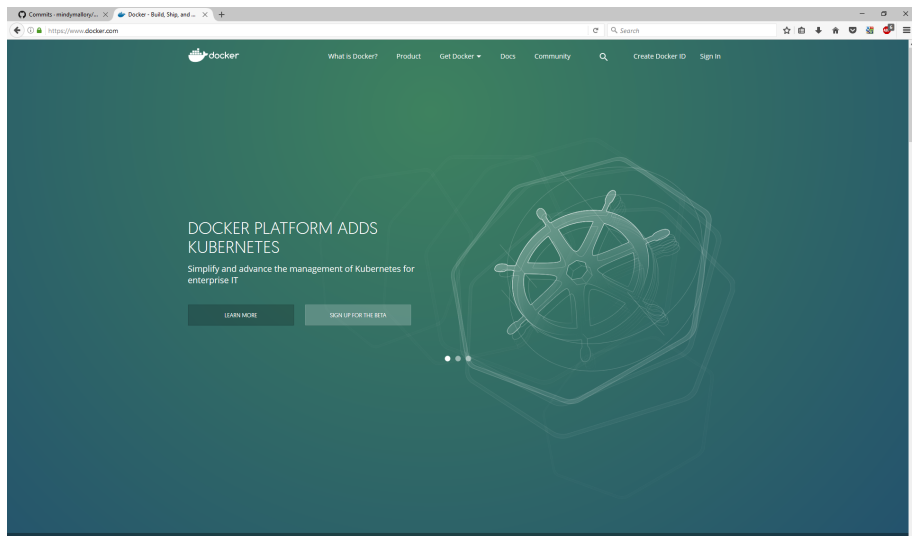
How to Prevent Your Code from Breaking

I haven't learned Docker yet. . .



How to Prevent Your Code from Breaking

But Docker allows you to keep a copy of R and RStudio exactly as it is today, so your code can never break due to an update.



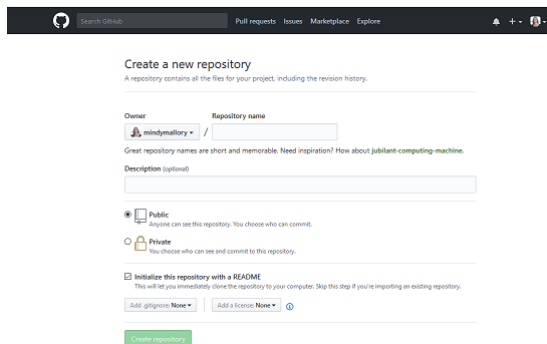
The Basics - Get Set up On Your Own

- Install R, RStudio, Git, and Gitkraken
- Install the following packages in R by executing the following commands in the RStudio console:

```
install.packages("xts")  
install.packages("tseries")  
install.packages("tsDyn")  
install.packages("broom")  
install.packages("vars")
```

The Basics - Get Set up On Your Own

- Create a new repository on Github.com
- Choose a meaningful repository name
- Be sure to initialize with a Readme file by clicking the checkbox (somehow it helps RStudio and GitHub set an initial connection)
- After creating the repository



The screenshot shows the GitHub web interface for creating a new repository. At the top is a dark navigation bar with the GitHub logo, a search bar, and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this is the 'Create a new repository' section. It includes a sub-header 'Create a new repository' and a note: 'A repository contains all the files for your project, including the revision history.' The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' dropdown is set to 'mindymallory'. The 'Repository name' field is empty. Below these is a hint: 'Great repository names are short and memorable. Need inspiration? How about *joyful-computing-machine*.' There is a 'Description (optional)' text area. The 'Public' radio button is selected, with the text 'Anyone can see this repository. You choose who can commit.' The 'Private' radio button is unselected, with the text 'You choose who can see and commit to this repository.' There is a checkbox for 'Initialize this repository with a README' which is checked. Below this are two dropdown menus: 'Add gitignore: None' and 'Add a license: None'. At the bottom is a green 'Create repository' button.

Figure 15:

The Basics - Get Set up On Your Own

- After creating the repository, click 'Clone or Download' and copy the link to the repository.

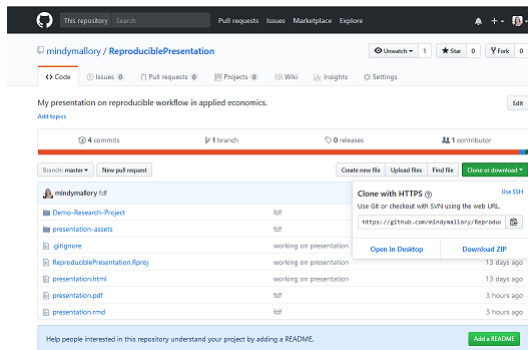
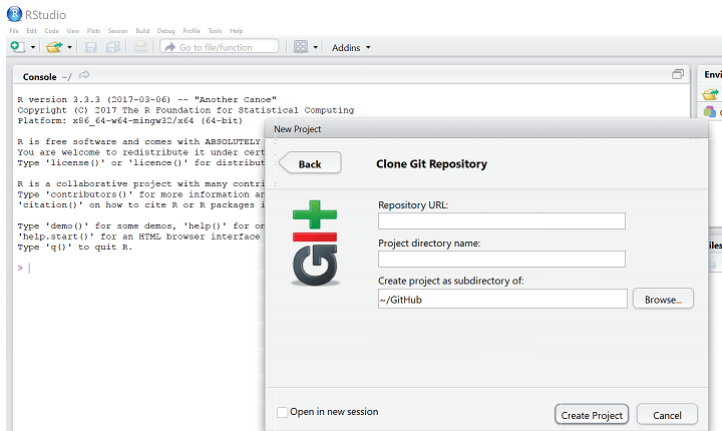


Figure 16:

The Basics - Get Set up On Your Own

- Open up RStudio and navigate through 'File' -> 'New Project'
- Choose 'Version Control' -> 'Git'
- Then paste the link you copied from github.com into 'Repository URL' and click 'Create Project'



The Basics - Get Set up On Your Own

Now your RStudio project is connected to Github. Periodically commit your local changes to the Github repository.

- Stage Changes
- Commit Changes
- Push Changes