Dataset Builder Guide

Clayton Young

2023-07-26

Table of Contents

Overview

The goal of this document is to enable users of these tools to build BraNCH datasets independently using a repeatable, replicable process.

Topics Covered:

- Orientation to the process
- Installing the necessary tools required for generating these datasets
- File organization
- Generating timepoints
- Wrangling, cleaning, and scoring instruments
- Building off previous versions of data freezes
- Building and generating datasets
- Adding new instruments

Breakdown of instrument roles after training:

Task	**Clayton/tech**	**You**	**Users**
Update existing instrument	Y	Y	N
Add new instrument	Y	M	N
Add new list of PIDNs	Y	Y	N
Select and download data	Y	Y	Y

Notes

- This guide is designed to complement the Quarto files referenced in the builder. It will help you set up and utilize the Quarto files effectively, while also covering the various features of the data set builder tools. For detailed explanations of the code, refer to the documentation within the Quarto files. For troubleshooting, please consult this guide.
- Code is in the GitHub repository
- Data is in the R Drive and Stored locally

Training Agenda

Session 1: Setup and Orientation

1. Overview of the BrANCH dataset build process
2. Setup and installation of necessary tools: R, R Studio, GitHub, xcode command line tools, gfortran, cmake, and package dependency check using renv

Session 2: File System Organization and LAVA Query Data

1. Understand and set up the file structure
2. Download LAVA Query data
3. Run timepoint curation script

Session 3: Data Wrangling, Cleaning, and Scoring

1. Download additional LAVA Data
2. Run data wrangling, cleaning, and scoring script
3. Download data freeze from R Drive
4. Compare the output of the refresh script with data freeze

Session 4: Dataset Generation

1. Run dataset generation script
2. Run data downloader script
3. Download csv and excel versions of the dataset(s)

Session 5: Full Run

1. Import data required for full dataset build
2. Generate and download full dataset(s)

Session 6: Adding Instruments

1. Add a new instrument(s) to the data freeze
2. Other common scenarios and troubleshooting

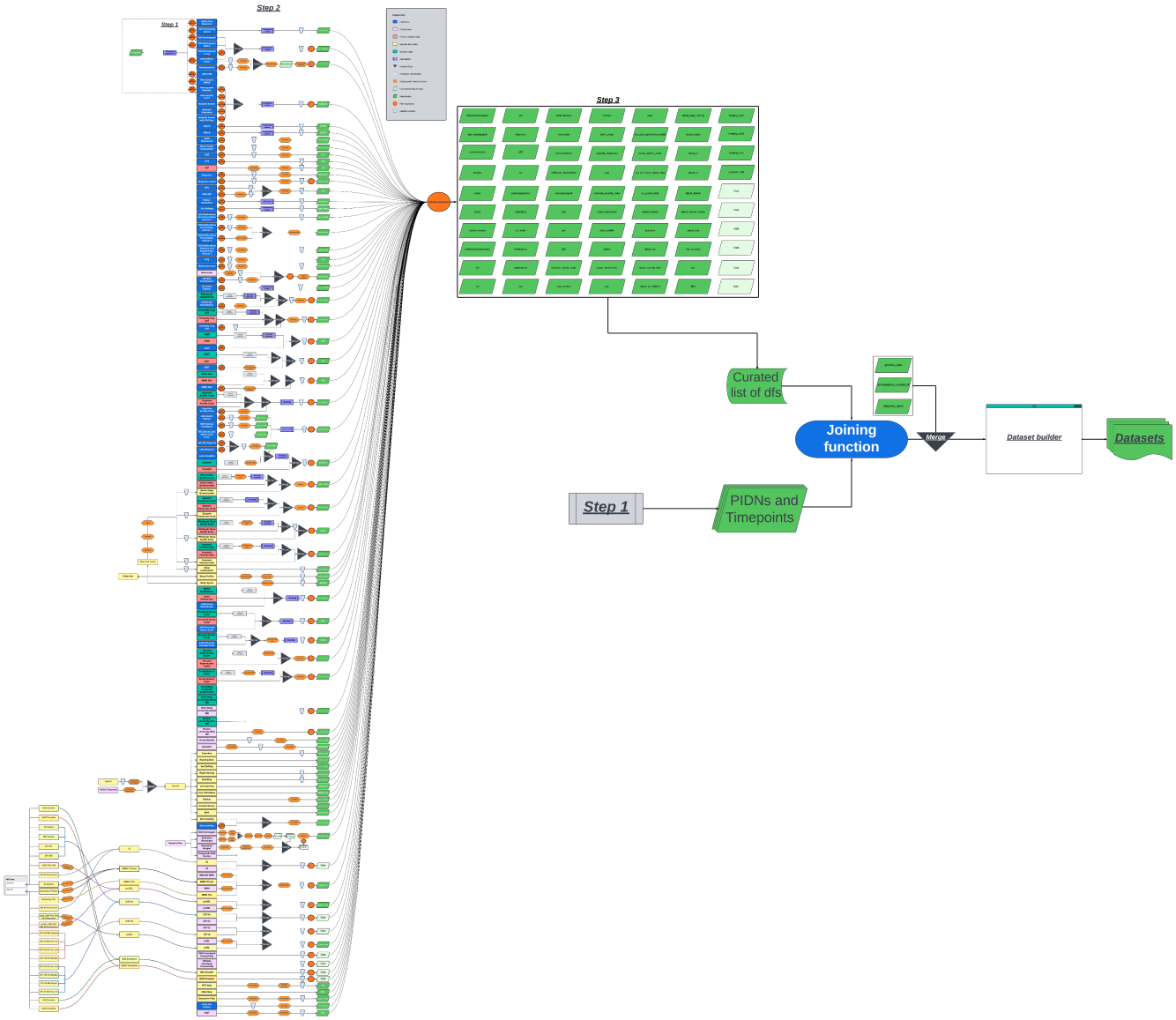
Orientation

By the end of this session, you should:

1. Be familiarized with the dataset build process

Repeatable Process

- Many moving parts in this process that required archaeology
- Our goal is to make this general process repeatable:

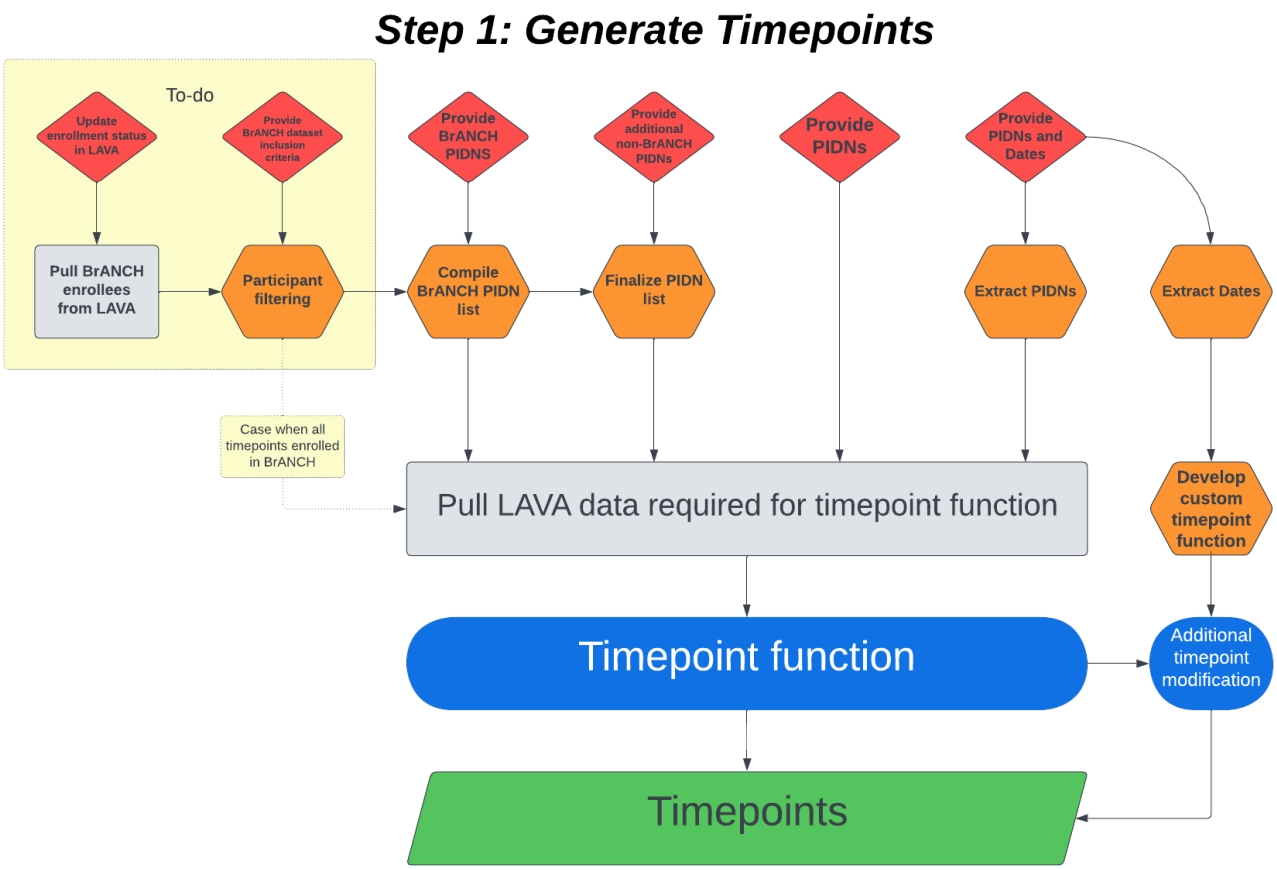


Validating format of incoming data

- Part of making this process repeatable is ensuring incoming data matches data currently in use
- This is done by storing templates of the current files and comparing imported data to templates
- Detecting differences early on gives us confidence in our data outputs

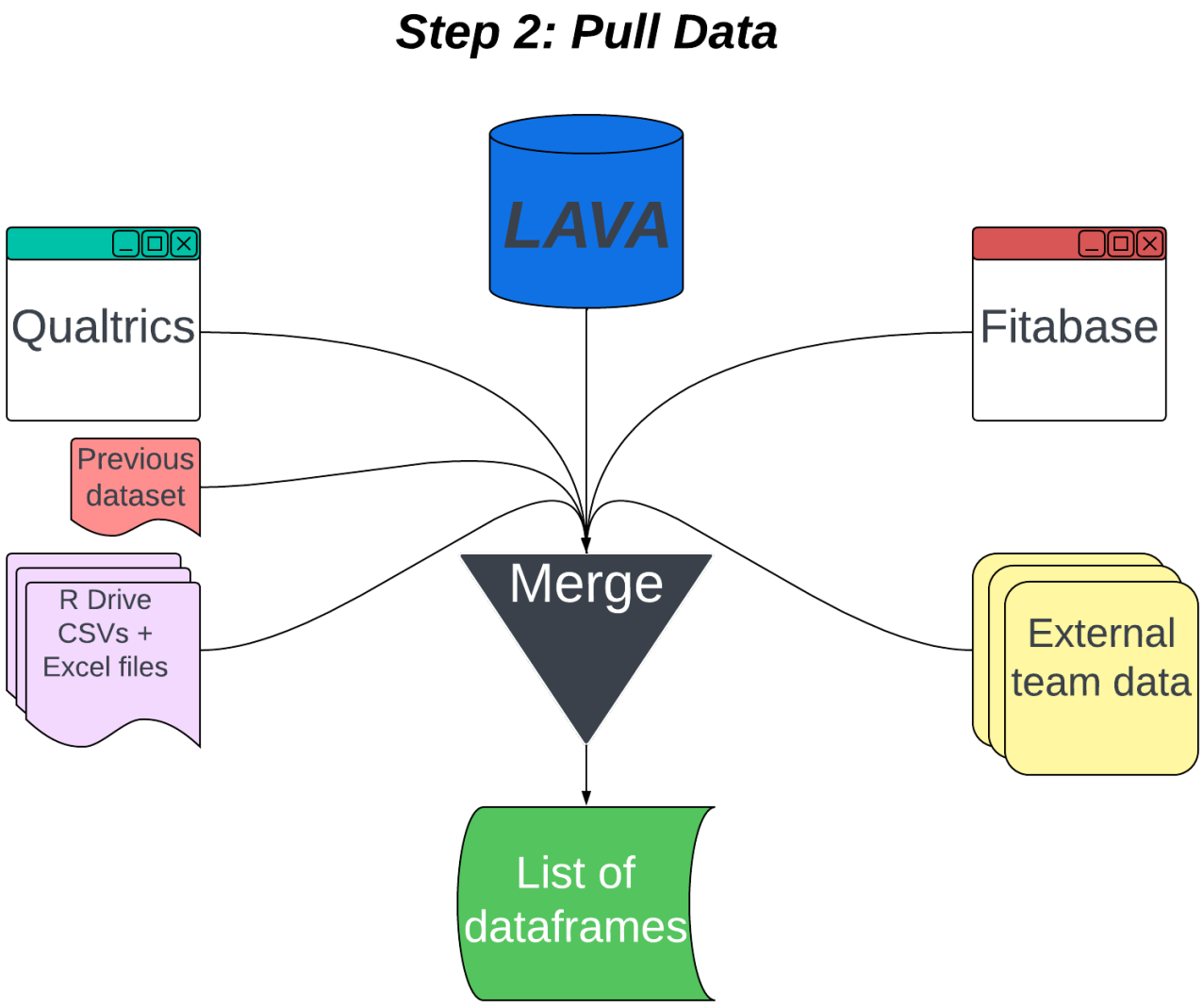
BrANCH Timepoint Generation

- The first step in the process involves generating timepoints using the criteria provided by BraNCH
- Generating timepoints requires importing data from LAVA and a set of PIDNS
- If dates are already present with PIDNS, we can skip this process if desired



Cleaning and Scoring

- Following generating the timepoints, cleaning and scoring procedures transform the data
- Format checks are in place here to ensure data is in correct structure before transformations
- This data is stored and used to build the datasets

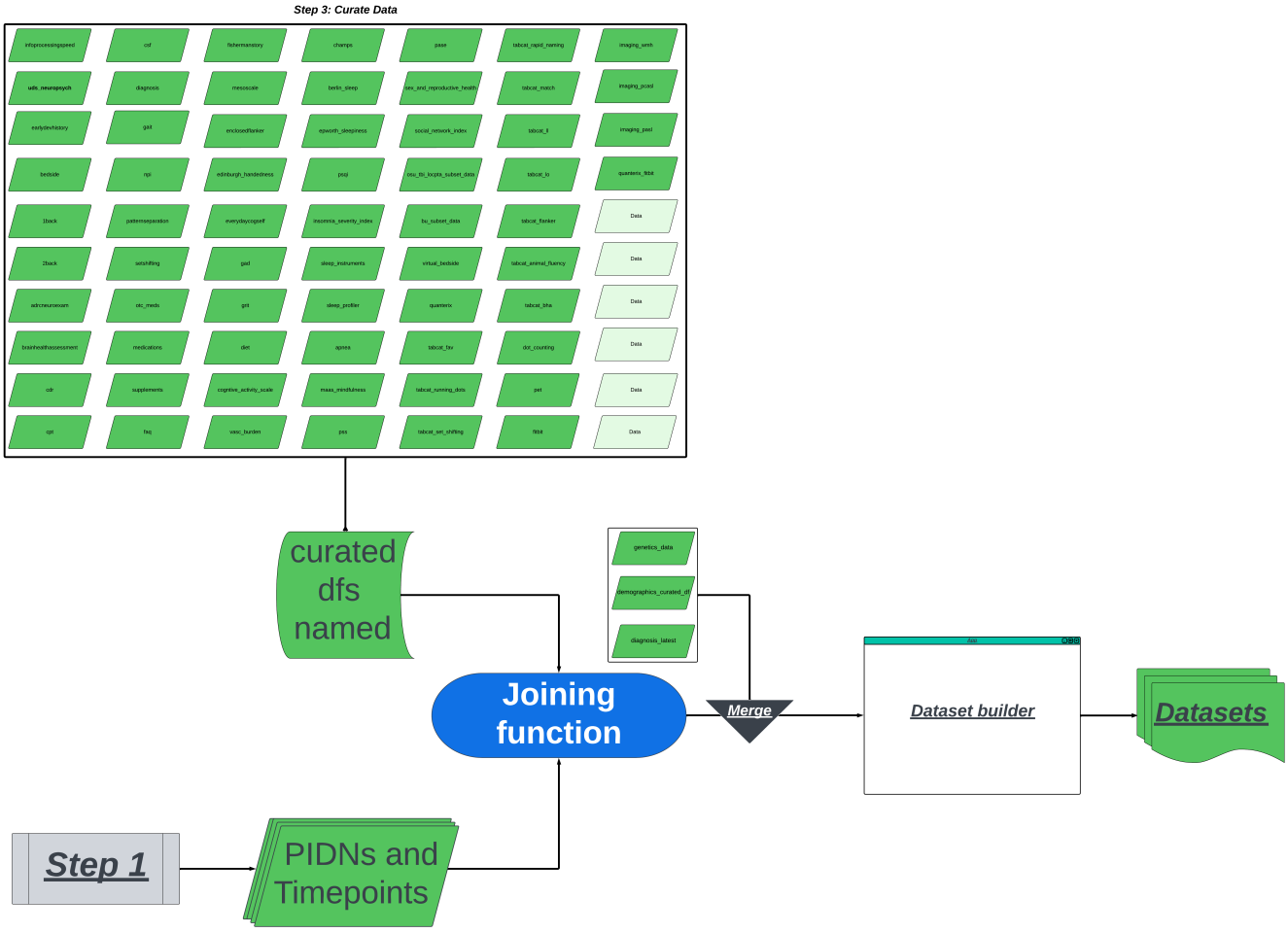


Storing Data and Detecting Differences

- If building off current build of dataset, discrepancies are detected
- Options to overwrite existing records or keep previous data
- If not building off previous data, only new data is stored as a pin, which is a locally versioned storage of data

Generating Datasets and Selecting Variables

- PIDNs and Timepoints are joined to the transformed data in this step
- Datasets are also stored in as pin
- These pins are read in by dataset builder to select variables of interest to download



Setup

By the end of this session, you should:

1. Have your environment set up to run the dataset builder

2. Have the file structure stored on your computer

3. Have the required packages installed

4. Downloaded the current data freeze

Installations

macOS (11+) :

Recommendation: If you have R and R studio installed, the additional tools we'll need might not work if you don't start fresh. I'd recommend opening RStudio and running `.libPaths()` in the R Studio console. Delete the contents in those folders, delete R and RStudio, and empty your trash!

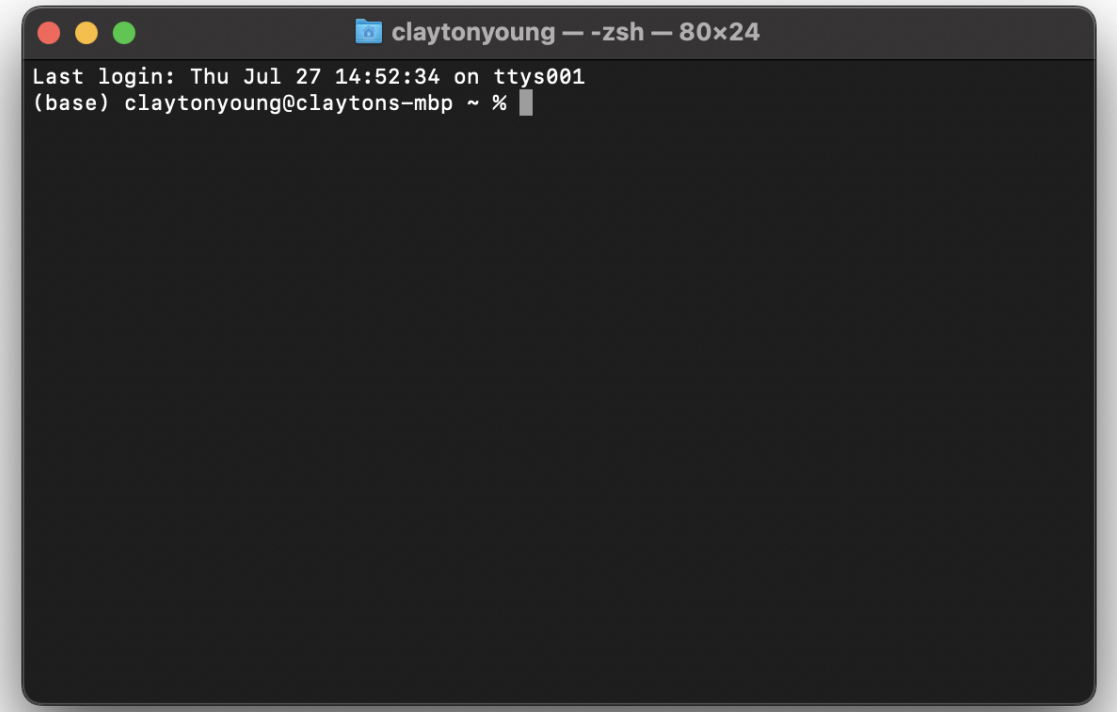
To start fresh, run `.libPaths()` in your RStudio console, which will display where your R data (not R scripts) is currently stored. The path should contain a folder called `R.Frameworks'. Delete this folder (will also delete all contents contained), delete R and RStudio Apps by going to your applications folder and sending them to the trash. Empty your trash bin after everything is deleted. Restart your computer and continue with homebrew installation.

Apple doesn't provide all the necessary tools needed, so we'll need to install some additional tools:

- Homebrew
    - R and Rstudio

Homebrew

- This tool install the missing dependencies and ensures they're working together properly
    - We'll use this to install xcode command line tools, R, and R studio
- Find and open the terminal (search applications for 'terminal'):


claytonyoung — -zsh — 80×24
Last login: Thu Jul 27 14:52:34 on ttys001
(base) claytonyoung@claytons-mbp ~ %

- Install homebrew by pasting this in a new terminal window:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- Enter your user password (used to log in to computer)
- Accept xcode command line tools prompt by clicking RETURN
- Accept created directories
- Follow instructions upon completion
  - For example, paste the output **similar** to this in **your** terminal and click RETURN:

```
echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.zprofile
eval "$(/opt/homebrew/bin/brew shellenv)"
```

R and RStudio (using Homebrew):

- Once Homebrew is done installing, install R
- Open the terminal and paste:

```
brew install r
```

- Press RETURN
- This will take a while
- When R is done installing, install R Studio by pasting this into the terminal

```
brew install --cask rstudio
```

- Press RETURN
- When R studio is done downloading, you'll see the R Studio app in your applications

Set up R Studio

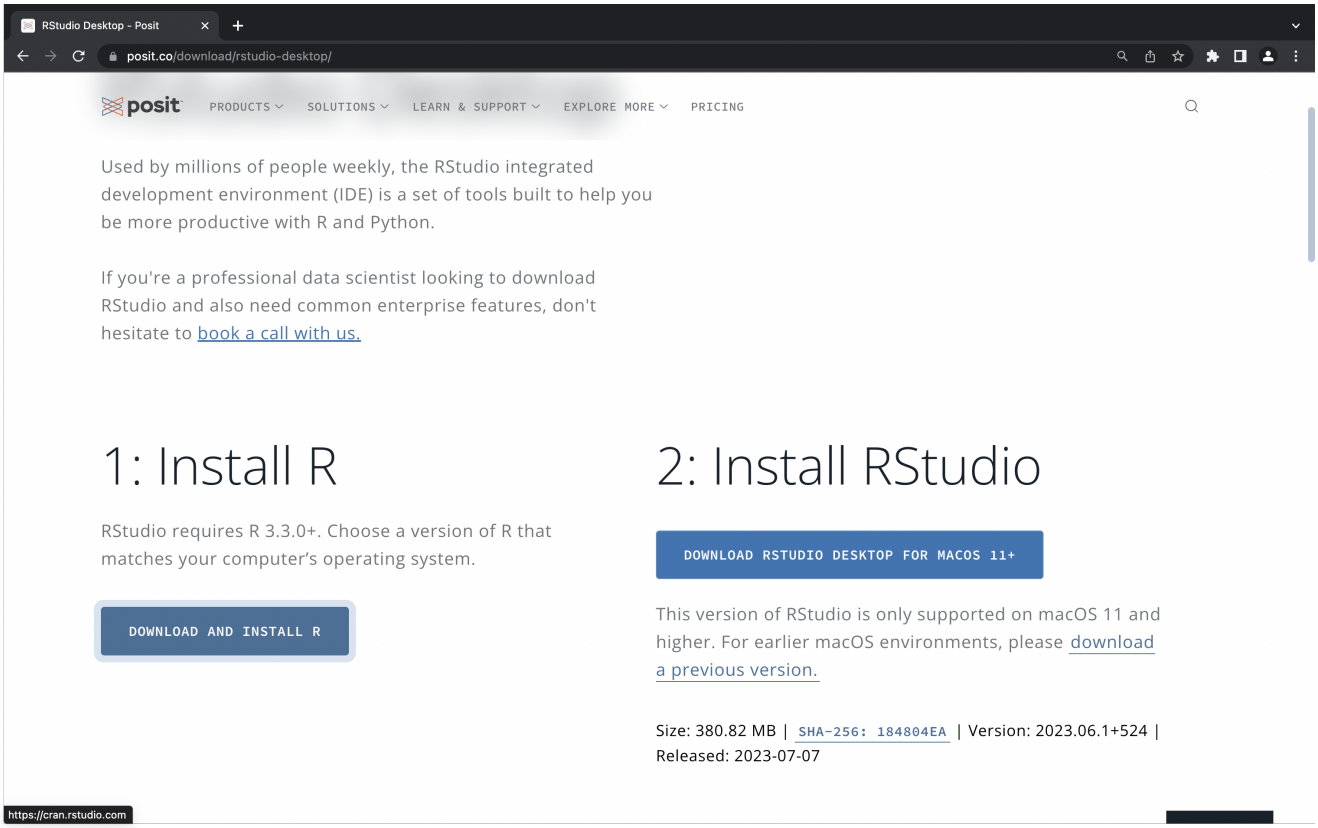
- Outside of an R Project in R studio (just open RStudio app)
  - run `install.packages("renv")`
  - open the project file
  - run renv::restore()
  -

Windows installations (needs verification):

- R and R Studio
- Rtools
- GitHub Desktop

Installing R and R Studio

- Download R and R Studio
- First Download R for the specific version for your system



CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2023-06-16, Beagle Scouts) R-4.3.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize network load.

Submitting to CRAN

Submissions are closed: This year's CRAN submission summer break (CRAN team vacation and maintenance work) is from Jul 21, 2023 to Aug 7, 2023.

- Download **and** install base R and RTools

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

R for Windows

Subdirectories:

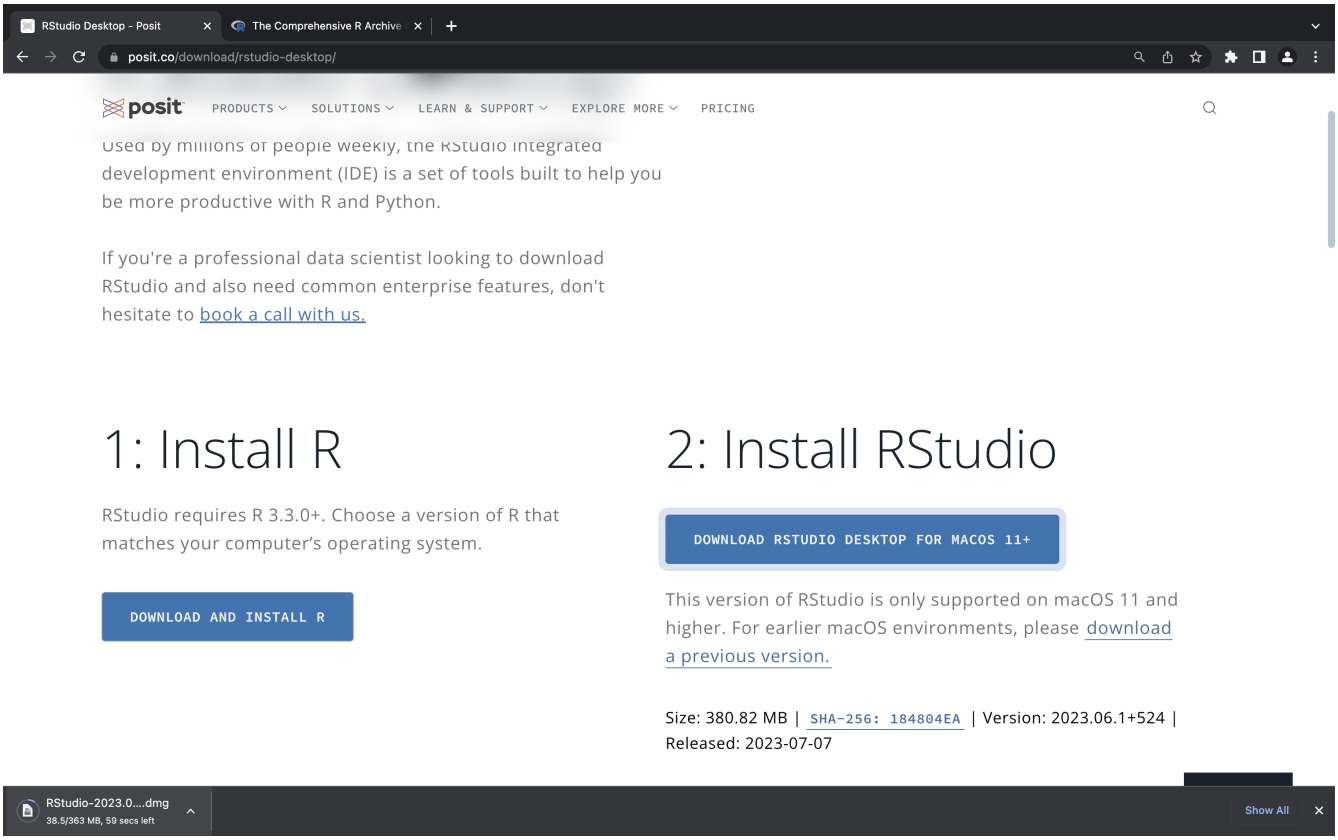
base	Binaries for base distribution. This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 3.4.x).
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 3.4.x).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

- Go back to the posit homepage to download RStudio Desktop once R is installed



- You should now have both R and R Studio installed

Install GitHub and Clone the Repository

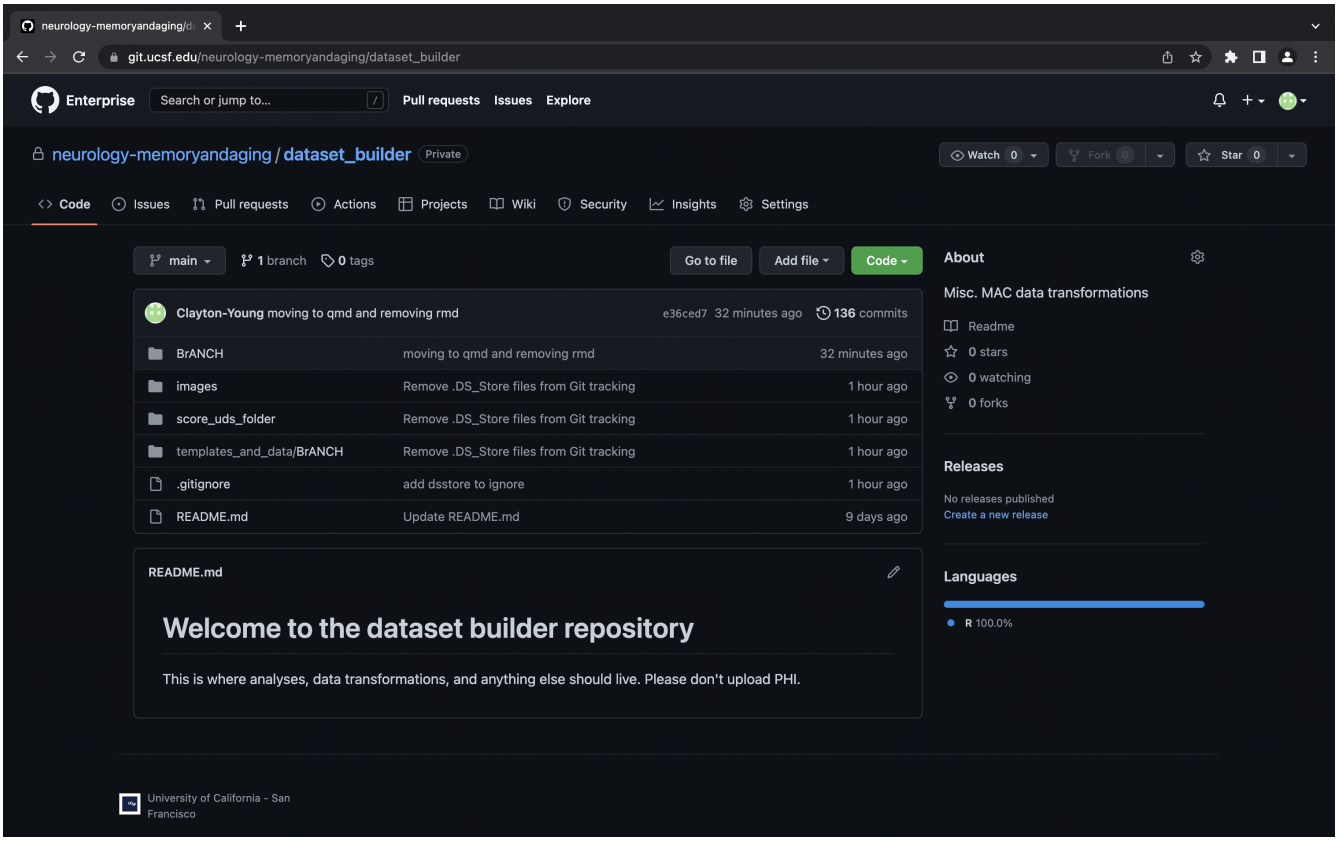
- Follow this link and log in to MyAccess

  GitHub

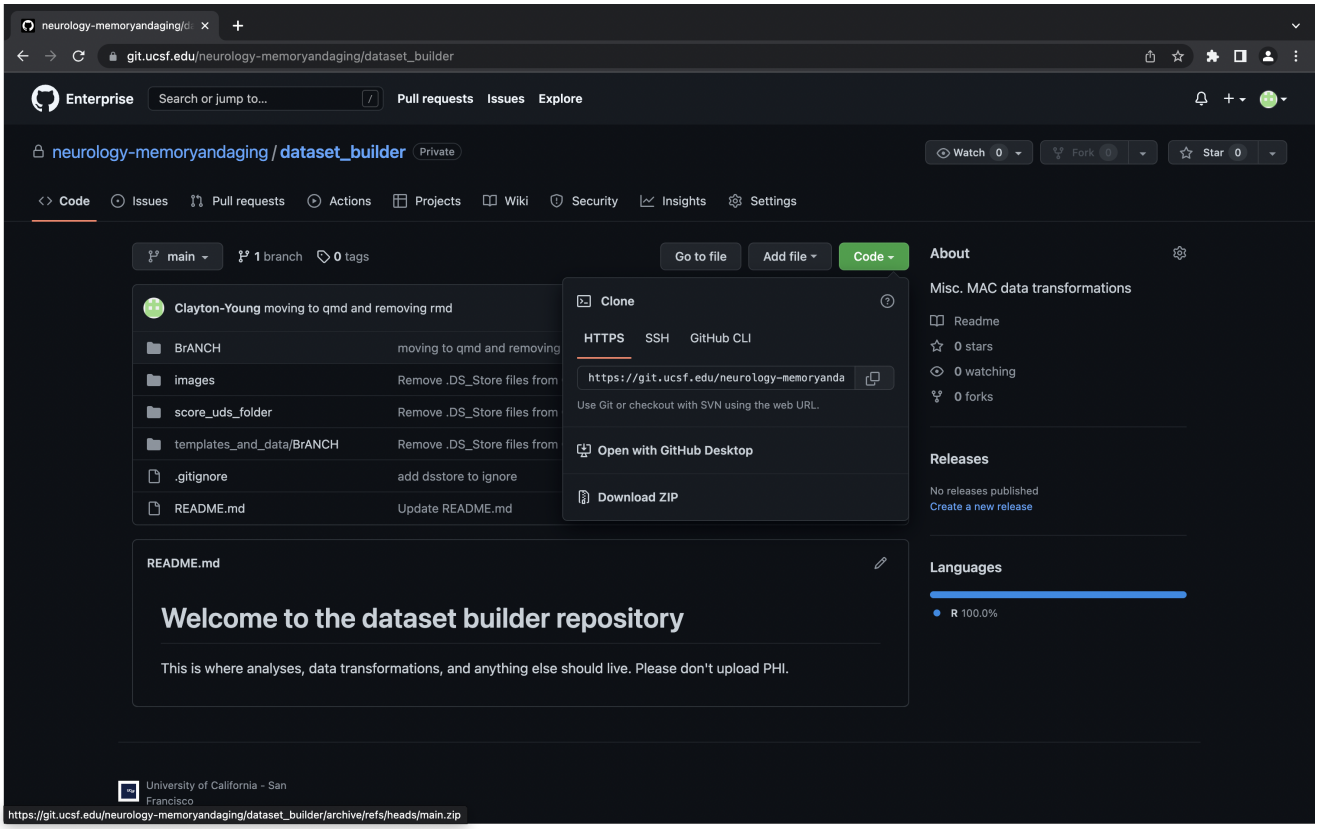
- Once logged in, go to the repository located here:

  https://git.ucsf.edu/neurology-memoryandaging/dataset_builder

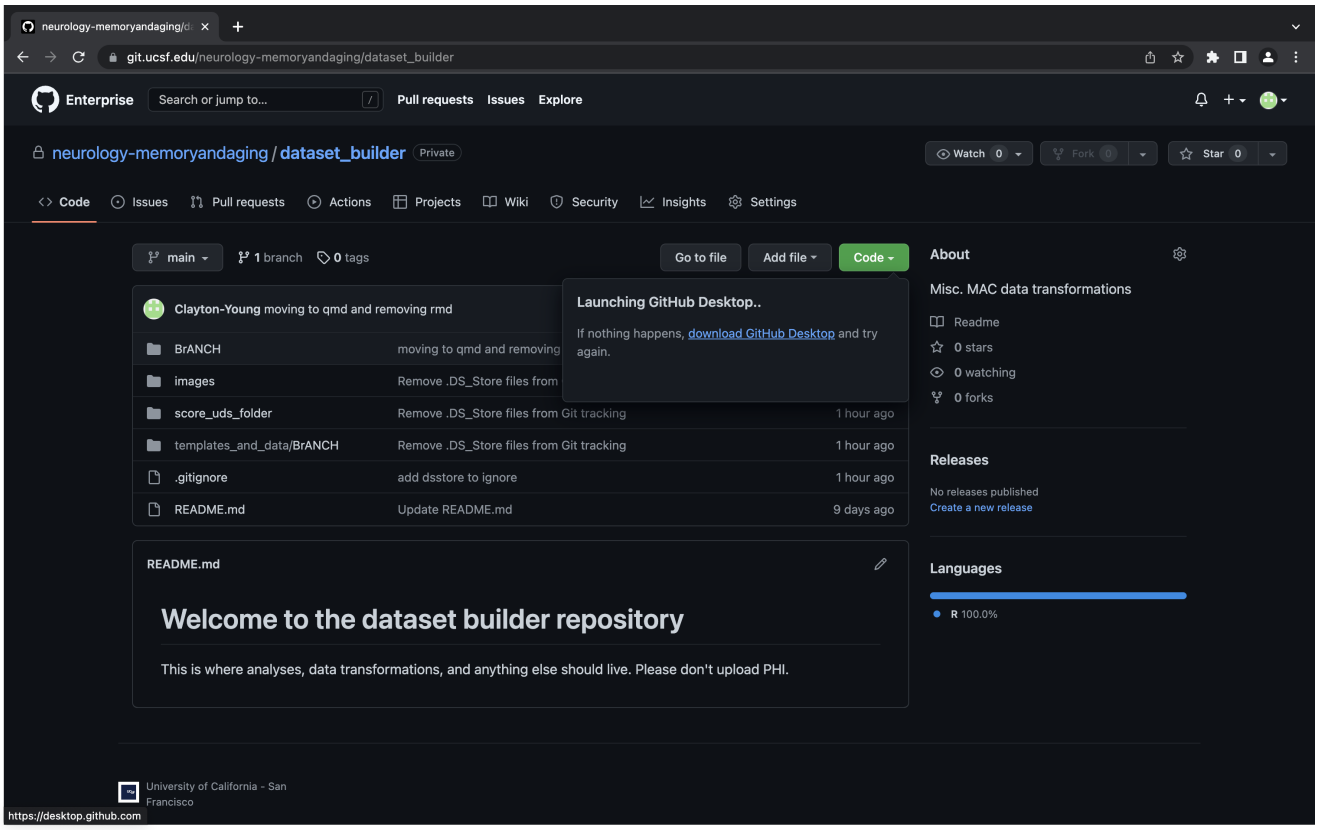
- Click the green ‘code’ button



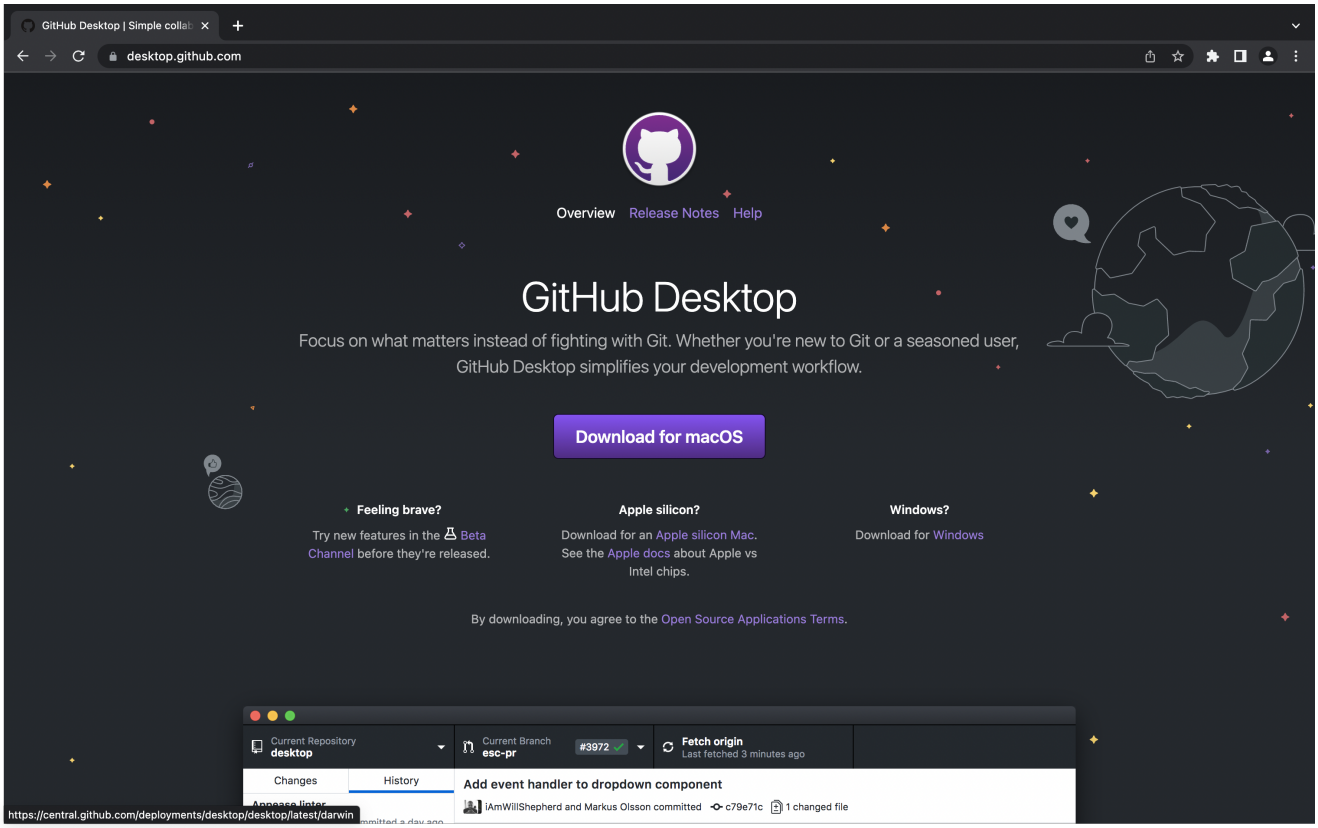
- Click ‘Open with GitHub Desktop’



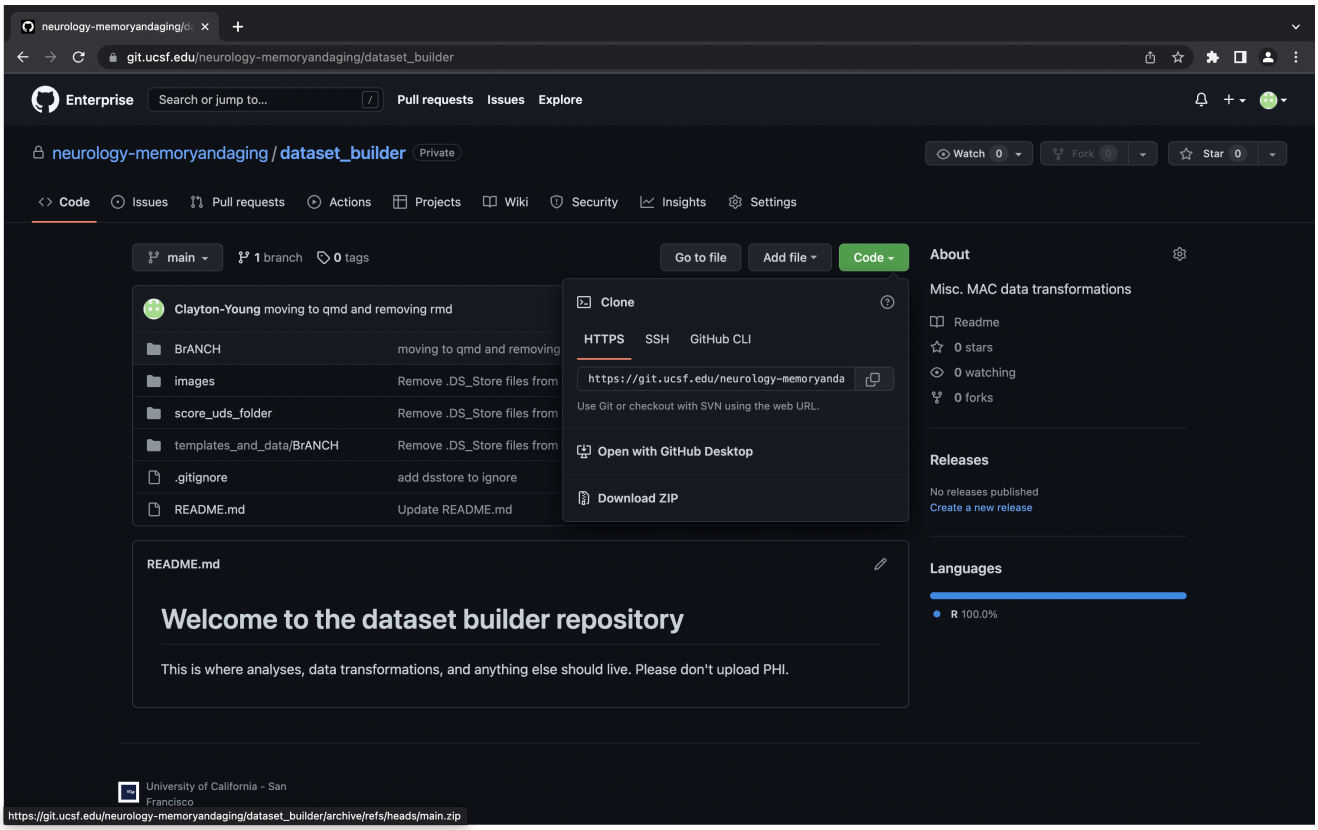
- Download the GitHub Desktop App by following the link ‘Download GitHub Desktop’



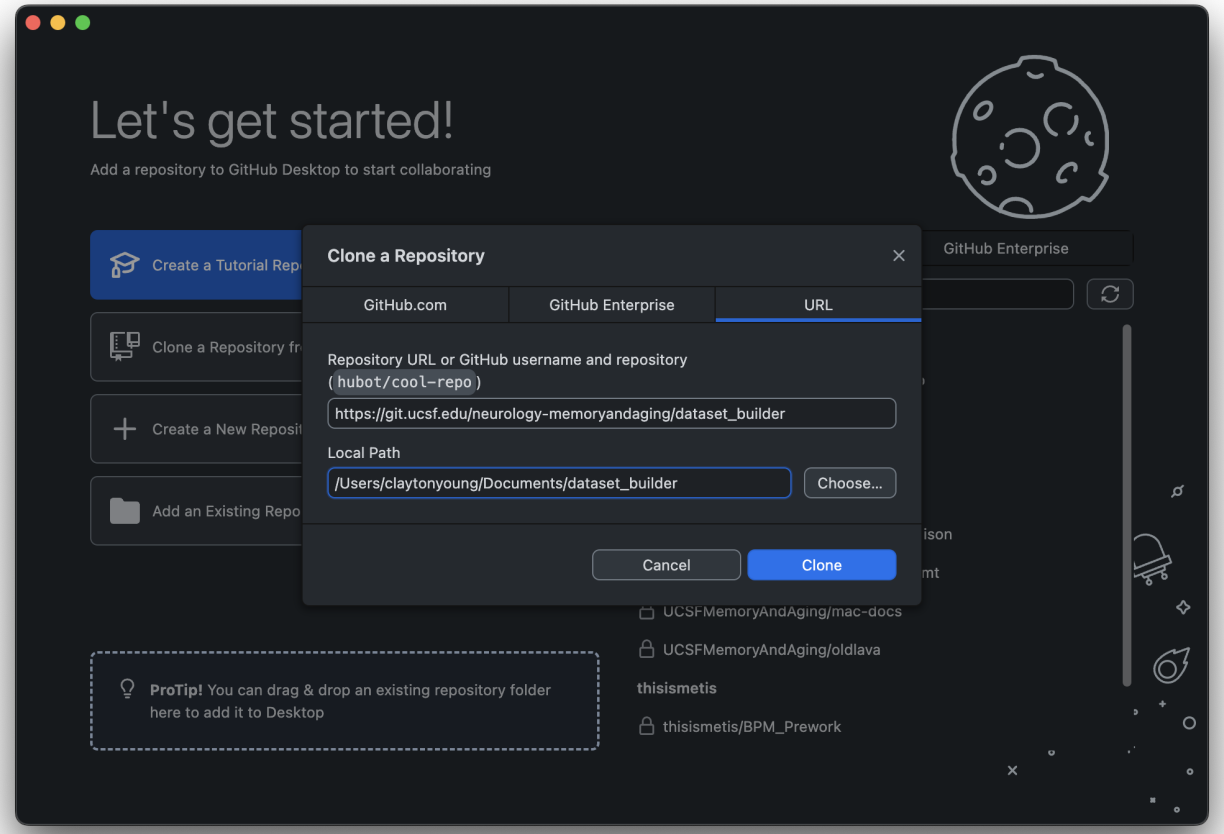
- Install for your system



- Once Downloaded, you may need to repeat this step: Click ‘Open with GitHub Desktop’

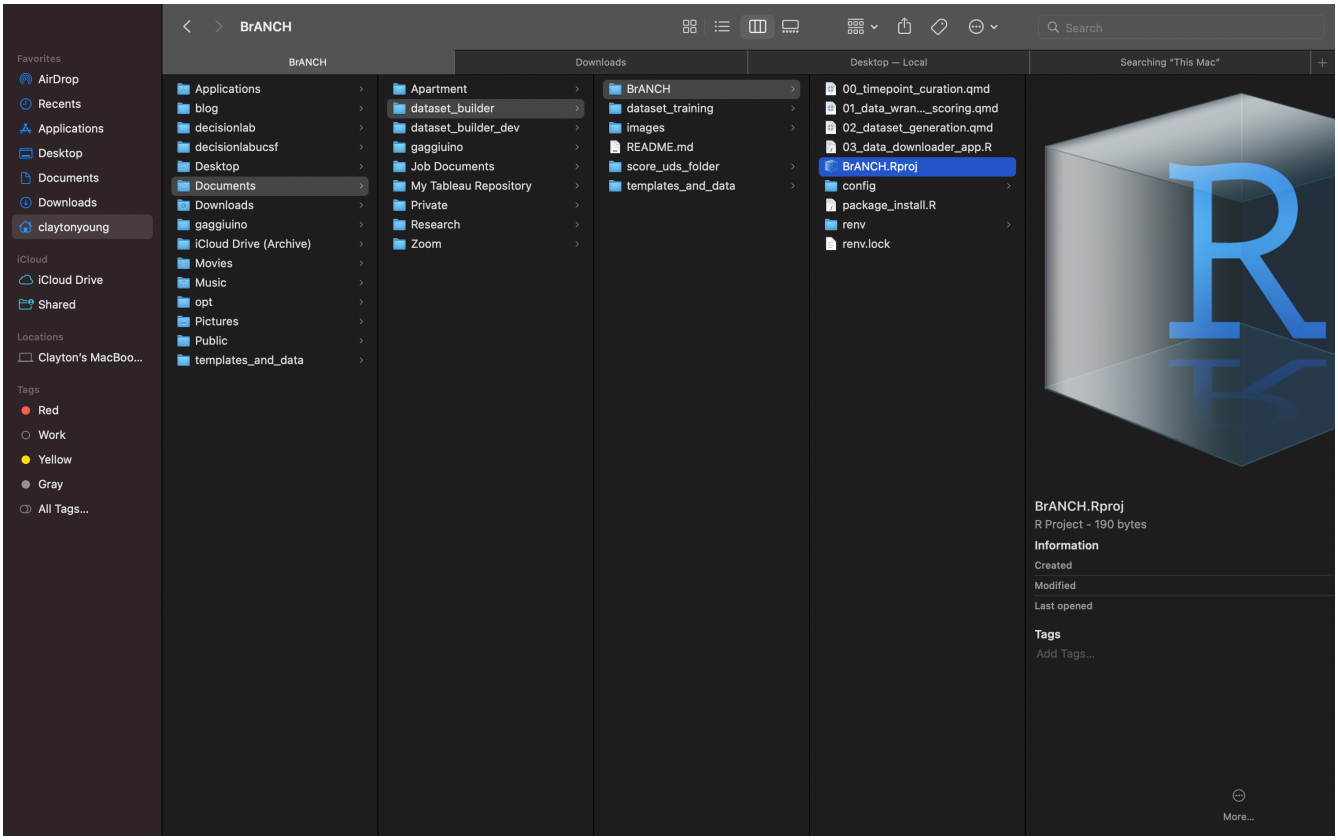


- Choose the local path you want to clone the repo to and click ‘Clone’
    - If using Windows, I recommend the default path: `C:Users/yourusername/Documents/GitHub/` or another place other than ‘Documents’ since Windows treats this as your home directory, which is where other files will be copied to in the next step

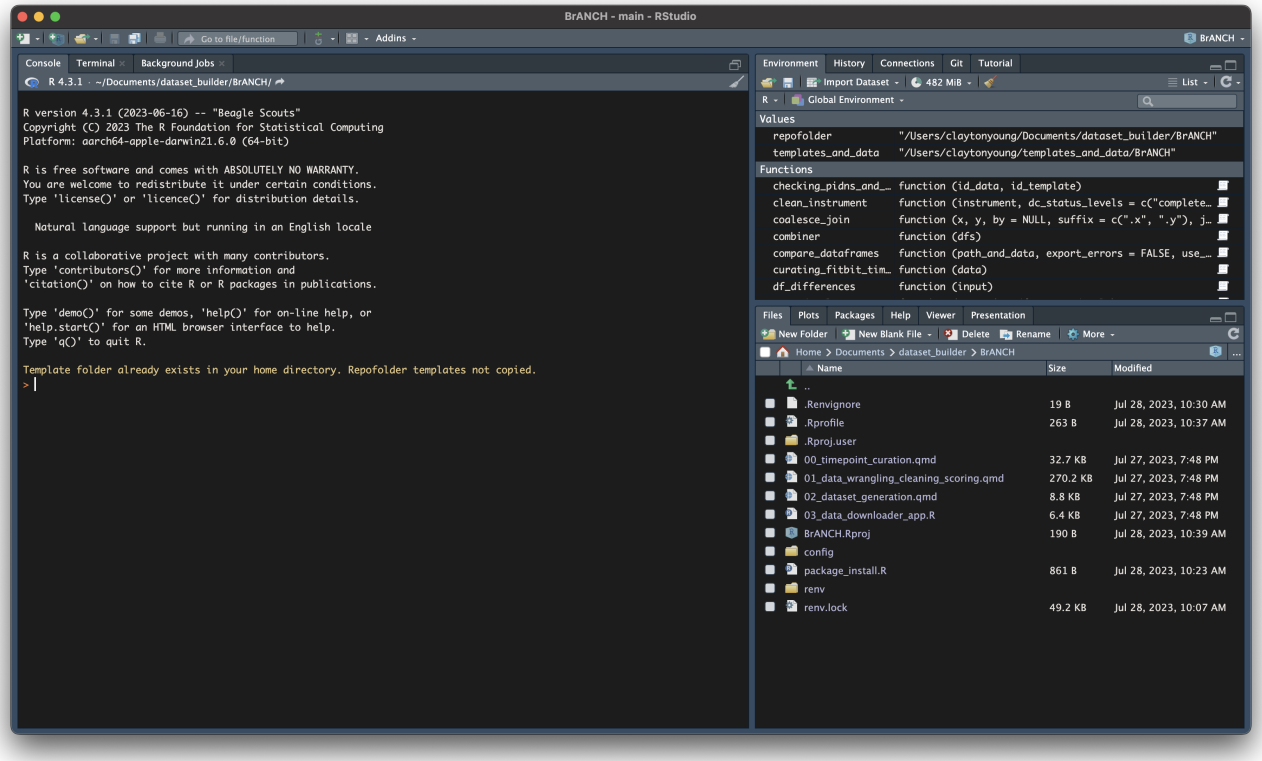


Package installations

- Open the .Rproj file found in the repository you’ve cloned
- For me, it looks like this:



- You will now see the functions and paths loaded to the top right section of the RStudio screen



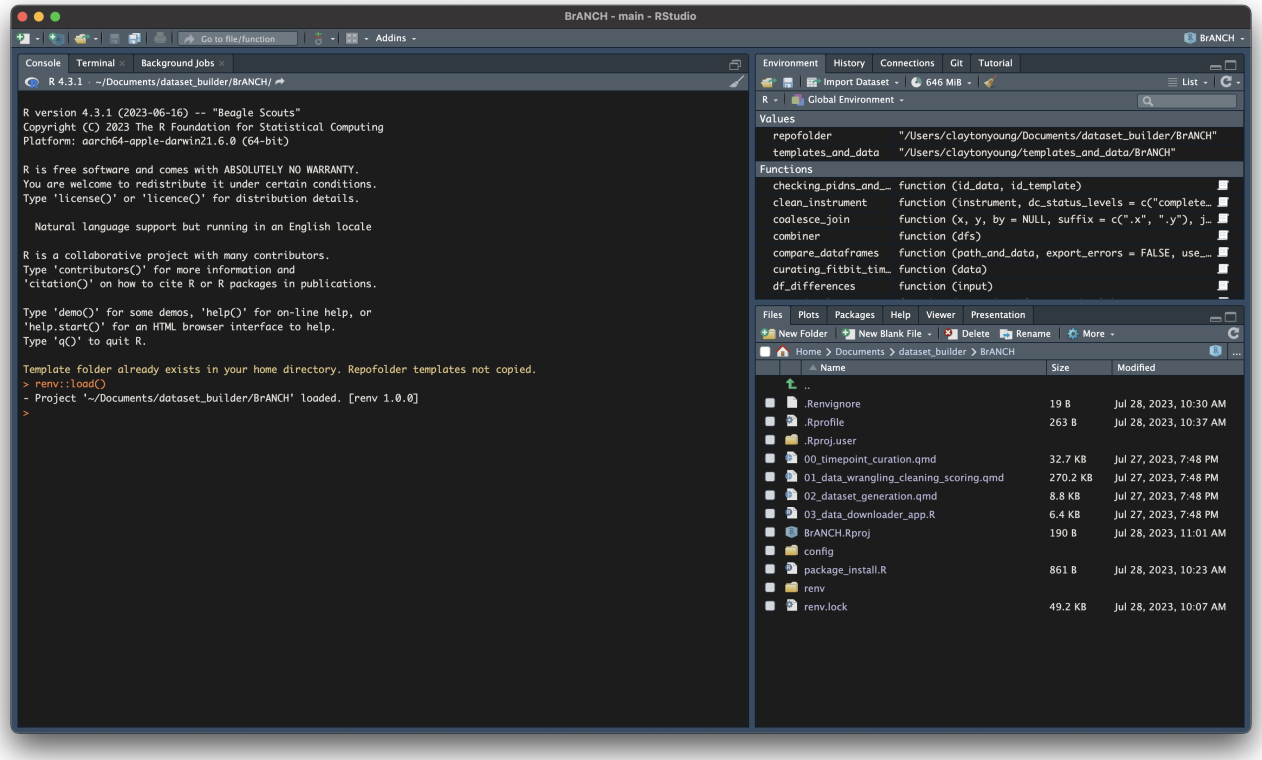
- The setup script also copies the templates and data folder to your home directory
  - It first checks whether that folder exists in your home directory to prevent overwriting the folder
  - See the message in the console in the above picture

Installing Packages with renv

renv is a package that installs the version of other packages for a specific project

- newer/older versions of the packages installed via renv will not overwrite your existing version outside of the project,

1. From the BrANCH.Rproj sesssion in RStudio
2. run `install.packages("renv")`
3. run `renv::load()` or `renv::activate()` to load the info of the projects dependencies
    - `renv::activate()` should prevent you from needing to run `renv::activate` or `renv::load` again in the future by setting activate as the default for the project
4. run `renv::restore()` to install the packages required
    - This will prompt you with the option to install packages
    - If you'd like to install the packages, press "Y" and hit 'Enter' to install them
    - This can take a while the first time you run it, so be patient



Working from Current Data Freeze

Map R Drive (Copied from Macipedia):

REQUIRED: Access to the R Drive

MAC ARF

Mapping Network Drives from Windows 7

1. On the desktop, right-click on the “My Computer” icon (sometime labeled just “Computer”), and choose “Map Network Drive”
    - Alternatively, you can also launch “Windows Explorer”, and right-click on the “Computer” item in the left-hand window.
2. Select Drive Letter that you want to use for the drive.
3. In the Folder field type the server and share name.
4. The general syntax looks like: `\\full.server.name\sharename`
5. For example, The R: drive is mapped using: `\\mac-srv-nas2.ucsf.edu\macdata`
6. Make sure the box “Reconnect at Logon” IS checked.
7. Click Finish.

Mapping Network Drives from MAC OS X

1. Open a Finder Window.
2. Select “**Connect to Server”** from the”**Go”** menu
    - (shortcut: Command+K in Finder)
3. In the “Server Address” box, type in the path to the network drive that you want to connect to with ‘smb://’ in front of the server name and a ‘/’ to separate the server name from the share name. For example:

    For the R: drive, use: `smb://mac-srv-nas2.ucsf.edu/macdata`
4. You will be prompted for a name and password:
    - Click the button that says Connect As: “Registered User”
    - Name: use “campus\<your MAC login>”, for example, “campus\mwschaffer”
    - Password: your UCSF/MAC password.
    - You might want to click the box that says “Remember this password in my keychain”. This will help you next time you map the drive.

Note: You can make an alias to a drive by right-clicking on the mounted drive and selecting “Make Alias”. If you wish to have one or more drives mount automatically at startup, you can drag and drop them in Login Items found in System Preferences > Accounts (make sure you have chosen to save your username & password in your keychain).

Retrieve data freeze:

- From the R Drive, copy the file ‘templates_and_data.zip’ from this path:
    - Windows: `R:/projects/hillblom/Datasets/dataset_builder/`
    - macOS: `Volumes/macdata/projects/hillblom/dataset_builder/`
- Unzip the file and move the entire folder `templates_and_data` with all it’s contents to:

    - Windows: `C:/Users/*yourusername*/Documents/`
    - macOS:

      `~/`

      or

      `/Users/claytonyoung/`

- Moving forward, you'll be working from the current data version of the data freeze

File System Organization

By the end of this session, you should:

1. Understand the file structure and it's purpose

Purpose

- The intention of this file system is to track and validate incoming data
- By logging errors, we're able to see what differs from a valid data input that is used for future transformations
- Versioning the data product and data output files allows us to revert back to an existing file
    - We're also able to see where differences occure by looking through the current and previous versions
- Versioning also allows us to share data products and have others build off them
- One caveat here is that data products are not uploaded to GitHub, so versioned data is stored locally or on a shared drive
    - Thus, the templates are copied to the users local machine so they can work from there as a step to prevent pushing PHI to the repository

Overview of system

- The folder you cloned from the Github repository is located in the path you chose
- This folder is named 'dataset_builder' and should contain a BrANCH, score_uds_folder, and templates_and_data folder at *minimum*
- The files you'll run are located in the cloned repo folder `dataset_builder/BrANCH/` and are named:
    - `00_timepoint_curation.qmd`
    - `01_data_wrangling_cleaning_scoring.qmd`
    - `02_dataset_generation.qmd`
    - `03_data_downloader_app.R`

These files also call on other files within that directory, as well as the `score_uds_folder`.

Home Directory Files

When you open the `BrANCH.Rproj` file, the folder `templates_and_data` will be copied to your home directory.

- On windows, this is `users/yourname/documents/`
- On macOS, this is `users/yourname/` or `~/`
- This attempts to prevent uploading PHI to the GitHub repo

Home Directory File Structure

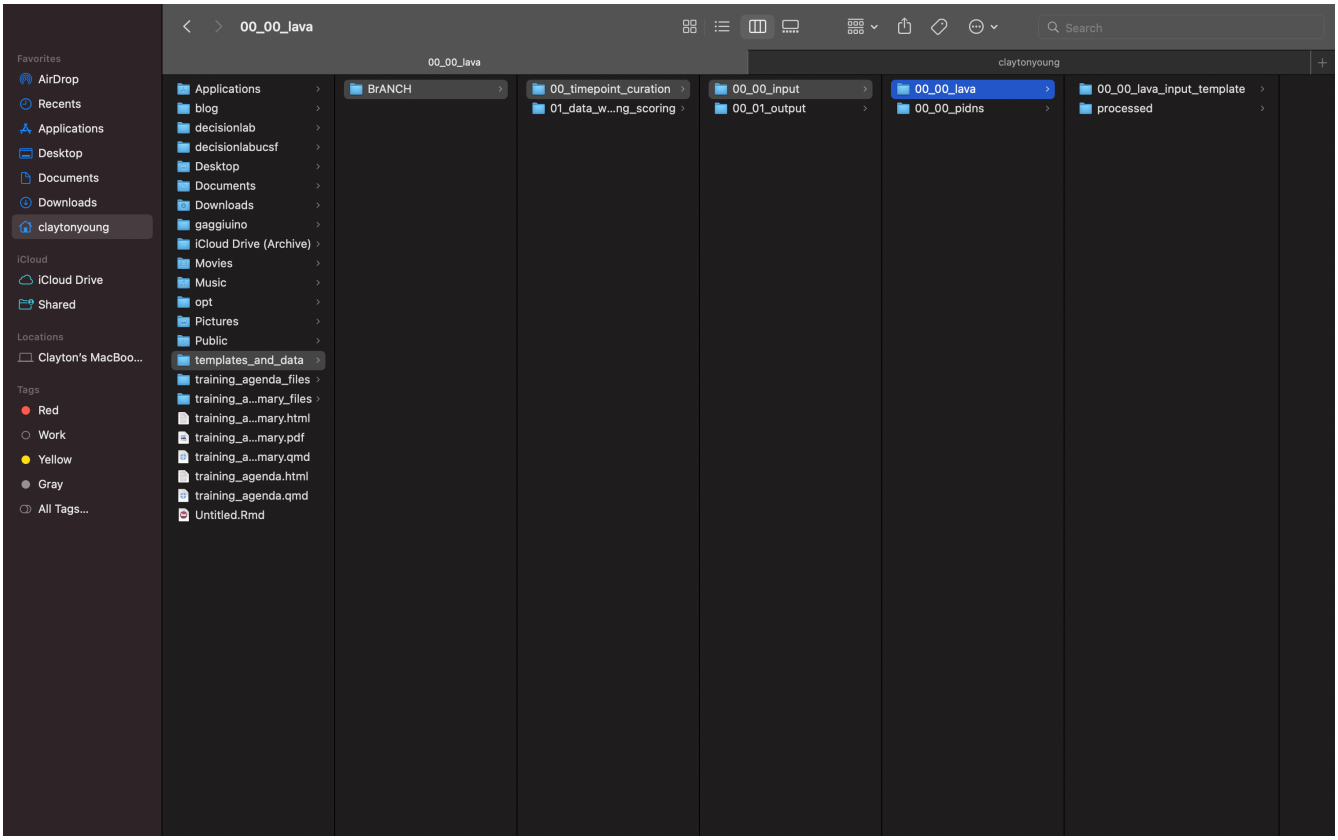
Within the `templates_and_data` folder in your home directory

- A 'BrANCH' folder initially hosts two folders:
  - `00_timepoint_curation`
  - `01_data_wrangling_cleaning_scoring`
- Within these folders, there should *at least* live a subdirectory with 'input' in it's name
  - `00_00_input`
- These 'input' folders contain subfolders where new data should be placed
  - `00_00_lava`
    - The data templates are also in that folder (`00_00_lava`)
      - `00_00_lava_templates`

Example

Overall, it'll look like this:

- You'll place new data in the blue-highlighted directory, so it'll be among the processed and template folder

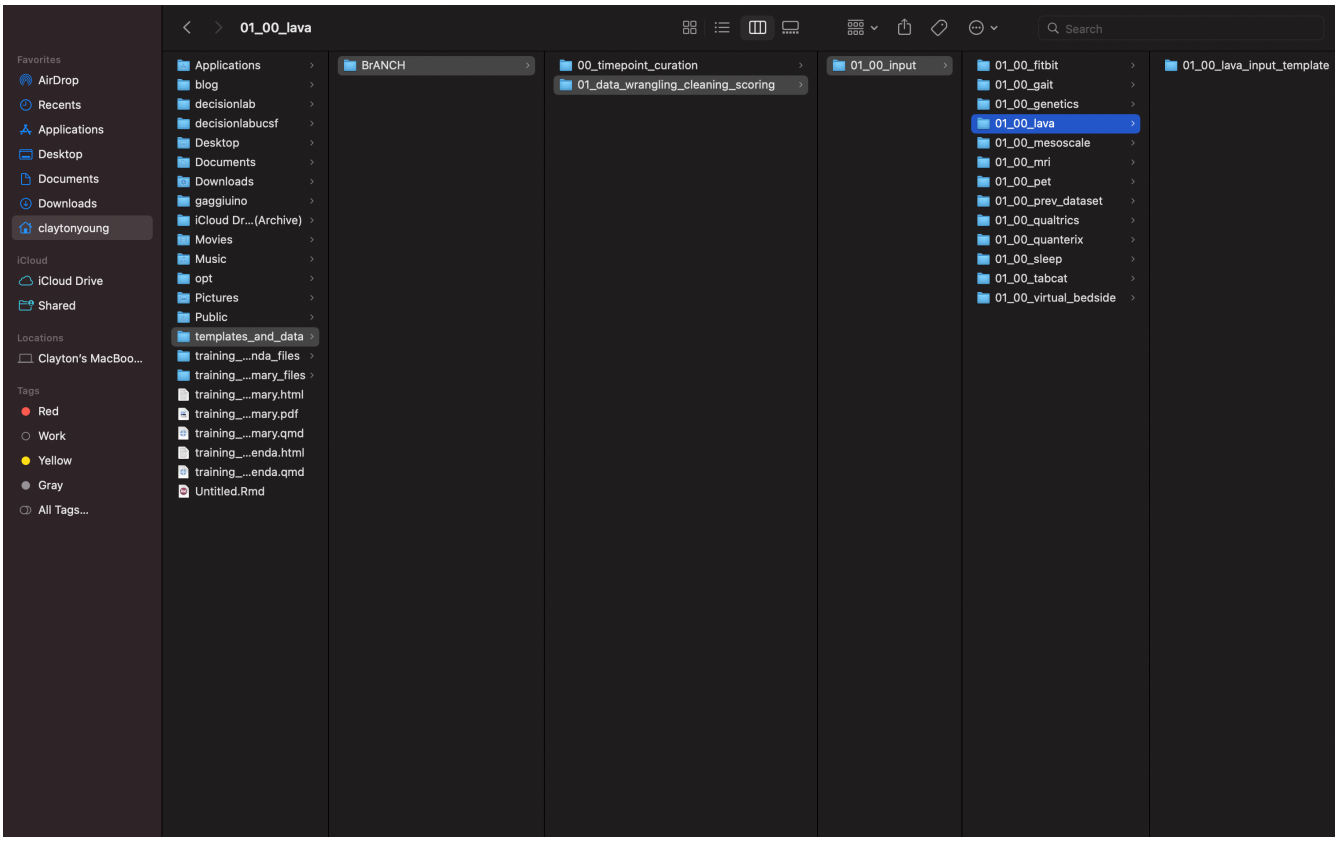


- When running the qmd files, you'll place new data within the folder with that same name
  - When running `00_timepoint_curation.qmd`, you'll place new lava data in the directory pictured above
  - When running `01_data_wrangling_cleaning_scoring.qmd`, you'll place lava data pictured

below:

- Note that this directory differs beginning after the ‘BrANCH’ folder

  `~/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring`



The subfolders in the input folder subdirectories are named with the type of data (e.g., lava, pet, fitbit, etc.) you place in that folder

The Life of a Datafile

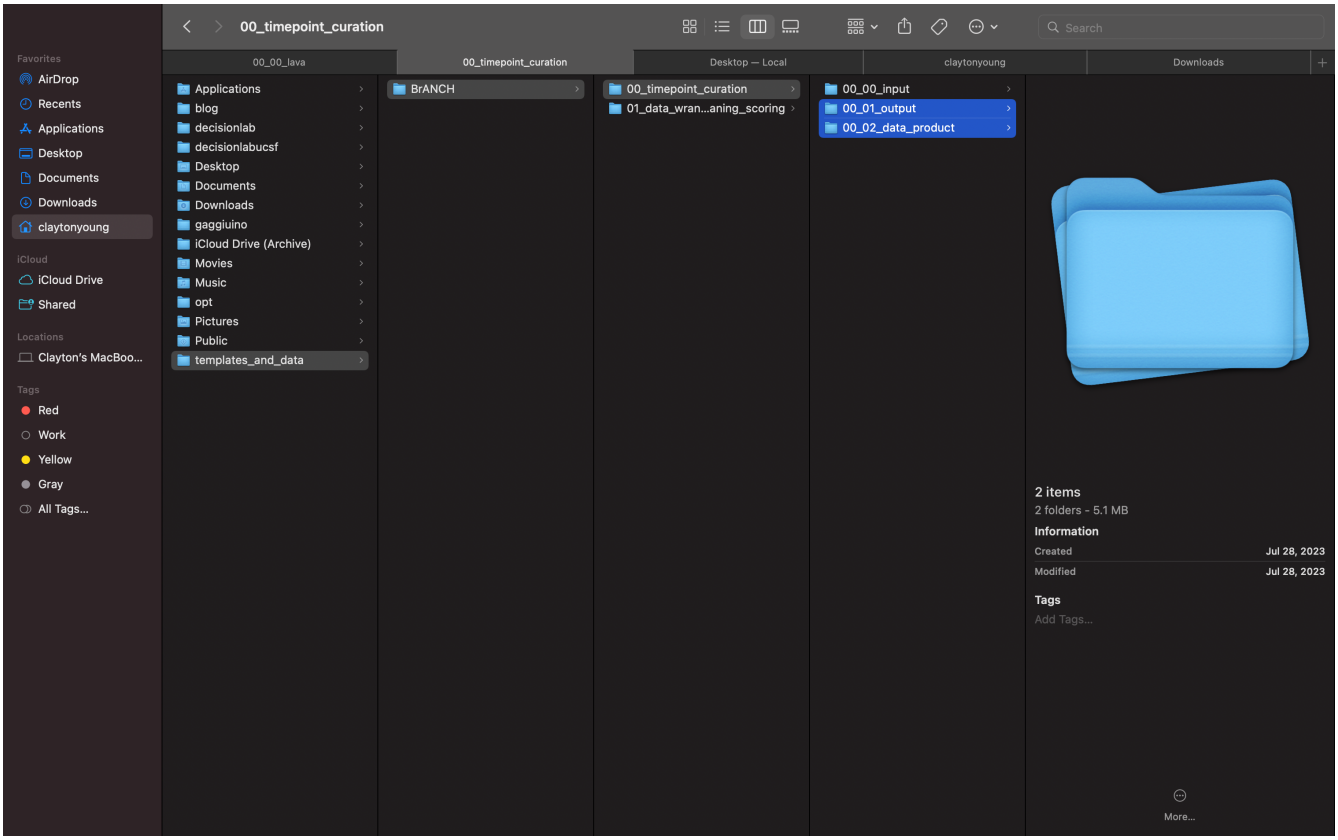
- Data is placed in an input subdirectory (`00_00_lava`)
- Running the qmd file
  - Imports the datafile into R
  - Moves the file to a ‘processed’ folder in that input subdirectory

    `00_00_input/00_00_lava/processed/` * creates a processed folder if one doesn’t exist

  - Compares the data structure to the data template in that input subdirectory

    `00_00_input/00_00_lava/00_00_lava_input_template`

  - If in expected structure, the data is stored in the output directory `00_01_output/00_01_lava` and versioned
  - If not in the expected structure, an `errors` folder will be created in the input subdirectory (`00_00_input/00_00_lava/errors/`), along with an excel file with information about the error
  - At the end of the qmd, a locally versioned storage of the data to be used in the next steps will be saved in a single file in the directory `` `00_timepoint_curation/00_02_data_product ``


00_timepoint_curation
Search
Favorites
AirDrop
Recents
Applications
Desktop
Documents
Downloads
claytonyoung
iCloud
iCloud Drive
Shared
Locations
Clayton's MacBoo...
Tags
Red
Work
Yellow
Gray
All Tags...
00_00_lava
00_timepoint_curation
Desktop — Local
claytonyoung
Downloads
Applications
blog
decisionlab
decisionlabucsf
Desktop
Documents
Downloads
gaggiuino
iCloud Drive (Archive)
Movies
Music
opt
Pictures
Public
templates_and_data
BrANCH
00_timepoint_curation
01_data_wran...aning_scoring
00_00_input
00_01_output
00_02_data_product
2 items
2 folders - 5.1 MB
Information
Created
Jul 28, 2023
Modified
Jul 28, 2023
Tags
Add Tags...
More...

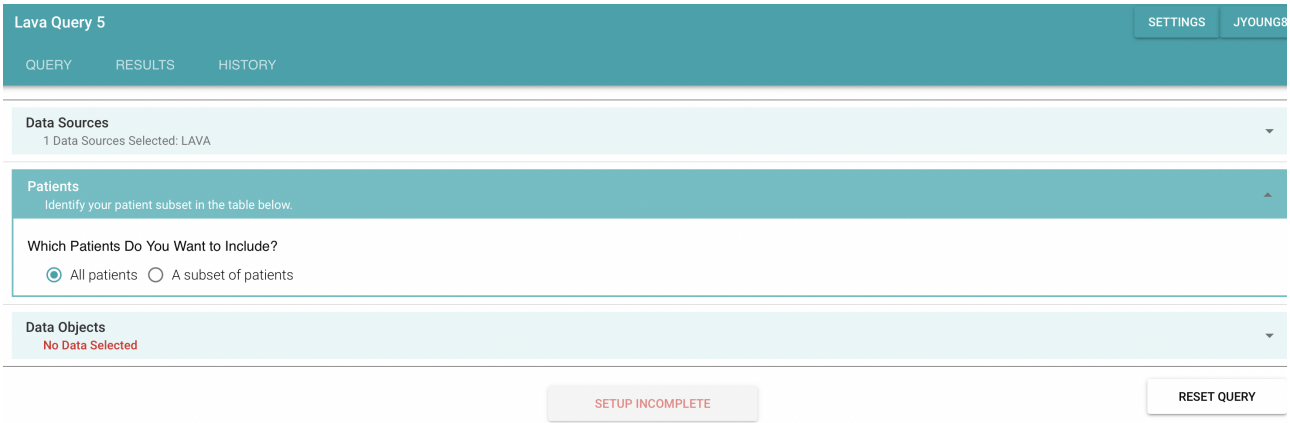
File System Organization and LAVA Query Data

By the end of this session, you should:

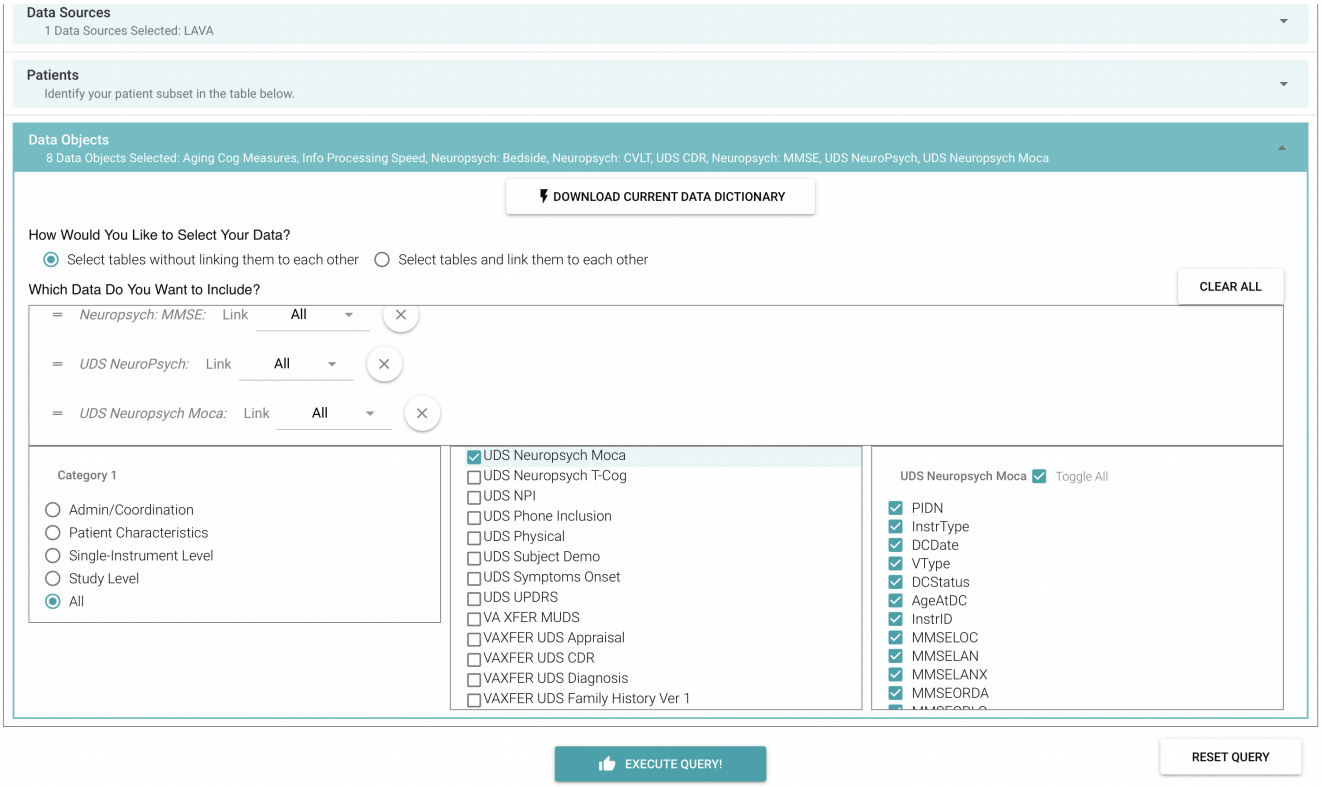
1. Have downloaded LAVA Query data

LAVA Query Data

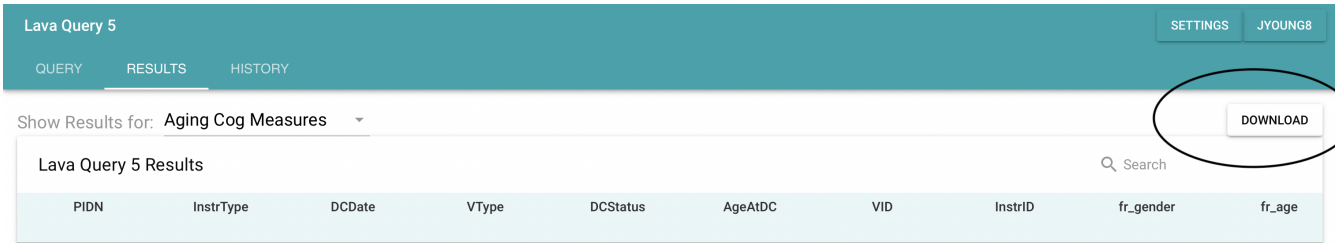
- If offsite, access the VPN (Pulse secure)
- Follow this link and login
  - https://knect.ucsf.edu/query
- From LAVA Query
- Data Sources should only be: LAVA
- Patients: All Patients



- Data Objects:
  - Category 1: All
  - Select these tables:
    - Aging Cog Measures
    - Info Processing Speed
    - Neuropsych: Bedside
    - Neuropsych: CVLT
    - Neuropsych: MMSE
    - UDS CDR
    - UDS NeuroPsych
    - UDS Neuropsych Moca

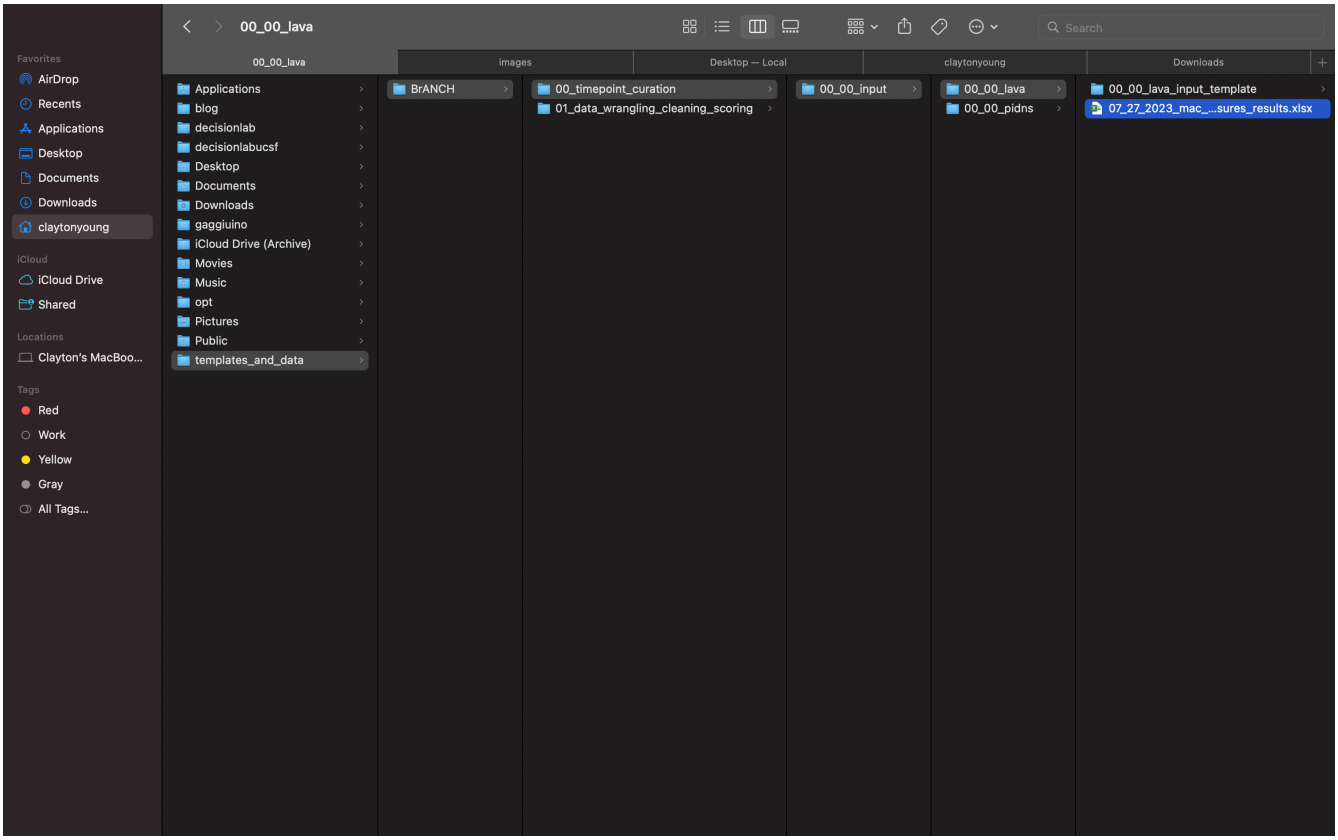


- Execute Query
  - Will likely take a few minutes
- Download the data



Move the File

- Place the dowloaded data in it's correct folder
  - We're using this data for the first part of the process, creating timepoints
  - Since it's lava data, we're placing it within the 00_00_lava folder



- We can now move on to the next step!

Timepoint Curation

By the end of this session, you should:

1. Have run timepoint curation script

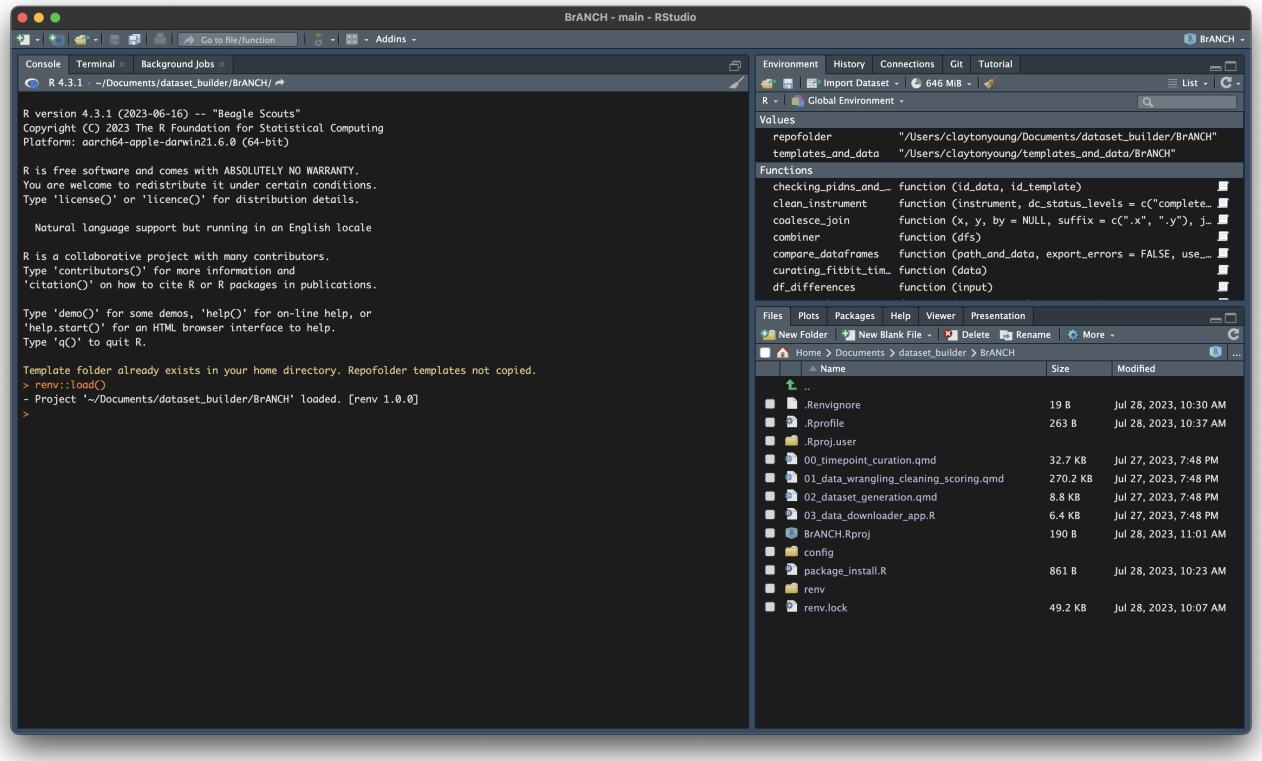
Run Timepoint Curation

- This script generates timepoints for a single OR multiple list of PIDNS
  - In other words, you can create timepoints for multiple datasets by placing separate lists of pidns in the pidn input folder
- **The purpose if generating timepoints is to maximize the amount of data we link to a date**
  - We want to capture the most amount of data possible for each timepoint without duplicating the instrument data linked
  - This algorithm was developed with the guidance of BrANCH (namely, Rowan) and involves comparing instrument data to other date data for each data collection date
- A list of pidns is required to run this script, as well as the LAVA data listed below:

```
-   Aging Cog Measures

-   Info Processing Speed

-   Neuropsych: Bedside

-   Neuropsych: CVLT

-   Neuropsych: MMSE

-   UDS CDR

-   UDS NeuroPsych

-   UDS Neuropsych Moca
```

Open `~/Documents/dataset_builder/BrANCH/BrANCH.Rproj` file

- If using renv and you didn't call renv::activate() during setup, call `renv::load()` in the Rstudio console and hit Enter/Return
  - This will load the installed packages installed from our initial setup
  - If you're notified of additional packages that need to be installed, enter `renv::restore()` in the console
- If you're not using renv, and are notified additional packages are required but are not installed, install those packages by clicking the 'install' button as seen in the setup section



Running 00_timepoint_curation.qmd

- You should have the downloaded LAVA file in the path

`~/templates_and_data/BrANCH/00_timepoint_curation/00_00_input/00_00_lava/`

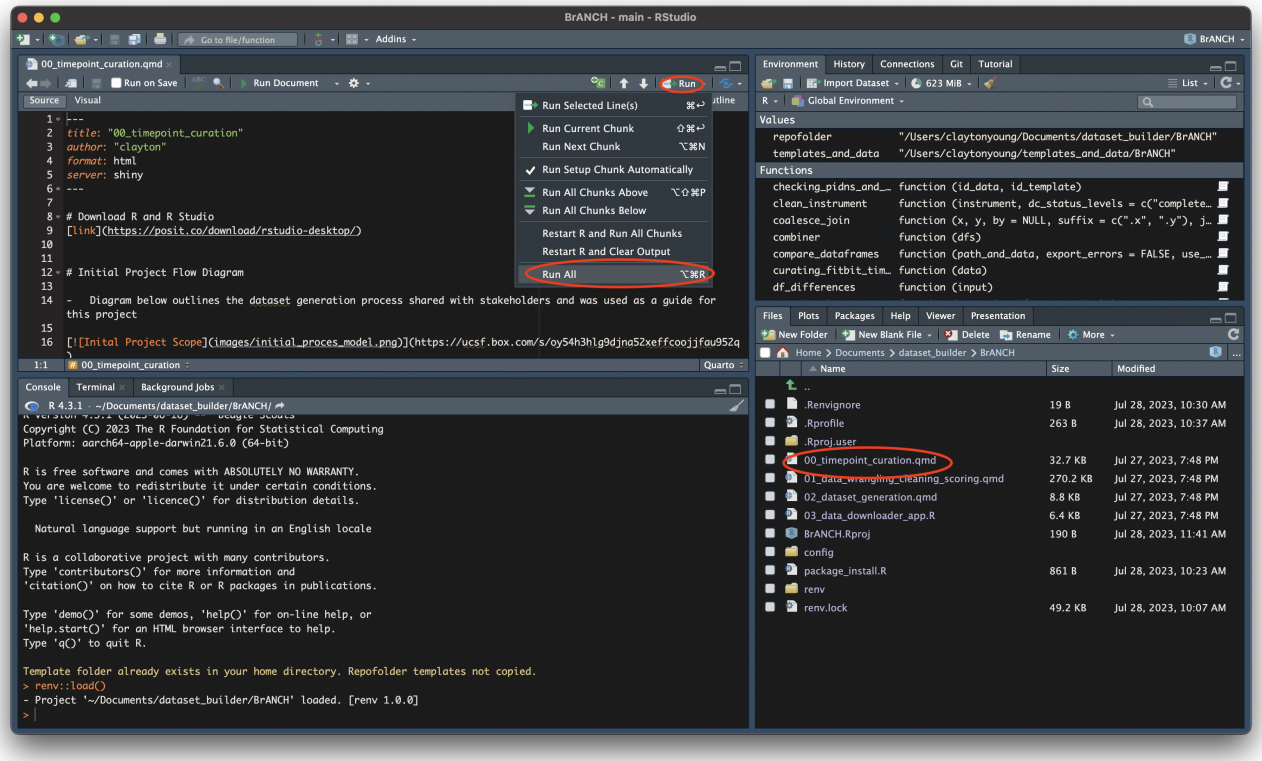
- If using a new list of PIDNs (not test_pidns.csv), move that file to

`~/templates_and_data/BrANCH/00_timepoint_curation/00_00_input/00_00_pidns/`

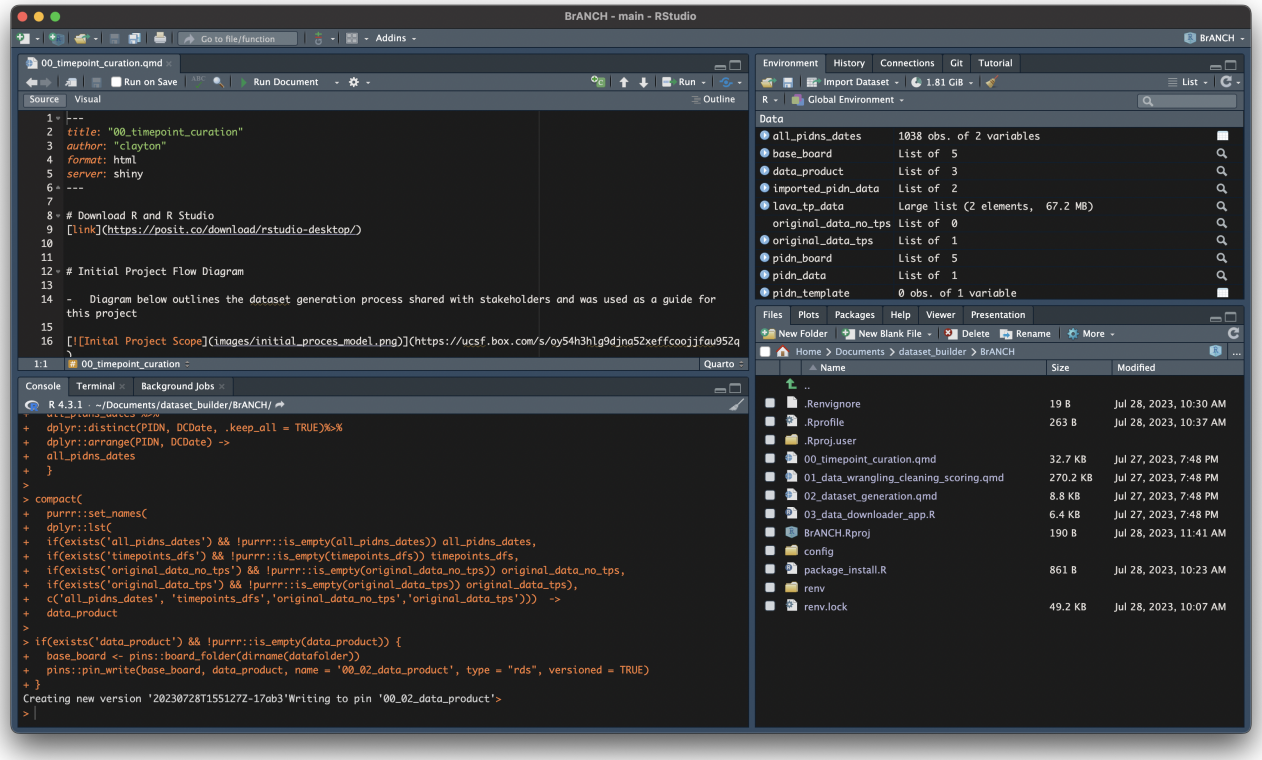
- This only requires that a column 'PIDN' exists in the file
- If there are PIDN and DCDates, it will be treated differently, and timepoints will not be generated
- NOTE: name the file(s) whatever name you want suffixed with '_pidns'
  - e.g., `whateverdatanameyouwant_pidns.csv`

Open `00_00_timepoint_curation.qmd`

- Click run all from the drop down at the top of the screen (circled in red)



- You should now see a message in the console that the data product was created



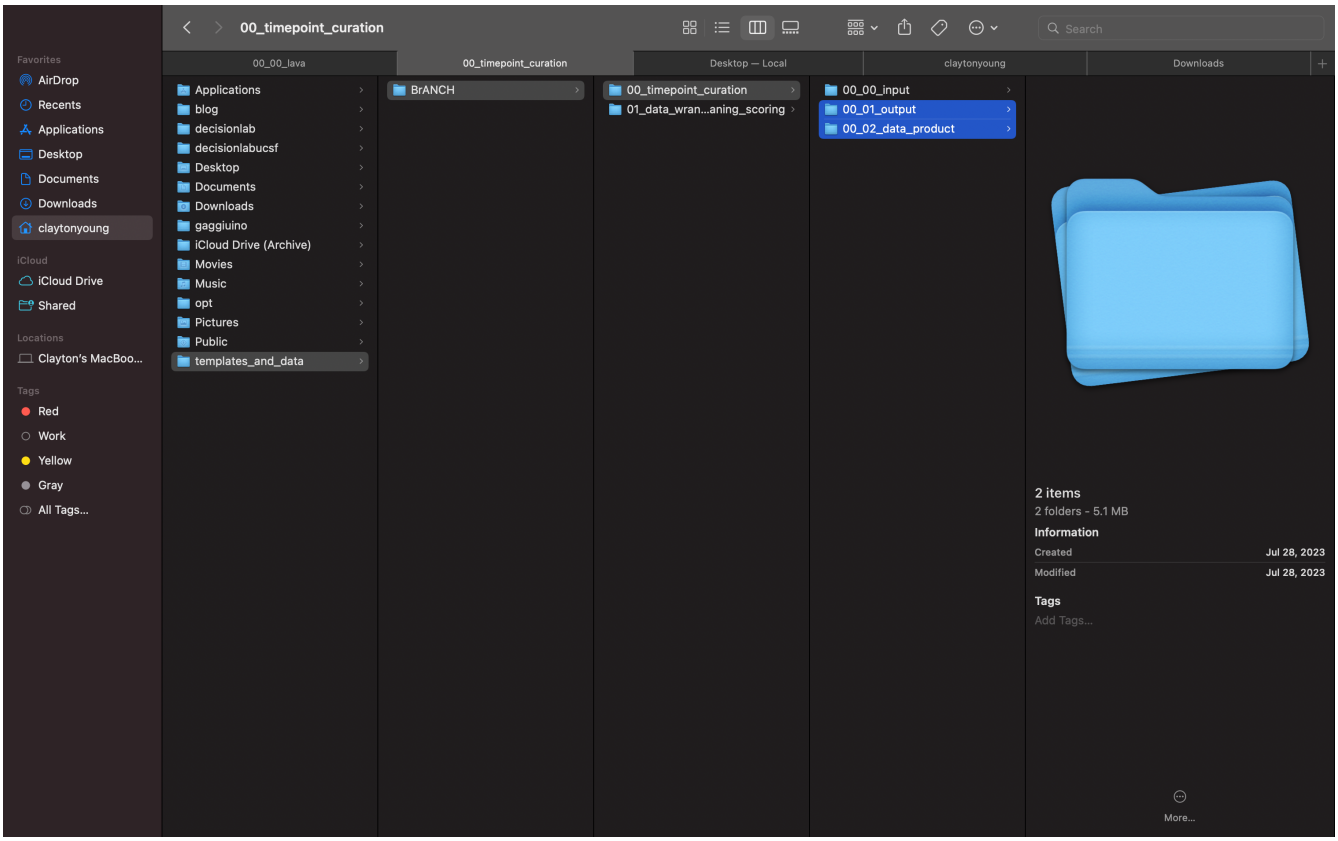
- Check there are no errors in the error folder if it exists

`~/templates_and_data/BrANCH/00_timepoint_curation/00_00_input/00_00_lava/errors`

- If it does exist, inspect the errors and modify the input file or change the arguments in the data_comparison function (we'll get into that later)

- Check there are two new folders (data_product and output) in the directory

`~/templates_and_data/BrANCH/00_timepoint_curation/`



- Our original input folders are moved to the processed folder
- The output of the data (after it's imported) is stored in the output folder as a versioned datafile
- The data product is a versioned data file that contains all the data we'll carry on to our next step

Data Wrangling, Cleaning, and Scoring

By the end of this session, you should:

1. Have run `01_data_wrangling_cleaning_scoring.qmd`

2. Downloaded the datafreeze from the R Drive

3. Downloaded additional LAVA Data

4. Compared output of refresh with datafreeze

Set up Environment

- Jump to setup section

Gather LAVA Data

- As an example, we'll use ADRC NeuroExam
- Using same steps as the previous session, download 'ADRC NeuroExam' from LAVA
- Copy the previous LAVA export for timepoint_curation from

  `~/templates_and_data/BrANCH/00_timepoint_curation/00_00_input/00_00_lava/processed`

- Move/paste the new LAVA data and the copied file to the lava data input folder for the wrangling, cleaning and scoring script path:

  `~/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/01_00_input/01_00_lava/`

Map R Drive (Copied from Macipedia):

Mapping Network Drives from Windows 7

1. On the desktop, right-click on the "My Computer" icon (sometime labeled just "Computer"), and choose "Map Network Drive"
   - Alternatively, you can also launch "Windows Explorer", and right-click on the "Computer" item in the left-hand window.
2. Select Drive Letter that you want to use for the drive.
3. In the Folder field type the server and share name.
4. The general syntax looks like: `\\full.server.name\sharename`
5. For example, The R: drive is mapped using: `\\mac-srv-nas2.ucsf.edu\macdata`
6. Make sure the box "Reconnect at Logon" IS checked.
7. Click Finish.

Mapping Network Drives from MAC OS X

1. Open a Finder Window.
2. Select "**Connect to Server"** from the"**Go"** menu
    - (shortcut: Command+K in Finder)
3. In the "Server Address" box, type in the path to the network drive that you want to connect to with 'smb://' in front of the server name and a '/' to separate the server name from the share name. For example:

    For the R: drive, use: `smb://nsmaclfsmac1.mac-internal.ucsf.edu/macdata`

4. You will be prompted for a name and password:
    - Click the button that says Connect As: "Registered User"
    - Name: use "campus\<your MAC login>", for example, "campus\mwschaffer"
    - Password: your UCSF/MAC password.
    - You might want to click the box that says "Remember this password in my keychain". This will help you next time you map the drive.

Note: You can make an alias to a drive by right-clicking on the mounted drive and selecting "Make Alias". If you wish to have one or more drives mount automatically at startup, you can drag and drop them in Login Items found in System Preferences > Accounts (make sure you have chosen to save your username & password in your keychain).

Retrieve data freeze:

- From the R Drive, copy the file 'templates_and_data.zip' from this path:
    - Windows: `R:/projects/hillblom/Datasets/dataset_builder/`
    - macOS: `Volumes/macdata/projects/hillblom/dataset_builder/`
- Unzip the file and move the folder `templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/\'\'01_02_data_product` to:
    - Windows: `C:/Users/*yourusername*/Documents/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/`
    - macOS:

      `~/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/`
- This 'data freeze' file will now be used to compare and update the new, processed data placed in 01_00_lava

Run `01_data_wrangling_cleaning_scoring.qmd`

- Explore new data freeze built off existing data freeze
- View differences between old and new data

- matching errors, names, old_dfs, new_dfs, etc.

Outputs

New Data Freeze from Scratch

- Run `01_data_wrangling_cleaning_scoring.qmd`
- Without the current datafreeze (01_02_data_product folder), the script will build a new datafreeze based on data available and you'll be presented with this message:

  `No data comparisons present. Creating new data freeze.`

- After you've generated a new datafreeze, you can add new data by dragging the data files to their appropriate input folders and run this again. The freeze will be updated with the new data and carry forward the existing data.
- When a new datafreeze is built, you'll see this output signifying there's no current datafreeze found:

```
>
> # Chunk 106: final_cleaning
> # combines dfs with same name
> combiner(curated_dfs_named) ->
+   curated_dfs_named
>
> # keeps duplicated data with least NA values
> map(curated_dfs_named, clean_instrument, ungroup=TRUE) -> curated_dfs_named
>
> keep(curated_dfs_named, ~nrow(.x) >= 1) -> curated_dfs_named
>
> # Chunk 107: data_freeze
> # combine new and old qualtrics data and use that combination as new_df
> refresh_data(curated_dfs_named,
+              export_errors = TRUE,
+              stop_on_errors = TRUE,
+              save_freeze = TRUE,
+              add_new_instrument = FALSE,
+              overwrite_existing_data = TRUE,
+              expand_cols = TRUE) -> data_freeze
No data comparisons present. Creating new data freeze.
Creating new version '20230906T190211Z-89ffb'
Writing to pin '01_02_data_product'
```

Building Off Existing Data Freeze

No New Data

- If you run the 01 script and there isn't new data in the input directory, you'll receive the output below.

```
> refresh_data(curated_dfs_named,
+              export_errors = TRUE,
+              stop_on_errors = TRUE,
+              save_freeze = TRUE,
+              add_new_instrument = FALSE,
+              overwrite_existing_data = TRUE,
+              expand_cols = TRUE) -> data_freeze
No new data; returning original data.
! The hash of pin "01_02_data_product" has not changed.
• Your pin will not be stored.
>
> # Checking and applying the transformation for fitbit pidns based on refresh
> if ("fitbit" %in% names(data_freeze)) {
+   pluck(timepoint_data, 'original_data_no_tps','fitbit_pidns') <- transform_fitbit_data(data_freeze$fitbit)
+ } else if ("fitbit" %in% names(data_freeze$updated_data)) {
+   pluck(timepoint_data, 'original_data_no_tps','fitbit_pidns') <- transform_fitbit_data(data_freeze$updated_data$fitbit)
+ }
>
> # Writing updated fitbit data to pin
> pins::pin_write(tp_board, timepoint_data, name = '00_02_data_product', type = "rds", versioned = TRUE)
! The hash of pin "00_02_data_product" has not changed.
• Your pin will not be stored.
>
```

New Data

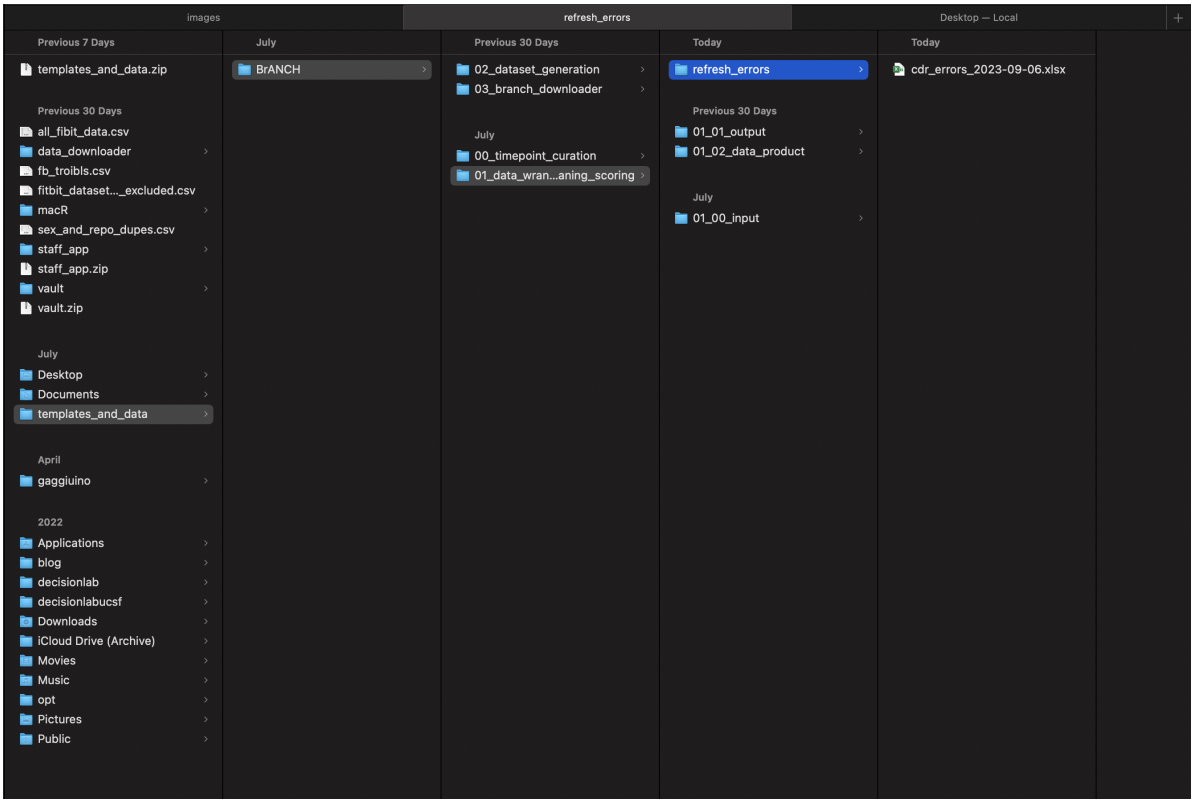
- If 01 is run and there is net new data on an existing instrument, those data will be updated, and you'll see the message below. If save_freeze, the data freeze will be stored as a pin and carried over to the next step (dataset generation).

```
> refresh_data(curated_dfs_named,
+              export_errors = TRUE,
+              stop_on_errors = FALSE,
+              overwrite_existing_data = FALSE,
+              expand_cols = FALSE,
+              save_freeze = TRUE,
+              add_new_instrument = FALSE)->lol
No errors; updating data.
Creating new version '20230907T223042Z-e7b94'
Writing to pin '01_02_data_product'
```

Refresh Data Function Arguments

- The function 'refresh_data' near the end of the markdown contains a few arguments that will dictate the output and datafreeze.
- These arguments are listed below with their default values, along with information about the argument:
  - export_errors = TRUE
    - Export errors will export discrepancies between the old and new data if there are differences detected in an instrument for the same PIDN and DCDate.
    - Example: The CDR values between new data and the existing data freeze have differing values. Since export errors is set to TRUE, we can see these differences in the path in the file with the instrument name and date the refresh_data function was ran:

      `~/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/refresh_errors/`



- The content of the file looks like this:

old			new		
-old[12791,]	1.0	6.0	+new[12791,]	1.0	7.0
-old[14016,]	0.5	3.5	+new[14016,]	0.5	3.0
-old[14022,]	0.5	3.5	+new[14022,]	1.0	4.5

- Alternatively, you can view the differences directly when “stop_on_errors” is set to FALSE by entering `data_freeze$df_matching_errors$cdr` in the console and hitting enter. When typing in the console, this `data_freeze$df_matching_errors$` should generate a drop-down for you to select and view the instrument with discrepancies you wish to investigate.

```
> # combine new and old qualtrics data and use that combination as new_df
> refresh_data(curated_dfs_named,
+              export_errors = TRUE,
+              stop_on_errors = FALSE,
+              overwrite_existing_data = FALSE,
+              expand_cols = FALSE,
+              save_freeze = FALSE,
+              add_new_instrument = FALSE) -> data_freeze
> data_freeze$df_matching_errors$cdr
old vs new
               cdr_global cdr_box
  old[12788, ]        0.5     2.5
  old[12789, ]        1.0     5.0
  old[12790, ]        1.0     4.5
- old[12791, ]        1.0     6.0
+ new[12791, ]        1.0     7.0
  old[12792, ]        0.0     0.0
  old[12793, ]        0.0     0.0
  old[12794, ]        0.0     0.0

old vs new
               cdr_global cdr_box
  old[14013, ]        1.0     7.0
  old[14014, ]        0.5     2.0
  old[14015, ]        0.0     0.0
- old[14016, ]        0.5     3.5
+ new[14016, ]        0.5     3.0
  old[14017, ]        0.0     0.0
  old[14018, ]        0.0     0.0
  old[14019, ]        0.0     0.0
  old[14020, ]        0.5     1.0
  old[14021, ]        0.0     0.0
- old[14022, ]        0.5     3.5
+ new[14022, ]        1.0     4.5
and 3 more ...

`old$cdr_global[14019:14025]`: 0.0 0.5 0.0 0.5 0.0 0.0 0.0
`new$cdr_global[14019:14025]`: 0.0 0.5 0.0 1.0 0.0 0.0 0.0

`old$cdr_box[12788:12794]`: 2.5 5.0 4.5 6.0 0.0 0.0 0.0
`new$cdr_box[12788:12794]`: 2.5 5.0 4.5 7.0 0.0 0.0 0.0

`old$cdr_box[14013:14025]`: 7.0 2.0 0.0 3.5 0.0 0.0 0.0 1.0 0.0 3.5 and 3 more...
`new$cdr_box[14013:14025]`: 7.0 2.0 0.0 3.0 0.0 0.0 0.0 1.0 0.0 4.5           ...
>
```

- Note that the indices are the row numbers, not the PIDN. To view based on the PIDN, you can use the row number to select the row from the old and new instrument data. We'll use row 12791 as an example. You can use these commands below as templates to check the specific rows out and get the all data. It looks like this person's data was updated with a corrected score, so we'll want to overwrite the existing records.

`data_freeze$new_existing_records_data$instrument[row,]`

`data_freeze$old_existing_records_data$instrument[row,]` **

```
> data_freeze$new_existing_records_data$cdr[12791,]
# A tibble: 1 × 5
   PIDN DCDate     instr_type cdr_global cdr_box
  <dbl> <date>     <chr>           <dbl>   <dbl>
1 28160 2023-06-29 CDR                 1       7
> data_freeze$old_existing_records_data$cdr[12791,]
# A tibble: 1 × 5
   PIDN DCDate     instr_type cdr_global cdr_box
  <dbl> <date>     <chr>           <dbl>   <dbl>
1 28160 2023-06-29 CDR                 1       6
```

- stop_on_errors = FALSE
  - This argument stops the data_freeze object from being written when any error is detected. While it's FALSE, you can investigate the data_freeze object to see info about the refresh.
  - When set to TRUE, you'll see this message when there are differences between the data freeze and new data:

    `Error in process_errors(df_matching_errors, export_errors, stop_on_errors, : ERROR DETECTED; function halted.`

- overwrite_existing_data = FALSE
  - This function is used to overwrite the existing records where differences were detected.
  - When set to FALSE, the original data is kept. Using the same cdr data above, we can see the object we use as the datafreeze, 'updated_data' kept the old record.

```
> data_freeze$updated_data$cdr%>%filter(PIDN == 28160)
# A tibble: 2 × 5
   PIDN DCDate     instr_type cdr_global cdr_box
  <dbl> <date>     <chr>           <dbl>   <dbl>
1 28160 2019-10-07 CDR                 1     4.5
2 28160 2023-06-29 CDR                 1     6
```

  - When set to TRUE, the new record was used to overwrite the existing record

```
> data_freeze$updated_data$cdr%>%filter(PIDN == 28160)
# A tibble: 2 × 5
   PIDN DCDate     instr_type cdr_global cdr_box
  <dbl> <date>     <chr>           <dbl>   <dbl>
1 28160 2019-10-07 CDR                 1     4.5
2 28160 2023-06-29 CDR                 1     7
```

- expand_cols = FALSE

- Some instruments may have new columns as time goes on. This argument is used to keep the original set of columns that were in the previous data freeze.
- Creating a new column in cdr instrument in our list of curated dataframes as below and only taking a few rows to avoid the values differences

```
curated_dfs_named$cdr <- head(curated_dfs_named$cdr%>%mutate(lol_new_column = 1))
```

- These column differences are detected and can be viewed the same way as the other differences (excel export or viewed in console).

```
> data_freeze$df_matching_errors$cdr
`old` is length 5
`new` is length 6

`names(old)[3:5]`: "instr_type" "cdr_global" "cdr_box"
`names(new)[3:6]`: "instr_type" "cdr_global" "cdr_box" "lol_new_column"

`old$lol_new_column` is absent
`new$lol_new_column` is a double vector (1, 1, 1, 1, 1, ...)
```

- If we want to accept new columns, set expand_cols to TRUE. **These columns won't be populated in existing data unless overwrite_exisiting_data is set to TRUE**. Otherwise, only new data will have values in those new columns. First is an example of when overwrite is FALSE. Second is when overwrite is TRUE.

```
> refresh_data(map(curated_dfs_named, head),
+              export_errors = TRUE,
+              stop_on_errors = FALSE,
+              overwrite_existing_data = FALSE,
+              expand_cols = TRUE,
+              save_freeze = FALSE,
+              add_new_instrument = FALSE) -> data_freeze
> data_freeze$updated_data$cdr
# A tibble: 14,031 × 6
    PIDN DCDate     instr_type cdr_global cdr_box lol_new_column
   <dbl> <date>     <chr>           <dbl>   <dbl>          <dbl>
 1     2 2005-03-29 CDR               1       8             NA
 2    16 1999-01-04 CDR               0.5     2             NA
 3    18 1999-09-30 CDR               1       5             NA
 4    21 1999-09-10 CDR               2      12             NA
 5    27 2003-06-11 CDR               0.5     3.5           NA
 6    27 2004-08-11 CDR               0.5     3             NA
 7    27 2005-08-12 CDR               1       4.5           NA
 8    52 1999-10-14 CDR               3      15             NA
 9    54 1999-09-30 CDR               2      12             NA
10    65 2009-02-20 CDR               0       0             NA
# ℹ 14,021 more rows
# ℹ Use `print(n = ...)` to see more rows
> refresh_data(map(curated_dfs_named, head),
+              export_errors = TRUE,
+              stop_on_errors = FALSE,
+              overwrite_existing_data = TRUE,
+              expand_cols = TRUE,
+              save_freeze = FALSE,
+              add_new_instrument = FALSE) -> data_freeze
> data_freeze$updated_data$cdr
# A tibble: 14,031 × 6
    PIDN DCDate     instr_type cdr_global cdr_box lol_new_column
   <dbl> <date>     <chr>           <dbl>   <dbl>          <dbl>
 1     2 2005-03-29 CDR               1       8              1
 2    16 1999-01-04 CDR               0.5     2              1
 3    18 1999-09-30 CDR               1       5              1
 4    21 1999-09-10 CDR               2      12              1
 5    27 2003-06-11 CDR               0.5     3.5            1
 6    27 2004-08-11 CDR               0.5     3              1
 7    27 2005-08-12 CDR               1       4.5           NA
 8    52 1999-10-14 CDR               3      15             NA
 9    54 1999-09-30 CDR               2      12             NA
10    65 2009-02-20 CDR               0       0             NA
# ℹ 14,021 more rows
# ℹ Use `print(n = ...)` to see more rows
```

- save_freeze = FALSE
  - This saves the freeze in a pin (locally stored versioned rdata file) in the 01_02_data_product folder when set to TRUE. If stop on errors is set to FALSE, it will write the data freeze when there are differences between the current freeze and new data.
- add_new_instrument = FALSE
  - This can be used to include new data in to the data freeze.
    - As an example, we can define a new df as lolz, which is just a copy of the existing cdr dataframe. Also removing the temp column in cdr.

      ```
      curated_dfs_named$lolz <- curated_dfs_named$cdr
      ```

```
curated_dfs_named$cdr <- curated_dfs_named$cdr%>%select(-lol_new_column)
```

- Setting add_new_instrument to TRUE, we can see a message confirming the new dataframe is included in the data freeze.
- Note that to add a new instrument, you should add the df's name to the list of objects in the `curated_instruments` chunk for it to be included in the `curated_dfs_named` object

```
> refresh_data(map(curated_dfs_named, head),
+              export_errors = TRUE,
+              stop_on_errors = TRUE,
+              overwrite_existing_data = FALSE,
+              expand_cols = FALSE,
+              save_freeze = FALSE,
+              add_new_instrument = TRUE) -> data_freeze
No new data; returning original data.
Adding new instrument(s).
```

Unique Cases

Reprocessing files

- If you've run `01_data_wrangling_cleaning_scoring.qmd` with new data, those data will be moved to the 'processed' folder
  - To rerun the dataset builder with those processed data:
  - ***Move*** the input data files from their 'processed' folders back to the input directory (e.g., 01_00_lava)
    - If you ***copy*** the files to their input directory and try rerunning the import part of the script, you'll see an error saying that file is already imported. This is because the same file(s) exist in the 'processed' folder *and* the input directory (e.g., 01_00_lava)
    - If you received the error above, delete the duplicated file in the 'processed' folder or the entire 'processed' folder itself if all files also exist in the input directory. We'll start fresh and the script will create that folder and move the file to it once it's read in.

Importing New Instruments

- See the New Instrument Template in 01_data_wrangling_cleaning_scoring.qmd
  - place new data in 01_00_new_instr folder
  - set the name mapping here
    - new_name_mapping ← c("raw_file_name" = "clean_file_name")
  - modify data as desired here and assign the name

```
if(!empty(pluck(new_instr_comparison_output,      'dfs',      'clean_file_name')))      {
pluck(new_instr_comparison_output,      'dfs',      '\'\'clean_file_name\'\'')      %>%
mutate(instr_type = 'Brief_lol') ->\'\'new_assigned_name }
```

  - Add assigned object to data_names in the header "Compile List of DFs"

        - e.g.,

          ```
          data_names <- c("bedside","new_assigned_name"
          ```

- We'll use new LAVA data as an example.
    - Download a new instrument from LAVA
    - Place in:

      ```
      ~/templates_and_data/BrANCH/01_data_wrangling_cleaning_scoring/01_00_input/01_00_lava
      ```

    - Modify the `compare_dataframes` (currently line 43) arguments in `01_data_wrangling_cleaning_scoring.qmd`
        - The function will read in all data in that folder and move the file to 'processed' in that subdirectory.
        - When the argument stop_on_error = TRUE in the compare_dataframes argument and there is a new instrument or an instrument that doesn't match it's template, the object lava_comparison_output will not be assigned. However, differences in the incoming instrument data and the template will be exported if export_errors is set to TRUE.
        - To only use instruments that currently have a template when there are additional instruments, set the arguments to:

          ```
          export_errors = TRUE, stop_on_error = FALSE, use_all = TRUE
          ```

        - To add new instrument templates **and** use the instrument, use the arguments:

          ```
          lava_data, export_errors = TRUE, stop_on_error = TRUE, use_all = TRUE, add_templates = TRUE
          ```

        - To modify an existing template when there are additional instruments you don't want to add, run the compare_dataframes with these arguments TWICE

          ```
          export_errors = TRUE, stop_on_error = FALSE, use_all = FALSE, add_templates = FALSE, replace_existing_templates = TRUE
          ```
          ** The first time will overwrite the existing template and the second time will incorporate the df based on that template you just added.

        - To modify an existing **and** add new templates, run this once:

          ```
          export_errors = TRUE, stop_on_error = FALSE, use_all = TRUE, add_templates = TRUE, replace_existing_templates = TRUE
          ```

        - To modify an existing template and use that dataframe when there aren't additional new dataframes you want to include/exclude, run

          ```
          export_errors = TRUE, stop_on_error = FALSE, use_all = TRUE, add_templates = FALSE, replace_existing_templates = TRUE
          ```

```{r import_lava_data, warning=TRUE, echo=FALSE, message=TRUE, results='hide'}
lava_path <- file.path(datafolder,'01_00_lava')

process_and_move_files_wrapper(lava_path, 'lava_data', modify_function = function(df) {
  df %>%
    set_names(~stringr::str_sub(stringr::str_remove_all(.x,c('mac_|_x.*|x\\d{2}_\\d{2}_\\d{4}_|_results')), start=0, end=27)) %>%
    purrr::map_if(~"DCDate" %in% names(.x), ~mutate(.x, DCDate = lubridate::as_date(DCDate))) %>%
    purrr::map_if(~"PIDN" %in% names(.x), ~mutate(.x, PIDN = as.double(PIDN)))
  })

#in case file was uploaded twice, combine dfs with same name from import (same lava instrument)
pluck(lava_data, 'all_processed_data') <- combiner(pluck(lava_data, 'all_processed_data'))

compare_dataframes(lava_data, export_errors = TRUE, stop_on_error = TRUE) -> lava_comparison_output

pluck(lava_comparison_output, 'dfs') -> lava
```

- Add name of instrument to “data_names” (currently line 6170)
  - Note: name was processed to snake_case and is stored in the lava object
  - Find object and save outside of lava object
    - type:
      - lava$name, where ‘name’ is the name of the instrument. Autofill will help if you find it if you type slow
      - assign the name of the object (using social function as an example)
      - social_function ← lava$social_function
      - add the name of your instrument to e.g., ‘social_function’ to the list of names of dataframes in data_names
- Modify the data refresh to include the new instrument
- `refresh_data(curated_dfs_named, export_errors = TRUE, stop_on_errors = FALSE, save_freeze = TRUE, add_new_instrument = TRUE) -> data_freeze`

Removing Instrument from Freeze

- In some cases, there are errors or other issues in the datafreeze data. In this case, you’ll need to remove the data entirely from the freeze, which is a bit of a manual process.
- To remove an instrument, load the datafreeze `refresh_data(tibble()) -> temp`
- You’ll see the message:

```
No new data; returning original data.
```

- Define the freeze board:

```
freeze_board <- pins::board_folder(dirname(datafolder))
```

- Set the instrument you wish to remove to NULL

```
temp$cdr <- NULL
```

- Write the pin of the datafreeze with the instrument removed (named temp) and include a description in the freeze

```
pins::pin_write(freeze_board, temp, name = '01_02_data_product', type = "rds", versioned = TRUE, description = 'removed cdr instrument')
```

- You’ll see confirmation of the pin being written. Future datafreezes you create will use this

version. You're also able to revert back to the previous version, but I won't cover that here.

```
Creating new version '20230907T230846Z-193e3' Writing to pin '01_02_data_product'
```

Collaboration

- Sometimes there will be multiple users building concurrently
- This process should be done chronologically with the shared product from the other
  - e.g.,
    - Emily refreshes fitbit data
    - Val refreshes lava data
    - Val should take the output of Emily's latest data product from 01_02_data_product and place it in her 01_02_data product and ensure this is the most recent data product within that directory

My data isn't updating

- Sometimes the builder pulls in data from multiple sources to combine into a single dataframe
- Ensure all data is available or modify as desired
- e.g., the bedside data requires multiple instruments.
  - If you didn't include all of these, the data will not be refreshed:

    ```
    if(!empty(pluck(lava, 'bedsidescreen')) && !empty(pluck(lava, 'bedsidescreen_witholdvars')) && !empty(pluck(lava, 'neuropsychbedside')) && !empty(pluck(lava, 'neuropsychcvlt')) && !empty(pluck(lava, 'bedsidealternates'))) {
    ```

  - Same with CHAMPS:

    ```
    if(!empty(pluck(fresh_qualtrics_data, 'champs')) &&nrow(pluck(fresh_qualtrics_data, 'champs')) >= 1 &&!empty(pluck(lava, 'hbudsphysical')) &&!empty(pluck(lava, 'udsphysical'))) {
    ```

- There are multiple cases of this, so verify all data has been imported!

Dataset Generation and Addition of New Instruments

By the end of this session, you should have:

1. Stored a pin of the test_pidns dataset

2. Run the dataset downloader

3. Downloaded csv and excel versions of the dataset(s)

4. Added a new instrument to the data freeze

Run `02_dataset_generation.qmd`

- If 01_02_data_product and 00_02_data_product exists, 02_dataset_generation will retrieve the data freeze and timepoints to generate a pin of the datasets, saving the generated datasets in a single pin in the folder

  `~templates_and_data/BrANCH/02_dataset_generation/02_00_data_product`

- The App in the next step will read this pin for you to select the dataset and variables you're interested in.

Run `03_data_downloader.R`

Select Data Source

- Select which dataset you want to download data from using this drop-down
- This reads in the pin created in the last step (`02_00_data_product`)

Column selection

- Select which variables to download either by instrument or by column (suffixed with instrument)
  - Selecting data by instrument select all columns in that instrument
  - Selecting by column name only selects those columns and required columns (PIDN, DCDate, etc)
  - You can select using a combination of both the name of the column and instrument
- If you'd like to select those columns again in the future, you can download a csv of those columns once you've selected them using the button

  `Export List of Selected Columns` ** Upload the csv using browse

  + `Upload CSV with Column/Instrument Names`

    - The app will select those columns if they exist in the dataset

Export Options

- Saving as a CSV downloads the dataset in long format with the instrument suffixed
    - Columns are suffixed to prevent duplicated columns
- Saving as Excel downloads the data with each instrument's data as it's own sheet within a single xlsx file