PSYC 259: Principles of Data Science

Week 2: File Organization and Workflow

Today

1. Project structure principles
   a. File/folder organization
   b. Version control
2. Advice: How to get programming help
3. BREAK
4. R language basics and importing tutorial

Project Structure Principles

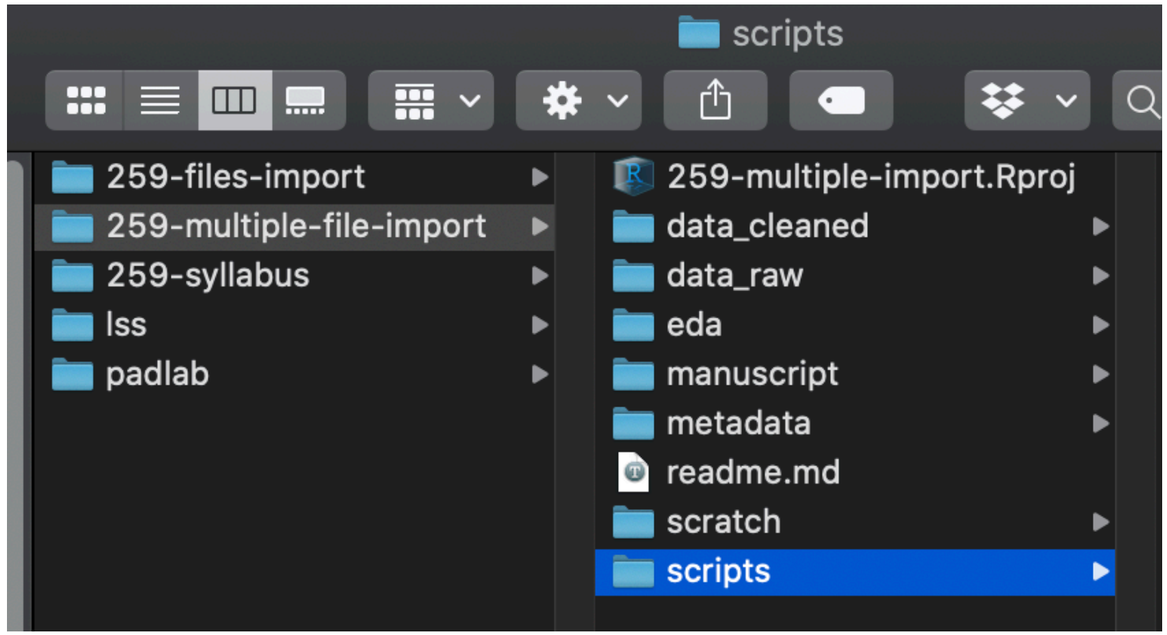
Two principles of project workflow

1. Folder organization creates rules and defines a workflow; establishes a location from which to build relative file paths
2. Version control tracks file history without duplication/clutter; allows for collaboration/derivation/experimentation

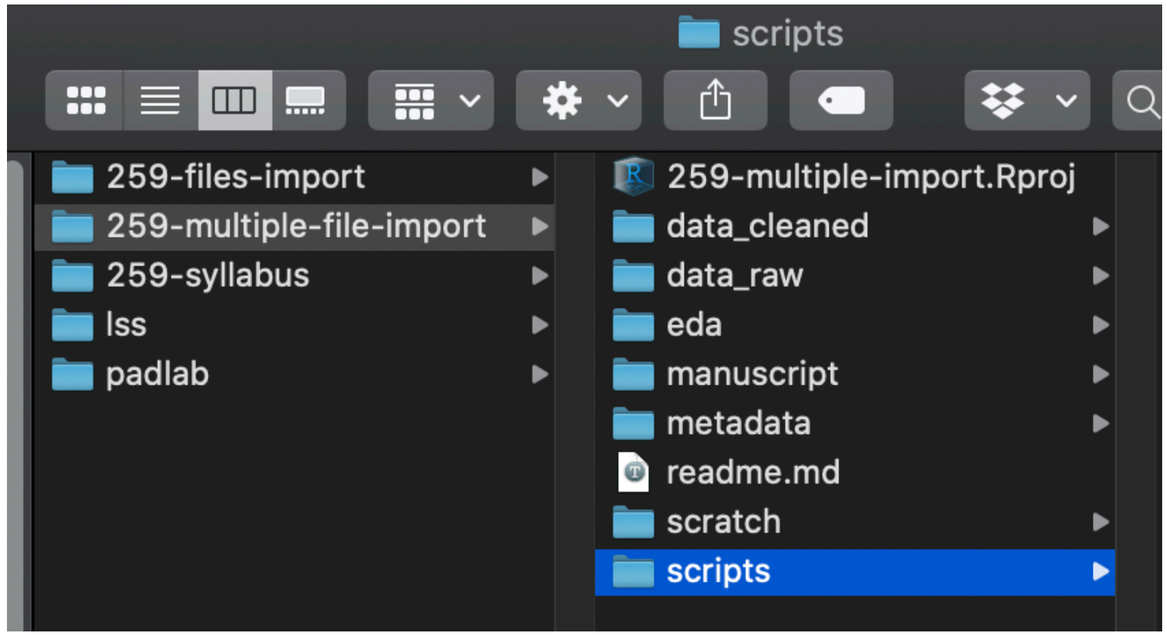
What principles should guide project structure?

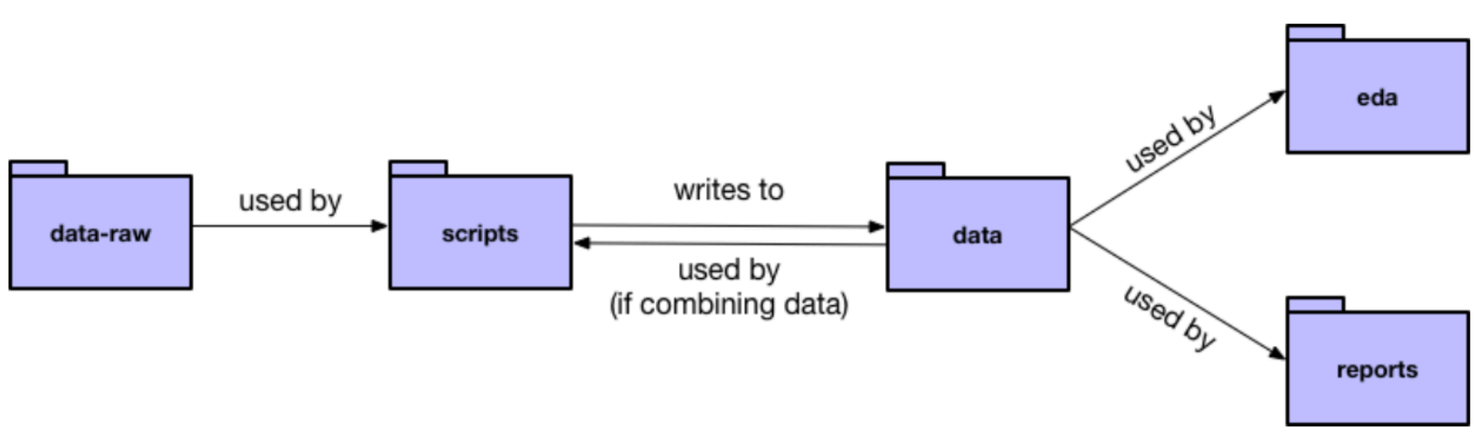
#1 File/folder organization

Folders should organize similar file **types** (w/in a root project folder)



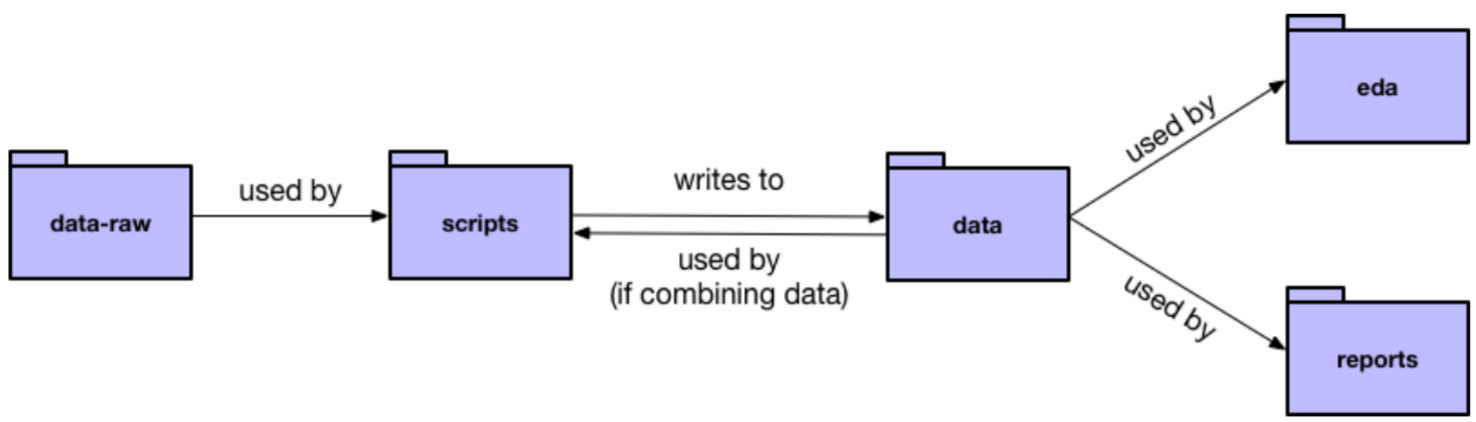
Folders should organize similar file **types** (w/in a root project folder)





Avoid having more nodes in this chart than are necessary

- Every intermediate step incurs a cost of maintenance
  - Think about what has to be re-run if data-raw changes
- Multiple endpoints don't incur cost



What is an R Studio Project?

- A project folder (directory) with some files that keep tabs on your command history (.RProj)
  - Open the RProj file to open the project
  - Can turn existing folders into projects
- What is real (e.g., persists after shutting down R)?
  - Not real: objects/data frames in your workspace
  - Real: data files and the scripts (.R files) used to work with them

Accessing files from your working directory

- Your RStudio Project folder is your default *working directory*
  - working directory: where R can find files
  - getwd() probably gives you something ugly like `/Users/johnfranchak/Documents/GitHub/project_folder`
- *Absolute file paths* like that should be avoided at all costs!
  - Absolute file paths don't transfer between computers with different user names or different operating systems
  - Bad for reproducibility, extensibility, and sharing
- But don't you need them to get into all of those directories you just told me I need to use?

Relative files paths 💪💪💪

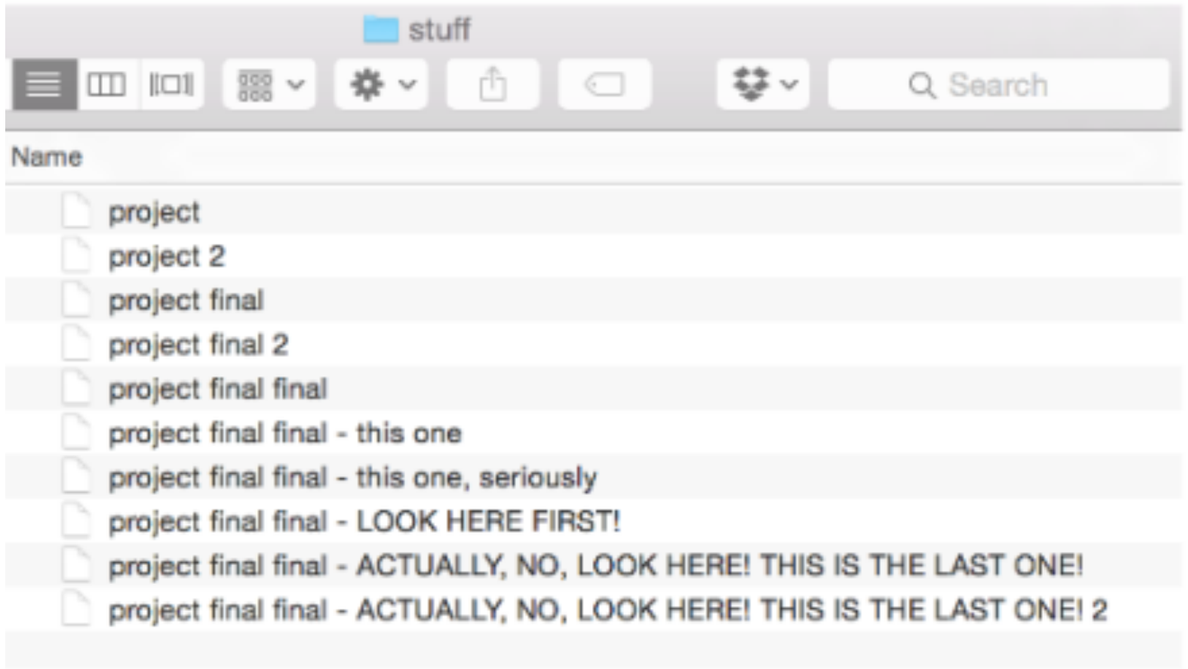
- Just tell R what to look for relative to your project (working) directory!
- append subfolder name to filename
  - “folder1/filename”
- *here* package detects the project directory and composes filenames from/to any folder
  - `here(“folder1”,”folder2”,”filename")`

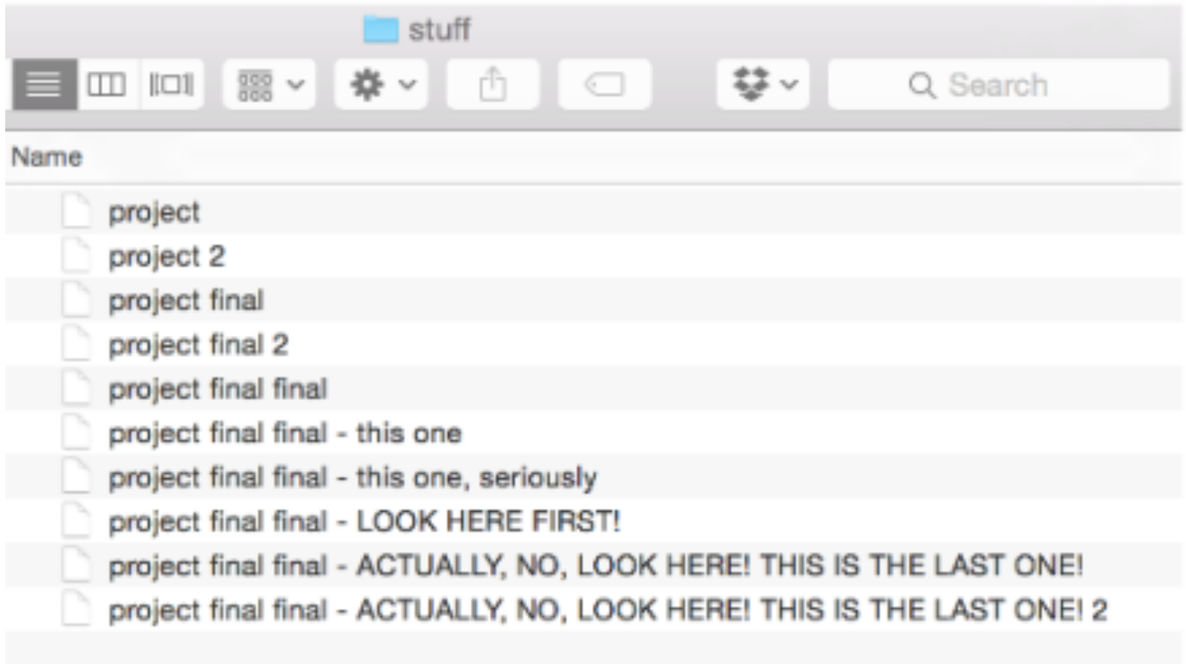
Other considerations

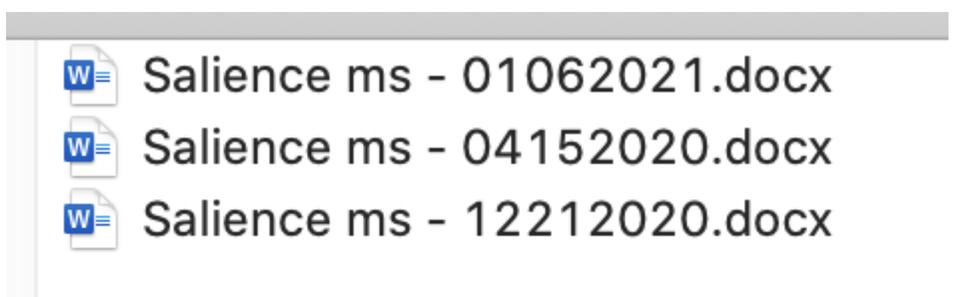
- Not everything can be automated, but you can try to limit human data entry to a single master table
  - Track notes about sessions, inclusion/exclusion info
  - Keep as part of project metadata, and use it to direct your scripts to pull the ‘right’ data
- Not every project can be contained in a local directory on a single computer (or on github)
  - Large datasets might need other solutions
  - “data_raw” might need to be “data_slightly_cooked”

What principles should guide project structure?

#2 Version Control


stuff
Search
Name
project
project 2
project final
project final 2
project final final
project final final - this one
project final final - this one, seriously
project final final - LOOK HERE FIRST!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE! 2


stuff
Search
Name
project
project 2
project final
project final 2
project final final
project final final - this one
project final final - this one, seriously
project final final - LOOK HERE FIRST!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE!
project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE! 2


Salience ms - 01062021.docx
Salience ms - 04152020.docx
Salience ms - 12212020.docx

stuff

Search

Name

- project
- project 2
- project final
- project final 2
- project final final
- project final final - this one
- project final final - this one, seriously
- project final final - LOOK HERE FIRST!
- project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE!
- project final final - ACTUALLY, NO, LOOK HERE! THIS IS THE LAST ONE! 2

- Salience ms - 01062021.docx
- Salience ms - 04152020.docx
- Salience ms - 12212020.docx

- SUBMISSION1-JEP
- SUBMISSION2-APP
- SUBMISSION3-APP
- SUBMISSION4-APP

- 2015-11-13-Recalibration.docx
- 2015-12-08-Recalibration.docx
- 2015-12-09-Recalibration.docx
- 2015-12-10-Recalibration.docx
- 2016-01-12-Recalibration.docx
- 2016-01-13-Recalibration.docx
- 2016-01-14-Recalibration.docx
- 2016-01-15-Recalibration.docx
- 2016-01-18-Recalibration.docx
- 2016-01-20-Recalibration.docx
- 2016-01-21-Recalibration.docx
- 2016-01-22-Recalibration.docx
- 2016-01-25-Recalibration.docx
- 2016-01-26-Recalibration.docx
- 2016-01-27-Recalibration.docx
- 2016-01-28-Recalibration.docx
- 2016-01-29-Recalibration.docx
- 2016-02-09-Recalibration.docx
- 2016-02-10-Recalibration.docx
- 2016-02-11-Recalibration.docx
- 2016-02-12-Recalibration.docx
- 2016-02-14-Recalibration.docx
- 2016-02-15-Recalibration.docx
- 2016-02-16-Recalibration.docx
- 2016-02-17-Recalibration.docx
- 2016-02-17b-Recalibration.docx
- 2016-02-18-Recalibration.docx
- 2016-02-19-Recalibration-KSK.docx
- 2016-02-19-Recalibration.docx
- 2016-02-23-Recalibration-DR.docx
- 2016-03-01-Recalibration.docx
- 2016-03-01-Recalibrationb.docx
- 2016-03-02-Recalibration.docx
- 2016-03-03-Recalibration-KSK.docx
- 2016-03-03-Recalibration.docx
- 2016-03-04-Recalibration.docx

Why do we do this to ourselves?

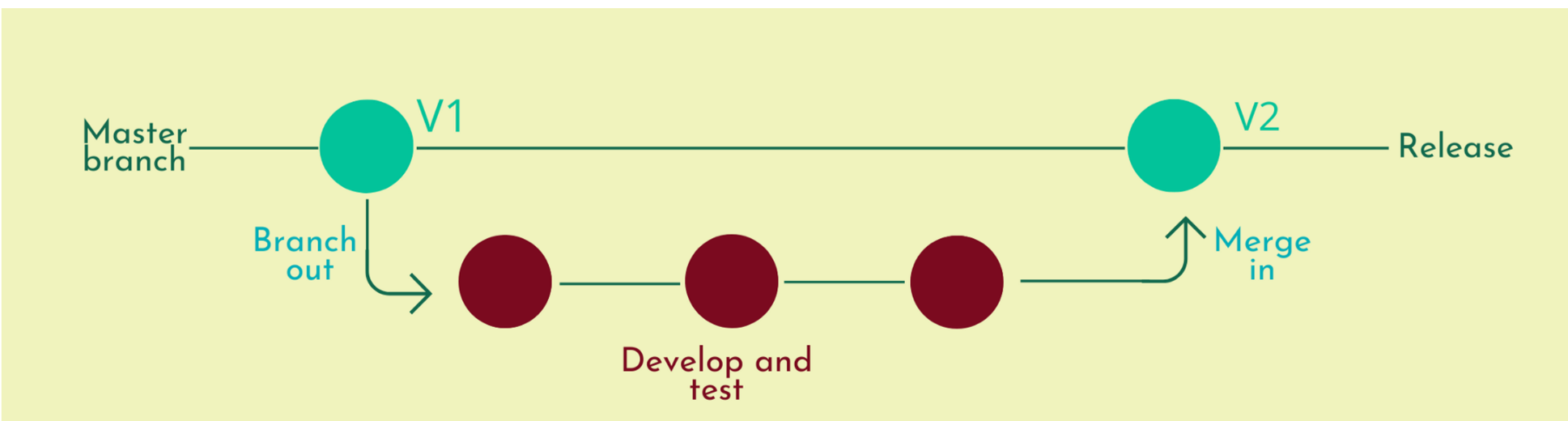
- Want to preserve the document history
  - Easier to take risks, delete things, etc. if you know you can go back (especially for writing code)
  - For collaboration, want to know who made changes
- But it's messy, inefficient, and easy to break
  - Cluttered file folders (imagine if you wanted to track every change to every type of file)
  - Copy -> paste -> rename is tedious
  - No record of *what* the changes were

The solution: Use version control software (e.g., git, svm)

- Define a **repository** (your project folder) and tell git which files to **track** vs **ignore**
- Place your repository on a cloud hub (e.g., GitHub)
- Git tracks what/when changes were made, and you tell git whether to **commit** those changes
- You can **push** those changes to the central hub to store them (now they're saved)
- Other users/computers can **pull** those changes from the hub to keep their local copy up-to-date

More advanced git features

- Create branches (alternate timelines) to develop and test new functions, then merge those changes back to the master branch when they're ready



Other considerations

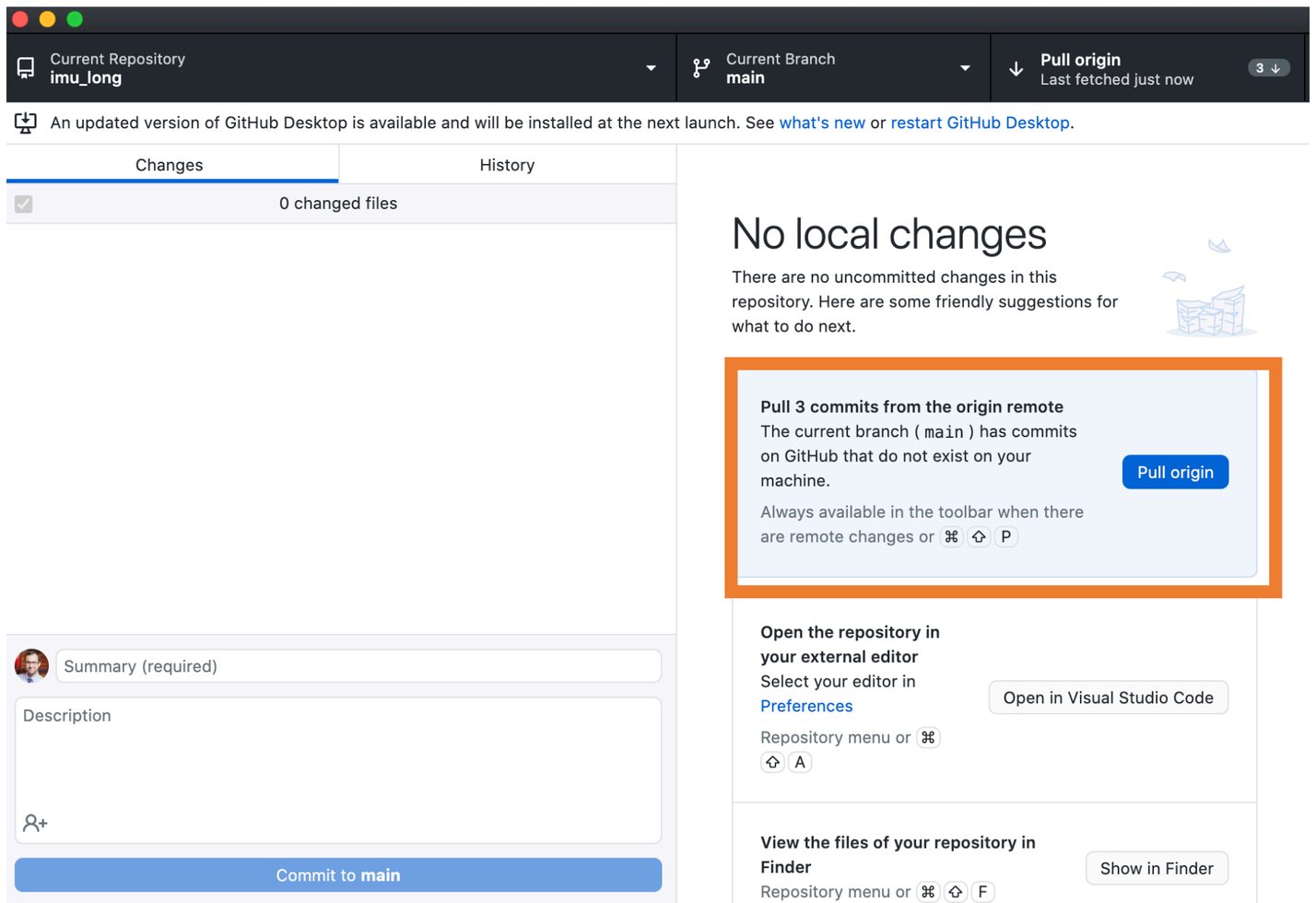
- Easiest to get started working on a solo project
- Collaborative version control can get messy if people forget to commit/push their changes or to pull the most recent version from the master repo
- git wants your folders out of dropbox or other cloud synced drives, so you have to commit to using it
- Lots of ways to use git (Github website, Github app, RStudio git window, command line)
- Public vs. private repos
- Flat files vs. binary files

Essential git skills to master

- Forking repos - creating a copy of a repo that points to the original [DONE]
- Cloning a repo - pulling a version from the cloud onto your local system [DONE]
- Pulling changes from the cloud
- Committing and pushing changes to the cloud
- Discarding unwanted changes
- Creating a new repo from an existing local directory

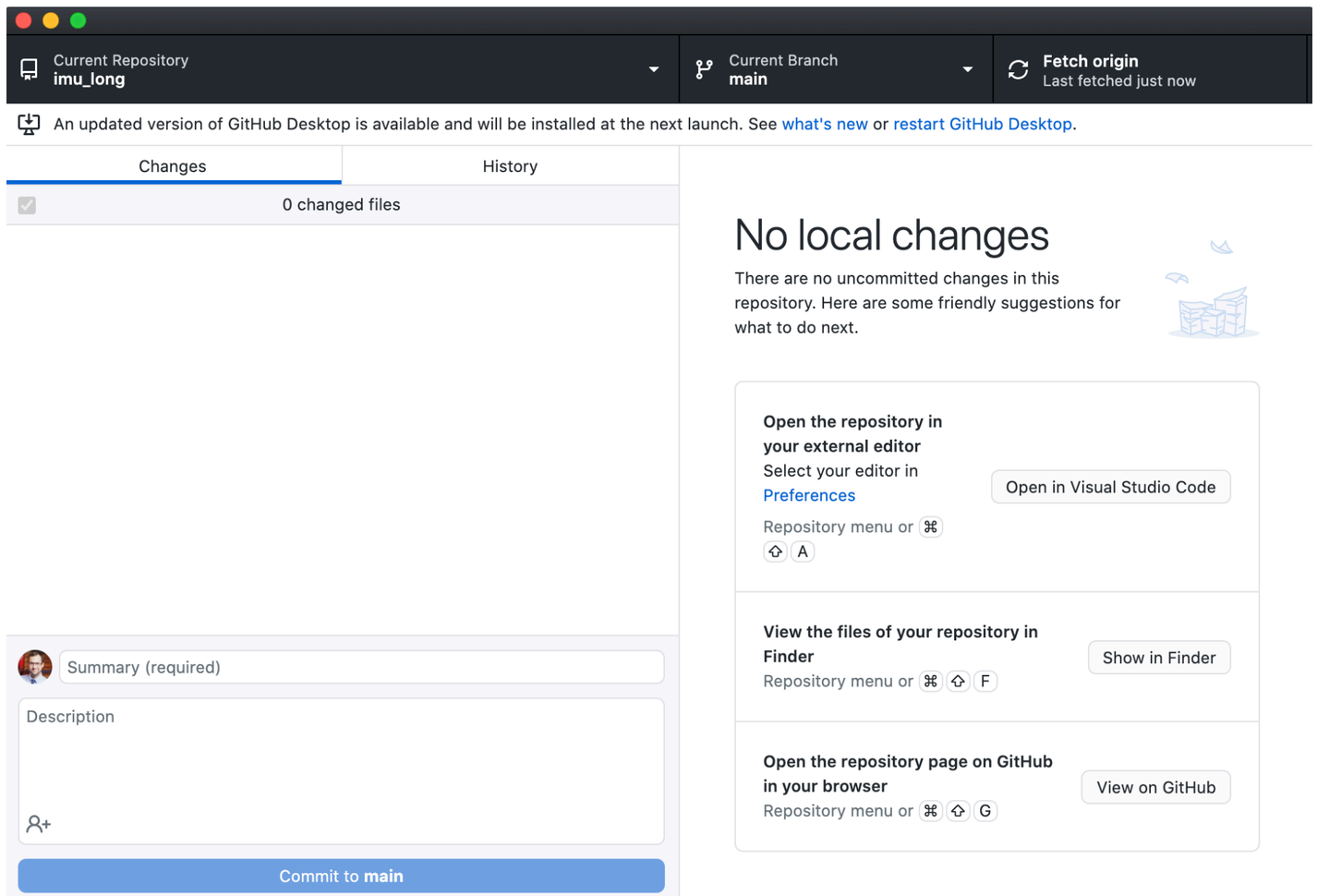
Pulling changes

- Before working on a local project, always check for cloud changes to pull



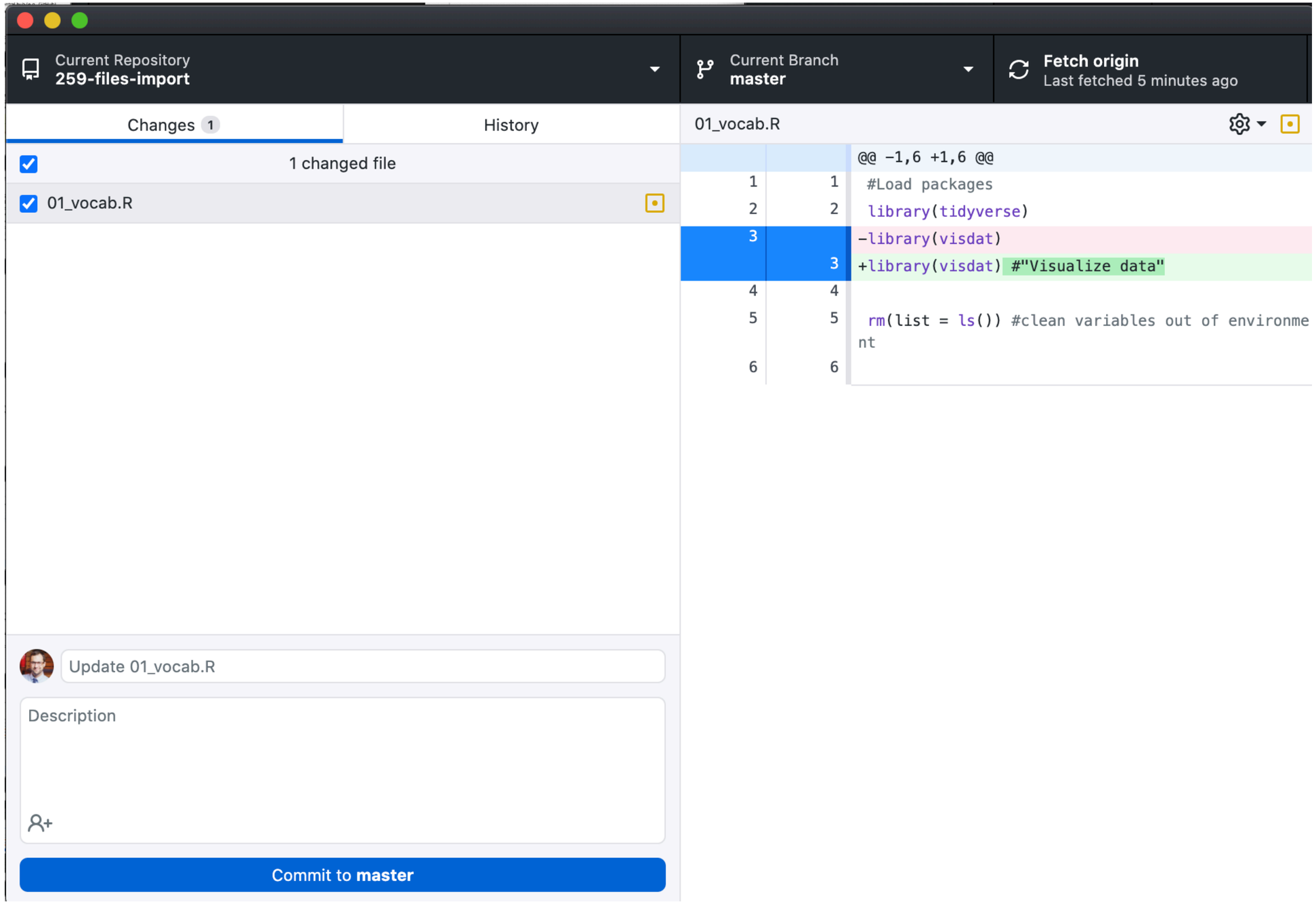
Pulling changes

- After pulling from the “origin”, you won’t be prompted to pull



Making changes to a file

- Open the “install_packages.R” file, make any change to the text, and save
- Github will automatically track any change made to a file in a repo


Current Repository
259-files-import
Current Branch
master
Fetch origin
Last fetched 5 minutes ago
Changes 1
History
01_vocab.R
1 changed file
01_vocab.R
@@ -1,6 +1,6 @@
#Load packages
library(tidyverse)
-library(visdat)
+library(visdat) #"Visualize data"
rm(list = ls()) #clean variables out of environment
Update 01_vocab.R
Description
Commit to master

Commit those changes

- If you want to keep changes (to one or more files), create a commit
  - Write a short message about the changes, and click commit
  - Your changes will disappear ("no local changes") because your local copy is updated
- Next, we **push** those changes to the origin on GitHub

Current Repository
259-files-import

Current Branch
master

Push origin
Last fetched 8 minutes a...
3 ↑

Changes | History

0 changed files

Summary (required)

Description

Commit to **master**

Committed just now
Revert "Revert "Added a helpful comment""

Undo

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote
You have 3 local commits waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or ⌘ P

Push origin

Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⇧ A

Open in Brackets

View the files of your repository in Finder
Repository menu or ⌘ ⇧ F

Show in Finder

Open the repository page on GitHub in your browser

Current Repository
259-files-import

Current Branch
master

Fetch origin
Last fetched just now

Changes | History

0 changed files

Summary (required)

Description

Commit to **master**

No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Preferences
Repository menu or ⌘ ⇧ A

Open in Brackets

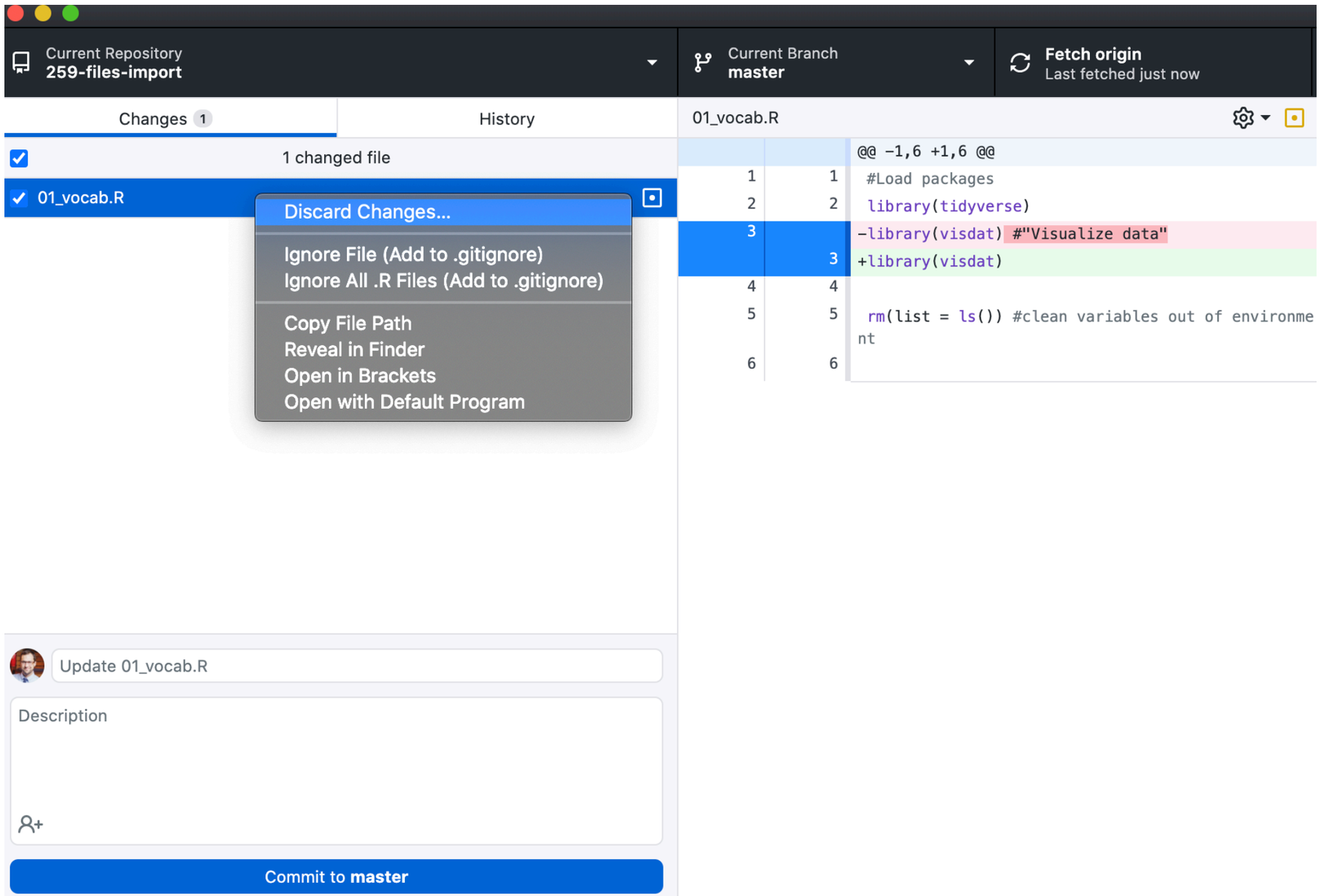
View the files of your repository in Finder
Repository menu or ⌘ ⇧ F

Show in Finder

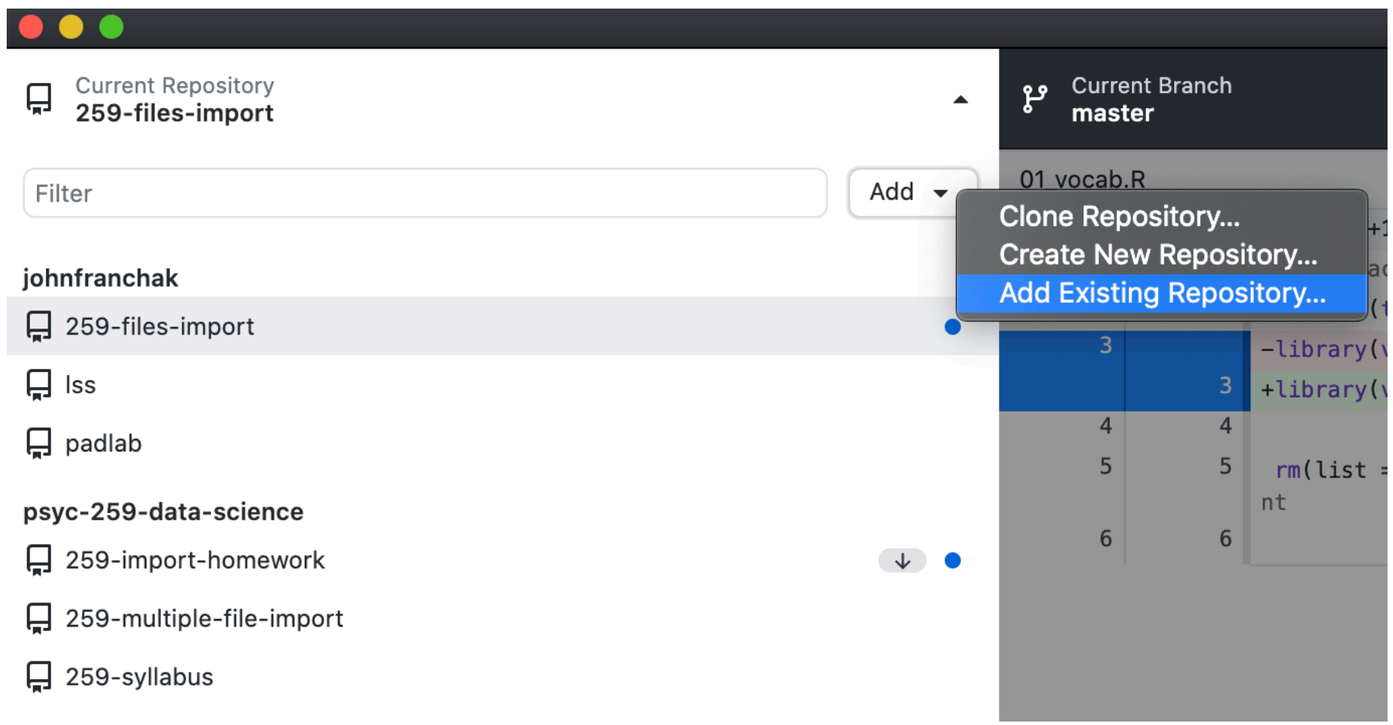
Open the repository page on GitHub in your browser
Repository menu or ⌘ ⇧ G

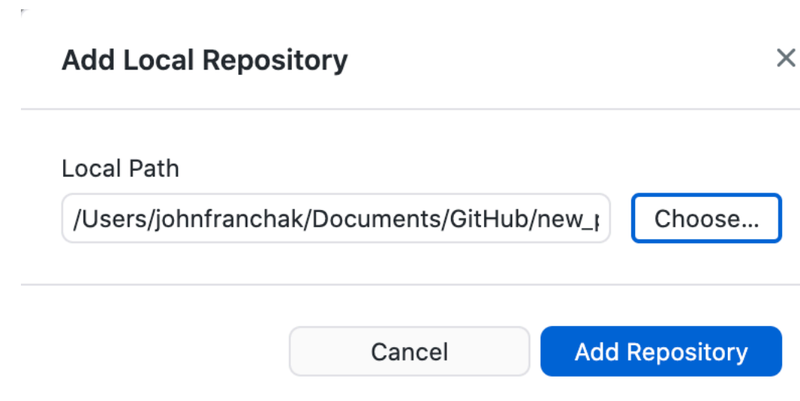
View on GitHub

Don’t want those changes? **Discard** them!



Add existing (local) repository to the Github app

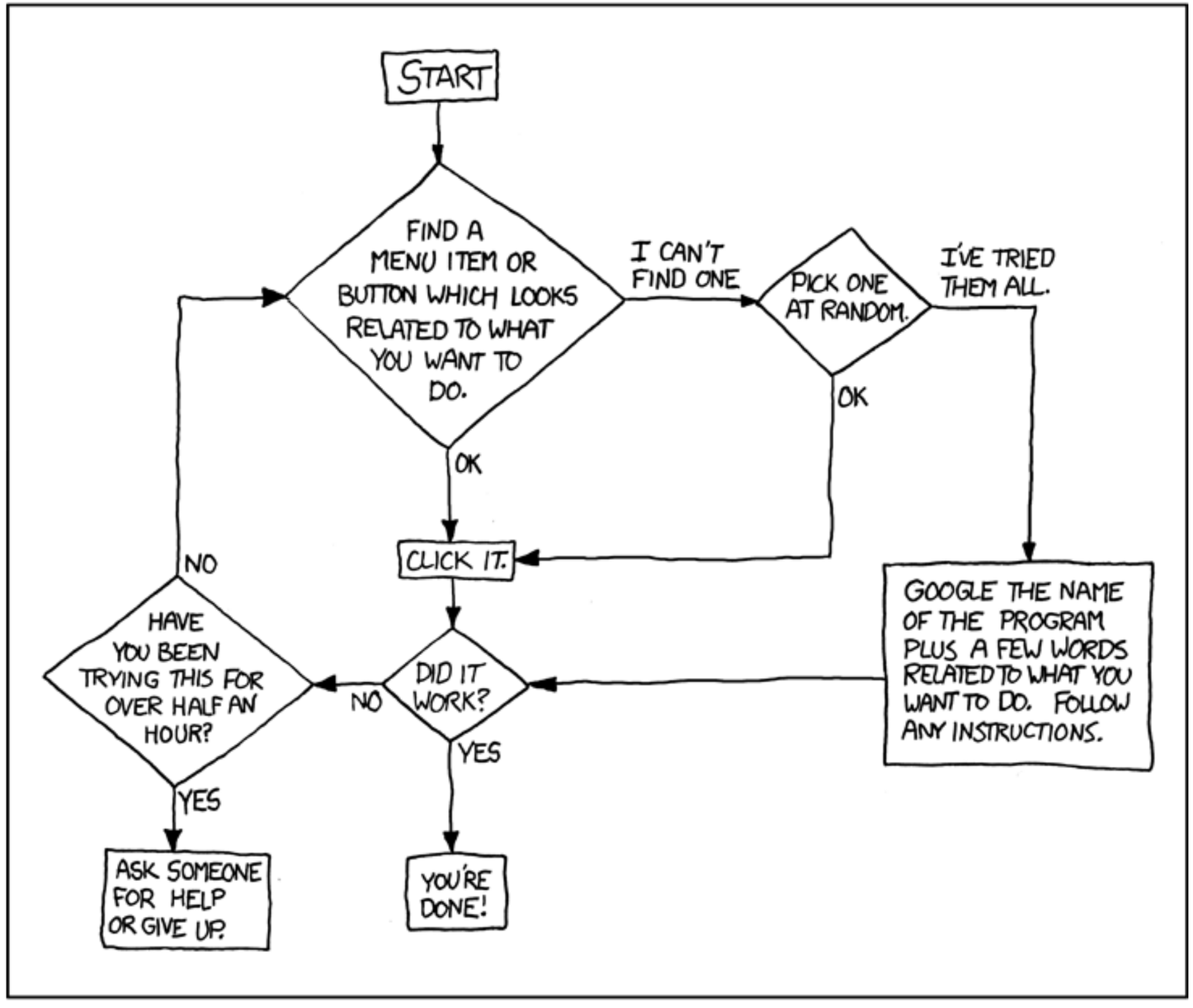




General advice:
How to get help

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS, AND OTHER "NOT COMPUTER PEOPLE."

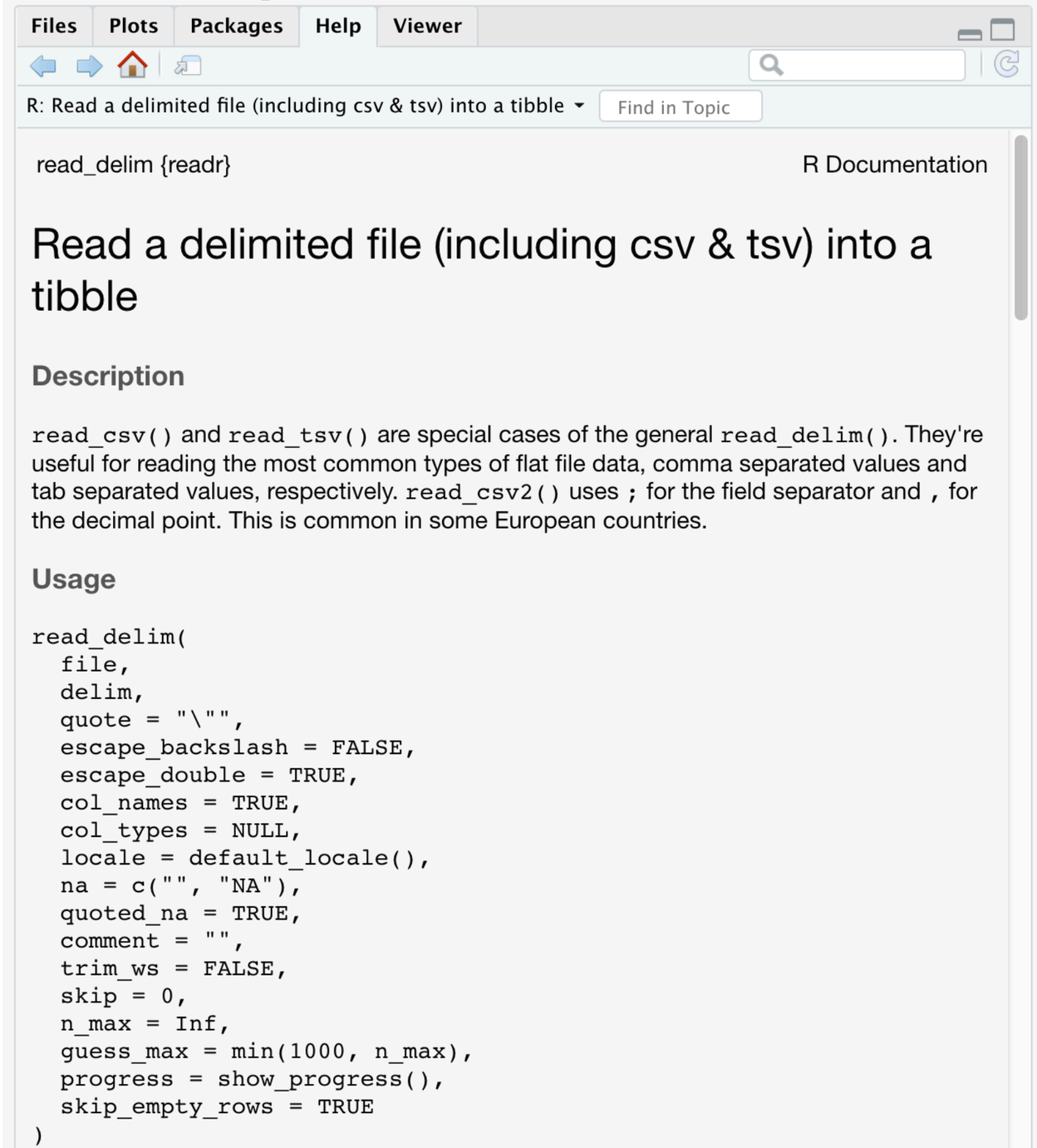
WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

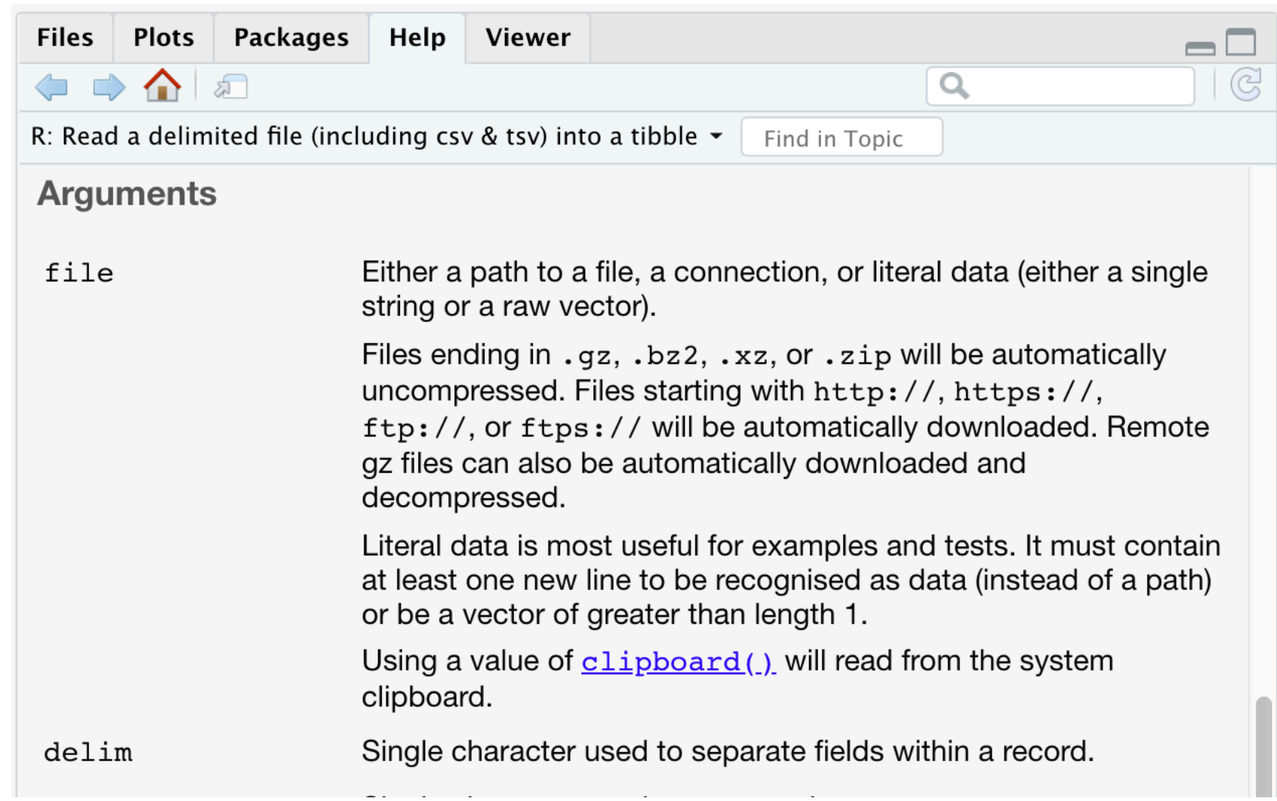
Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)



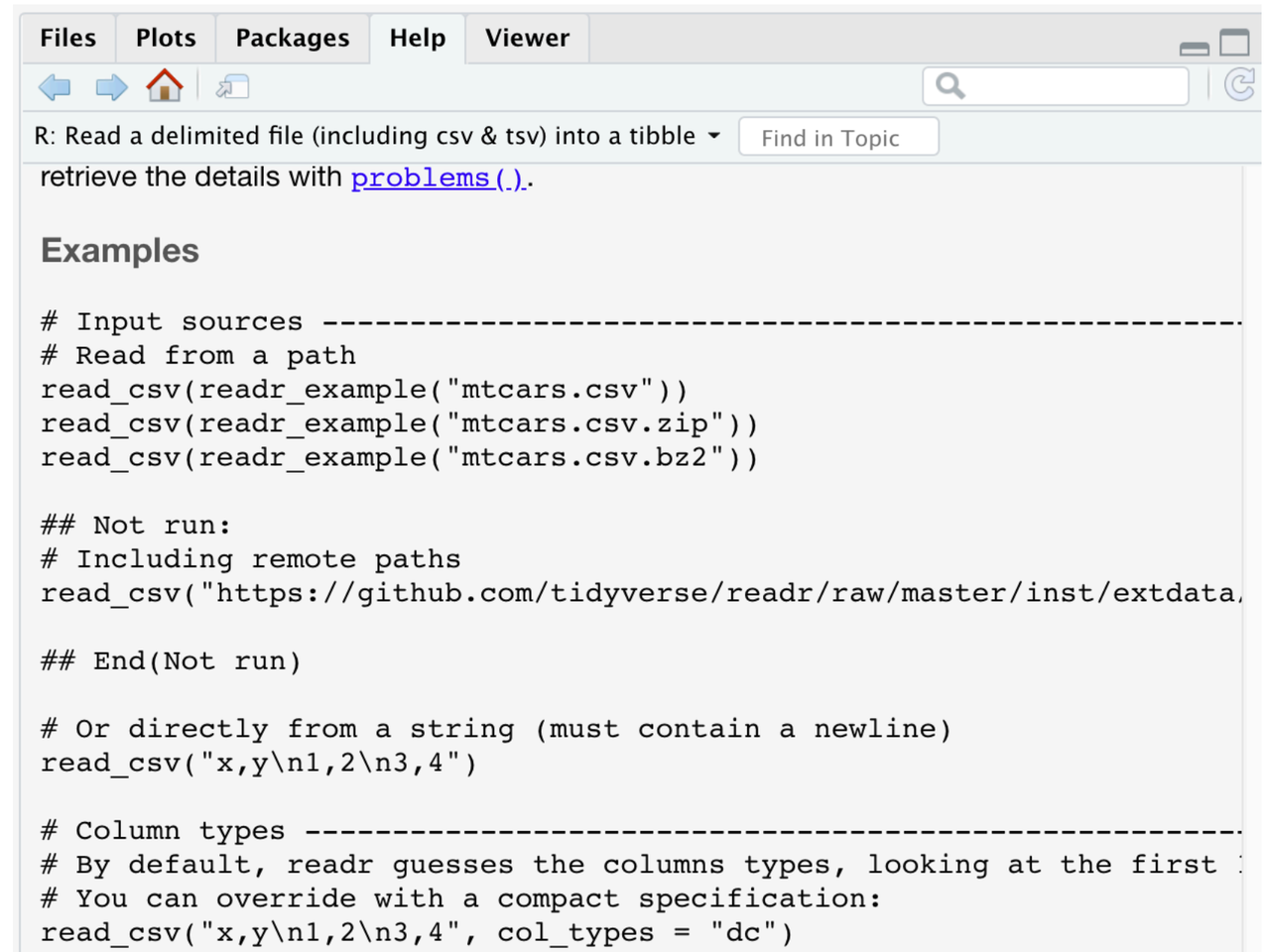
Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)
  - Scroll down to see the "arguments" definitions



Where to search for help

- R Documentation
  - ?function brings up the documentation for a function (assuming the package is loaded)
  - Scroll down to see the "arguments" definitions
  - Scroll down more to see examples



Where to search for help

- Package vignettes/blogs
  - https://cran.r-project.org/web/packages/readr/vignettes/readr.html
- Google
  - Blogs/instructional sites
  - StackExchange (especially for error text)
  - Tidyverse community
    - https://community.rstudio.com/c/tidyverse/6
- Take working examples and fiddle!