WATER AFFORDABILITY

Lauren Patterson
July 2021

Introduction

The water affordability analysis was made possible by having access to public data. These data are provided by:

(1) the federal government
    a. Census Bureau provides census data (household and income characteristics)
    b. Bureau of Labor Statistics provides employment data
    c. Environment Protection Agency provides Safe Drinking Water data (water system characteristics)
(2) Individual states that provide geospatial service area boundaries for their water systems
(3) Individual utilities that provide information on the rates they charge customers

The availability and ease of finding and accessing these data varies considerably. Once the data were collected, the spatial boundaries and rates data must be standardized into similar formats. The data census and water service data must also be spatially joined to estimate what water service rates apply to which households. Joining these data together enables us to calculate multiple affordability metrics for hundreds of utilities.

Aside from the effort of finding and standardizing rates data, the majority of the process has been automated. We are working on developing an online survey to allow others to enter rates data so that the rates database can be updated over time. Additionally, as years of data are added to the survey, a time series of rates will be built to show how rates are changing. The rates database will be open and available to the public and we encourage users to provide the website or source for which rates were found (recognizing many of the websites change and update over time).

Here, we describe the process of finding, accessing, and using the data to calculate affordability metrics (Figure 1). We have automated much of this effort to allow the dashboard to be continuously updated as new data become available. All scripts are written in R and are organized by whether the script is "access"ing data or "use"ing the data. The number refers to the order by which to run the script since some scripts depend on earlier stages being completed. If the numbers are the same they can be run in parallel and are not dependent on each other.

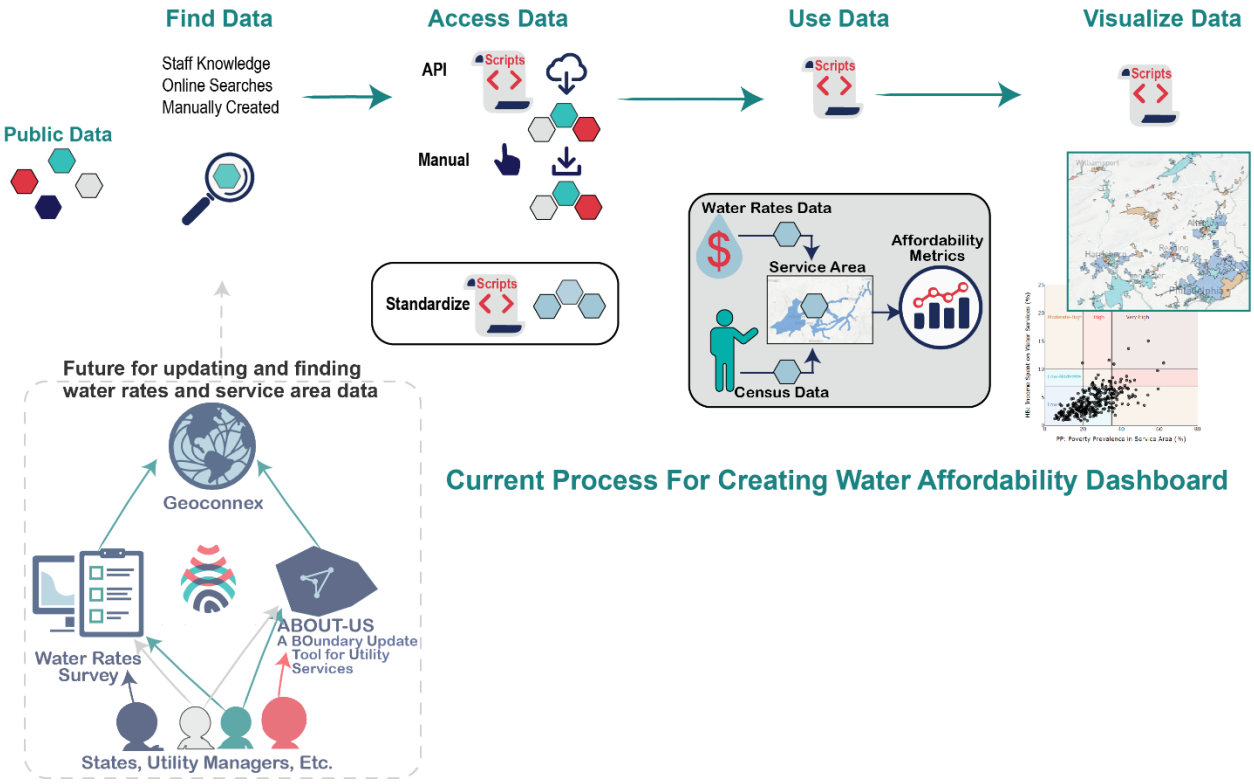


Figure 1. Process for finding, accessing, using and visualizing data in the water affordability dashboard. The continual updating and addition of new data will likely be through using tools currently in development by the Internet of Water.

Finding Data

To understand affordability, we must know something about the households and incomes in the communities being provided water services. Census data provides information regarding poverty, household income, and demographic characteristics. These data are accessed using the census API, with the exception of historic population data, which came from a manually download from https://www.nhgis.org/. In 2021, we used census data were ACS 5-year data for 2019. To find census data, you must know the variable names of interest. These names can be found here: https://api.census.gov/data/2019/acs/acs5/variables.html. The live version of the water affordability dashboard uses the 2019 data 5-year ACS data, and will be updated annually.

The development of this dashboard coincided with the COVID-19 pandemic, where large-scale unemployment took place quickly and for a sustained amount of time. We assume that as unemployment increases, the capacity for households where jobs were lost to pay for services will decrease. As such, we use data from the Bureau of Labor Statistics to understand how unemployment has changed in the counties intersecting water utility service areas.

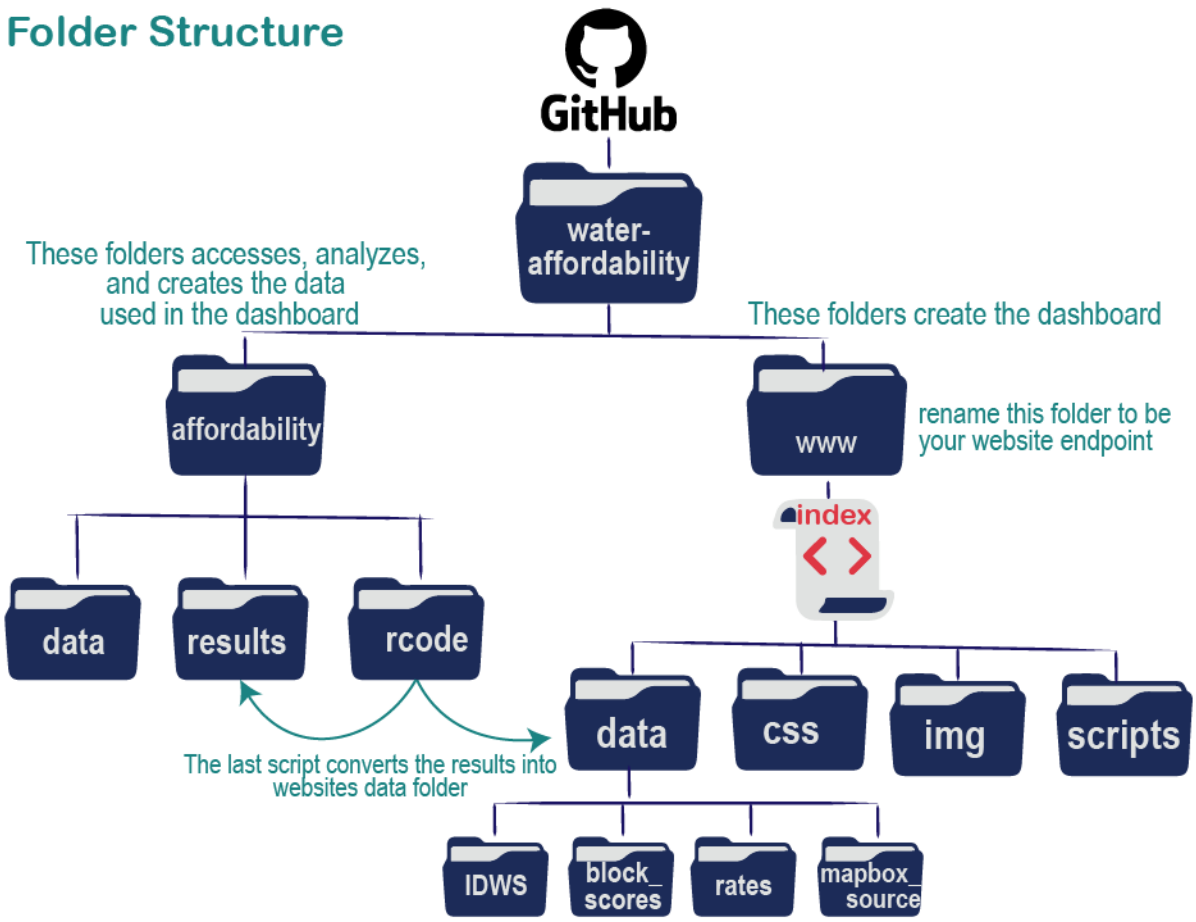
A few states provide the service area boundaries of water service providers, particularly drinking water. The service area boundaries are essential to link the cost of providing services with the households served by a particular utility. Since the availability of service area boundaries is a limiting factor in where

this approach can be applied, we experimented with using municipal boundaries as a proxy. At the request of stakeholders in Oregon, we used municipal boundaries as a proxy for the water service area; however, we hesitate to recommend such usage. While the boundaries might be accurate for some municipalities, many utilities have extended their service beyond municipal boundaries and/or do not serve all areas with a municipality.

Lastly, we were not able to find any publicly available rates data. The Environmental Finance Center at UNC and some states provide the estimated costs of service at certain levels of water usage. While this is certainly helpful, we thought it was important to build a database where users can see the underlying rate structures shaping those bill estimates. As such, we searched for rates data from utility websites, news articles, and bond documents to build the initial rates database. We are in the process of working with the Internet of Water project at Duke University to develop open online tools that allow utilities or interested stakeholders to update rates, enter new rates, and modify or create service area boundaries. The overall process for creating and maintaining the affordability dashboard are showing in Figure 1 and the supporting data are described in Table 1.

Table 1: Data descriptions and sources as of February 2021

Layer	Description	Source(s)
Utility Spatial Data Manual	-Service area boundaries for drinking water systems -OR (using municipal boundaries) -TX	 -OR: municipal boundary -TX boundaries
Utility Spatial Data Automated	-Service area boundaries for drinking water systems -CA -PA -NC	 -CA boundaries -PA boundaries -NC: ABOUT-US
Utility Rates Data Manual	Drinking water, wastewater, and stormwater rates for utilities. These data were manually collected from utility websites	-See rates metadata for links
Census Spatial Data Automated	-County boundaries -Tract boundaries -Block group boundaries -PA municipal boundaries -NC municipal boundaries -TX municipal boundaries	-automated from census api -PA: PASDA -NC: NC OneMap -TX: TxDOT
Census Spatial Data Manual	Municipal boundaries -CA -OR	 -CA: NHGIS places -OR: NHGIS places
Census Attribute Data Automated	Census API in R Tidyverse Tract: S1701_C01_042E S0101_C01_001E Block Group: B01001_001E B19001_001E B19013_001E B19001 group B06001 group B02001 group B03001 group B25034 group	 Poverty below 200% FPL Total Poverty Surveyed Total Population Total Households Median Income Income brackets Age brackets Race groups Ethnic groups House age brackets
Bureau of Labor Statistics Data Manual/Automated	Unemployment by county (currently manual but I think can automate)	https://www.bls.gov/lau/
EPA SDWA Data Automated	EPA data on drinking water systems	EPA API



Getting Started With Data

There are few things needed to get started.

Set up your folder structure

The easiest way to set this up is to clone the git hub folder: NIEPS-Water-Program/water-affordability: Data and code supporting the water affordability dashboard (github.com).

The scripts to access and create files are dependent on a specific file structure (Figure 2).

Figure 2: Folder structure needed for scripts to run propertly. The **affordability folder** contains all the code and data needed for the analysis. The **code folder** holds the R scripts access raw data and places those data in the **data folder**. The affordability metrics and analysis are placed in the **results** folder. The **www folder** contains the files needed to create the dashboard. The last r script transforms the data in the **results** folder into a simplified version located in the **www/data** folder of the dashboard. The **css** and **img folders** contain files that stylize the dashboard and provide static image. The **scripts folder**

contains javascript files that read in data from the **data folder** to create the charts and maps displayed in the dashboard. The www folder is optional.

Install R and a Text Editor (if needed)

Install R.

To run the files in the rcode folder that will access and update data, you will need to install R.

You can install R for Linus, Mac, or Windows here: https://cran.r-project.org/

Install RStudio (optional)

Rstudio is a really helpful platform for running R and being able to visualize code and plots together. You can install R Studio Desktop here: https://www.rstudio.com/products/rstudio/download/ for windows or mac. You can run R through RStudio.

Install a Text Editor (optional)

If you want to recreate or modify the dashboard for your own purposes then you will need a text editor.

Any text editor will do, but Visual Studio Code (https://code.visualstudio.com/). This is a free software that has some nice features for debugging and testing changes on your computer before it is pushed to a server (check out the "Go Live" feature). The editor changes font color and spacing based on the type of file (.css, .js, and .html).

Another commonly used, free, open source text editor is notepad++: https://notepad-plus-plus.org/downloads/. Notepad++ also stylizes the text based on file type and can be used to make edits to these files. Additional packages can be downloaded for debugging, etc.

Install Tippecanoe (optional for dashboard)

Tippecanoe is a way to create mapbox tilesets for large geojson files. We found this approach worked best for quickly loading mapbox. If you are on Windows 10, the following instructions will allow you to load Tippecanoe.

1. Open "Turn Windows Features On and Off" from the start menu
    a. Enable "Windows Subsystem for Linux"
2. Go to the Windows Store and install Ubuntu 18.04 LTS (or the latest version).
3. Restart your computer
4. Open Ubuntu from the start menu
    a. Install Tippecanoe using the following command lines or following Ubuntu instructions

        i. $ git clone https://github.com/mapbox/tippecanoe.git
        ii. $ cd tippecanoe
        iii. $ make -j
        iv. $ make install

Get your own api keys for generating maps and accessing data

Census API Key

To access census data through an API you will need an api key. A census API key is free and can be requested here: https://api.census.gov/data/key_signup.html.

In the code folder, open the \code\global0_set_apis_libaries.R script and add your census api key: census_api_key("yourKeyHereBetweenQuotations", install=TRUE, overwrite=TRUE);

Mapbox API Key (optional)

If you want to create a map in your dashboard then you will need to obtain a mapbox api key. Mapox api keys are free and can be created once you have an account. For more information, go here: https://docs.mapbox.com/help/getting-started/access-tokens/

Copy and paste that token into your **www/index.html** file. Search for (ctrl+F) "mapoxgl.accessToken" and paste your token behind the equal sign.

Once these steps are completed, you are ready to start accessing data for your utilities of interest!

Accessing Data

Each year, create a new version of the database using Zenodo. This way we can begin to build a historic database for water affordability. While the rates database will build over time, we may want to apply older service area boundaries reflective of what was current in a given year.

The folder structure should be:

- main folder (any name)
    - rcode (holds all R scripts described below)
    - data
        - census_time
        - rates_data (these data were manually generated)
        - sdwis
    - results

Each year, pull the most recent service area boundaries provided by states. This may require updating API links or a manual downloading data (Table 1). Ideally, shapefile boundaries will be more easily found and accessed via Geoconnex. Additionally, the rates database is currently manually being created and provided in Github. In the future, these rates data may be provided at a different location once the utility rate survey tool has been developed. There is also a 2 year lag with census data releases. So the rates and utility boundaries data reflect current conditions (although rates data and service area boundaries may not have changed for years) and the census data are from the 2019 5-year ACS survey. The overall process for accessing data is shown in Figure 3.

Running the code: loading libraries and variables used in several scripts

The first step is to run the \code\global0_set_apis_libaries.R script. The users will have to obtain their own census api key. A census API key can be requested here: https://api.census.gov/data/key_signup.html. The user can also change the folder year and the census year of interest. This will set up where data, results, and files for the dashboard are saved. Make sure the data and results folders exist in the directory. The user may also need to update the list of states and their fips codes. These variables will be carried through all access and use scripts. The script also loads all libraries required for analysis. Users will need to install any missing libraries.

The scripts are labeled in the order they need to run. There are three groups of code:

(1) The first script sets up the R environment, libraries, and variables needed to run all the other scripts.
(2) The second group of scripts are called “access” and are designed to access and build the historic database. These scripts take longer to run and only need to be used when building the initial database.
(3) The third group of scripts are called “use” scripts. These scripts use the data. They also access new data since the last time the script was run. These scripts can be run on a daily, weekly, or monthly basis to keep update the data in the dashboard.

The first time you run the scripts or each year as census, boundaries and rates data are updated. You will also need to run all scripts if you add a new state.

→global0_set_apis_libraries.R

→access1_census_data.R →access2_utility.data.R →access3_bls_data.R
→access4_sdwis_data.R

Then you will need to run the use scripts to calculate statistics and generate files for the dashboard.

→use1_calculate_rates.R →use2_calculate_afford_metrics.R
→use3_calculate_demographics.R →use4_create_files_for_dataviz.R →

The last script (use4) is optional and only needed if you want to create a dashboard.

If you only need to update the files then you run the following scripts in this order:

→global0_set_apis_libraries.R

→access2_utility.data.R → needed to create inside-outside geographic areas.
→access3_bls_data.R → needed to periodically update the monthly unemployment data.
→use1_calculate_rates.R →use2_calculate_afford_metrics.R
→use3_calculate_demographics.R →use4_create_files_for_dataviz.R

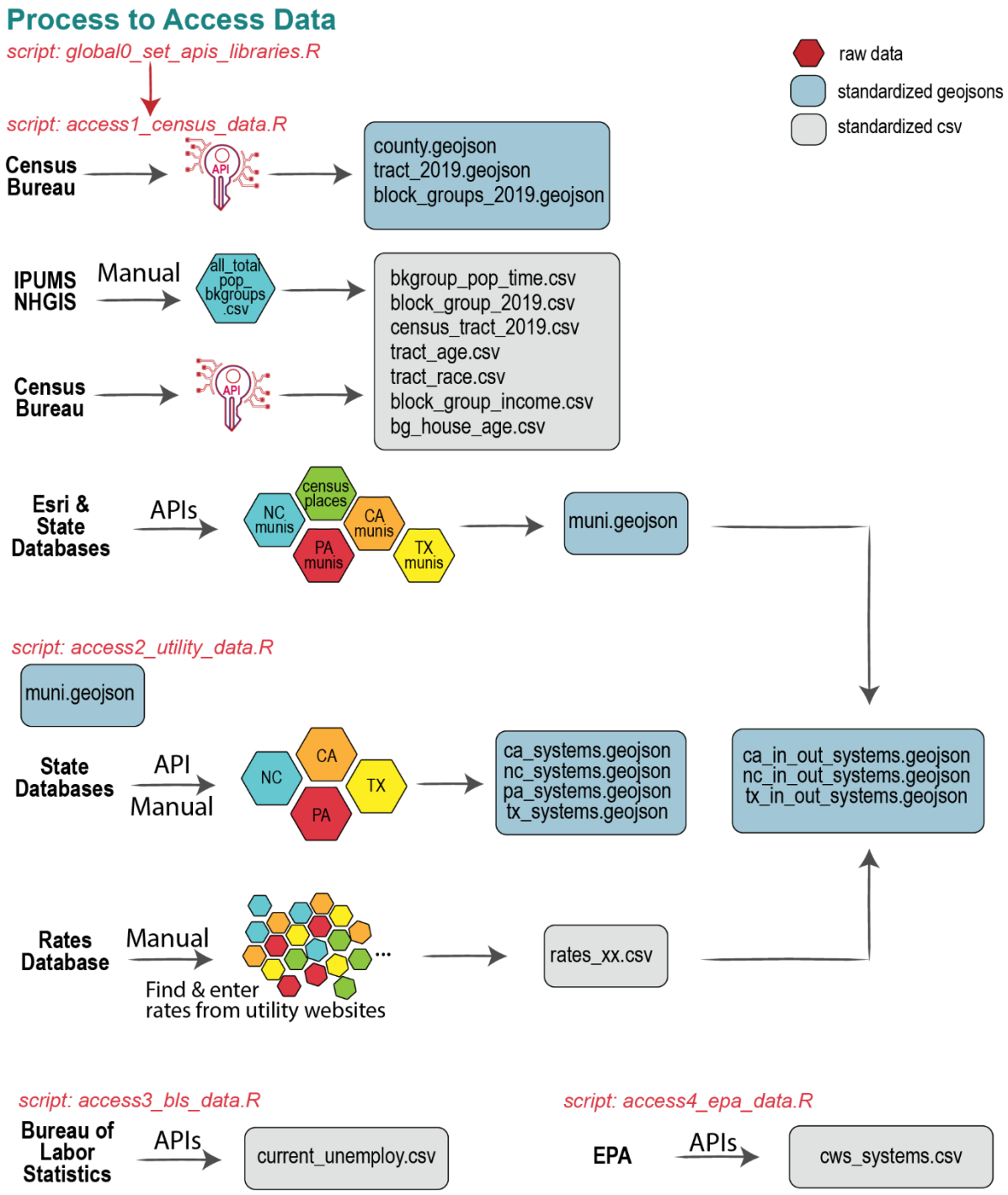


Figure 3. Flow diagram showing the process of obtaining data used to calculate affordability metrics.

0. Setting Global Variables and Directories (each time you run scripts)

In R studio, open **global0_set_apis_libraries.R**

- Add your census_api_key key to the script:
    - census_api_key("yourKeyHereBetweenQuotations", install=TRUE, overwrite=TRUE);
- Add any states or state fips to the lists. You can find state fips codes here: https://en.wikipedia.org/wiki/Federal_Information_Processing_Standard_state_code#FIPS_state_codes. Make sure these are two digit characters, such as "06" and not "6".
- Change the selected.year to match the year of census data (usually a 2 year lag from current year)
- Change the folder.year variable to save the data in the correct folder (make sure the folder exists).
- If you change the folder directory names, updated the swd_data, swd_results, and swd_html paths to match desired folder structure.

1. Census Spatial and Demographic Data (Update Frequency: Annual)

Each year update the most recent county, municipal, tract, and block group files. It is unlikely these will change much, but good to confirm and necessary when adding data from new states not already included in the dashboard. The majority of this code is automated. In terms of variables, the 5-year ACS is updated annually, and so to we will update the demographic data annually. Be sure to keep previous versions in case there is interest in observing changes over time. Additionally, you may need to update links as they change over time.

Code location: .code\access1_census_data.R

This code:

(1) Update municipal boundaries by calling data from an ESRI API.
(2) Uses the census api to read in county, tract, and block group shapefiles for selected year.
    a. This code simplifies the county boundaries since they will not be used for analysis. The tract and block group files are NOT simplified now. A simplified version will be saved for the dashboard later.
(3) Downloads most recent census data variables needed for analysis at the block group and tract scale.

Table 2: Files Needed for access1_census_data.R and the Files Created by the Script

	File Name	Description
Files needed	../data/census_time/ nhgis0033_ts_geog2010_blck_grp.csv	Block group population for 1990, 2000, and 2010. Data were obtained from IPUMS NHGIS, University of Minnesota (www.nhgis.org). All states are included in this file.
Files created	../data/county.geojson	Used to create the county map layer
	../data/muni.geojson	Used to create the municipal map layer
	../data/tract.geojson	Used to intersect tract boundaries with utility boundaries to calculate affordability metrics and demographic attributes of age, race, and enthnicity.
	../data/ block_groups.geojson	Used to intersect block group boundaries with utility boundaries to calculate affordability metrics and demographic attributes of population, income, and housing age.
	../data/or_systems.geojson	Used to create proxy utility boundaries for select municipalities in Oregon in the absence of utility service area boundaries
	../data/census_time/ census_tract.csv	Contains data on poverty survey and poverty prevalence used to calculate the poverty prevalence indicator
	../data/census_time/ block_group.csv	Contains data on population, households, and households by income bracket for calculating affordability metrics
	../data/census_time/ bkgroup_pop_time.csv	Used to create time series of population change from 1990 to present in the utility service area
	../data/census_time/ block_group_income.csv	Used to create chart of household income distributions in the utility service area.
	../data/census_time/ tract_age.csv	Used to create the chart showing population age in the utility service area
	../data/census_time/ tract_race.csv	Used to create the chart showing the racial and ethnic distribution in the utility service area.
	../data/census_time/ bg_house_age.csv	Used to create the chart showing the age of houses in the utility service area.

2. Utility Service Area Boundaries (Update Frequency: Annual)

The first step is to search state websites to see if new water service area boundaries or available (or if wastewater or stormwater boundaries become available). Update service area boundaries using the api or manual approach. Additionally, for some states that are amenable to using municipal boundaries as a proxy, we can create those boundaries (e.g. as done for Oregon). Eventually, water service area boundaries may be pulled from Geoconnex or ABOUT-US.

Code location: code\access2_utility_data.R

The code does the following:

(1) Uses apis when available to read in most recent utility boundaries. Some shapefile boundaries may need to be manually downloaded (e.g. TX) or created from municipal boundaries when those are used as a proxy (e.g. OR).
(2) Intersect and erase with municipal boundaries to create inside and outside areas for affordability calculations.
    a. This applies to CA, NC, and TX where inside and outside rates were common.
    b. To reduce the number of intersections, we load in the rates and apply the function to only those with inside and outside service areas. The rest are assigned "inside" rates to match the code for calculating affordability and rates later on.
    c. The NC shapefile is problematic and difficult to automate. Several systems break the code because of spatial challenges and require additional ms_simplify() to work or set as "inside".

Table 3: Files Needed for access2_utility_data.R and the Files Created by the Script

	File Name	Description
Files needed	../data/rates_data/ rates_xx.xlsx (where xx is the lower case state abbreviation)	Currently these are manually created spreadsheets containing information about rates. Here, the class category of "inside_outside" is used to define which water systems to intersect with municipal boundaries.
	../data/muni.geojson	The municipal boundaries created in access1_census.data.R are intersected with utility service areas to estimate the geographic location for "inside" rates and "outside" rates.
Files created	../data/xx_in_out_systems.geojson Where xx geojsons exist for ca, nc, and tx as of June 2021	The inside and outside shapefiles are used to calculate affordability metrics accounting for different rates and demographics.
	../data/xx_systems.geojson Where xx geojsons exist for ca, nc, pa, tx, and or as of June 2021.	These geojsons are used to link rates data with the demographic data to calculate affordability metrics and create all charts.

3. BLS Unemployment Data (Update Frequency: bi-monthly)

BLS is somewhat sporadic on how frequently they update their county unemployment data. As long as we are interested in tracking COVID-19 impacts on unemployment, this should be updated bi-monthly. Once the acute phase has passed, we can change to updating annually and remove one of the graphics from the dashboard.

Code location: code\access3_bls_data.R

The code does the following:

(1) Download BLS county level unemployment data used to show how unemployment has changed over time and in response to the COVID-19 pandemic.

Table 4: Files Needed for access3_bls_data.R and the Files Created by the Script

	File Name	Description
Files needed	../data/census_time/ current_unemploy.csv	The BLS provides 14 months of county level data at a time. We began creating this file early in the pandemic (2020) and continually add new data.
Files created	../data/census_time/ current_unemploy.csv	We add the new data and save over the file. This file is used to create the charts showing unemployment around utility service areas over time.

4. EPA SDWIS Violations Data (Update Frequency: as needed)

EPA updates the Safe Drinking Water Information System (SDWIS) dataset daily and users can update the data as frequently as desired. More frequent updates may be desired if incorporating violations data.

Code location: code\access4_epa_data.R

The code does the following:

(1) Uses EPA's API to download EPA SDWIS data as needed (at least once a year) to obtain characteristics about water systems such as population served and ownership.

Table 5: Files Needed for access4_epa_data.R and the Files Created by the Script

	File Name	Description
Files needed	Not dependent on any files	
Files created	../data/sdwis/ cws_systems.csv	The SDWIS data is used to provide attributes such as population served (system size) and ownership type that are used to filter systems in the dashboard.

Summary of Scripts to Access Data

Table 6 summarizes the steps needed to access data. Most of this code is run annually or when new states or systems are added to the rates database.

Table 6: Code to access data used for analysis or to create the dashboard.

Data	Code Name	Automation	Time to Access	Update Frequency
Global variables	global0_set_apis_libraries.R	50% automated	1 minute	As needed
Census data	access1_census_data	95% automated	30 minutes	Annually
Utility service areas	access2_utility_data	95% automated	1 hour+	Annually
Unemployment data (BLS)	access3_bls_data	100% automated	5 minutes	Monthly or Bi-monthly
SDWIS data (EPA)	access4_epa_data	100% automated	30 minutes	As needed
Utility rates data	TBD	100% automated	5 minutes	As needed

Using the Data

Affordability metrics can now be calculated once data have been downloaded and put into a standardized format. The following use codes must be run in the correct order after running the script: global0_set_apis_libraries.R. Figure 4 shows the process for calculating affordability metrics.

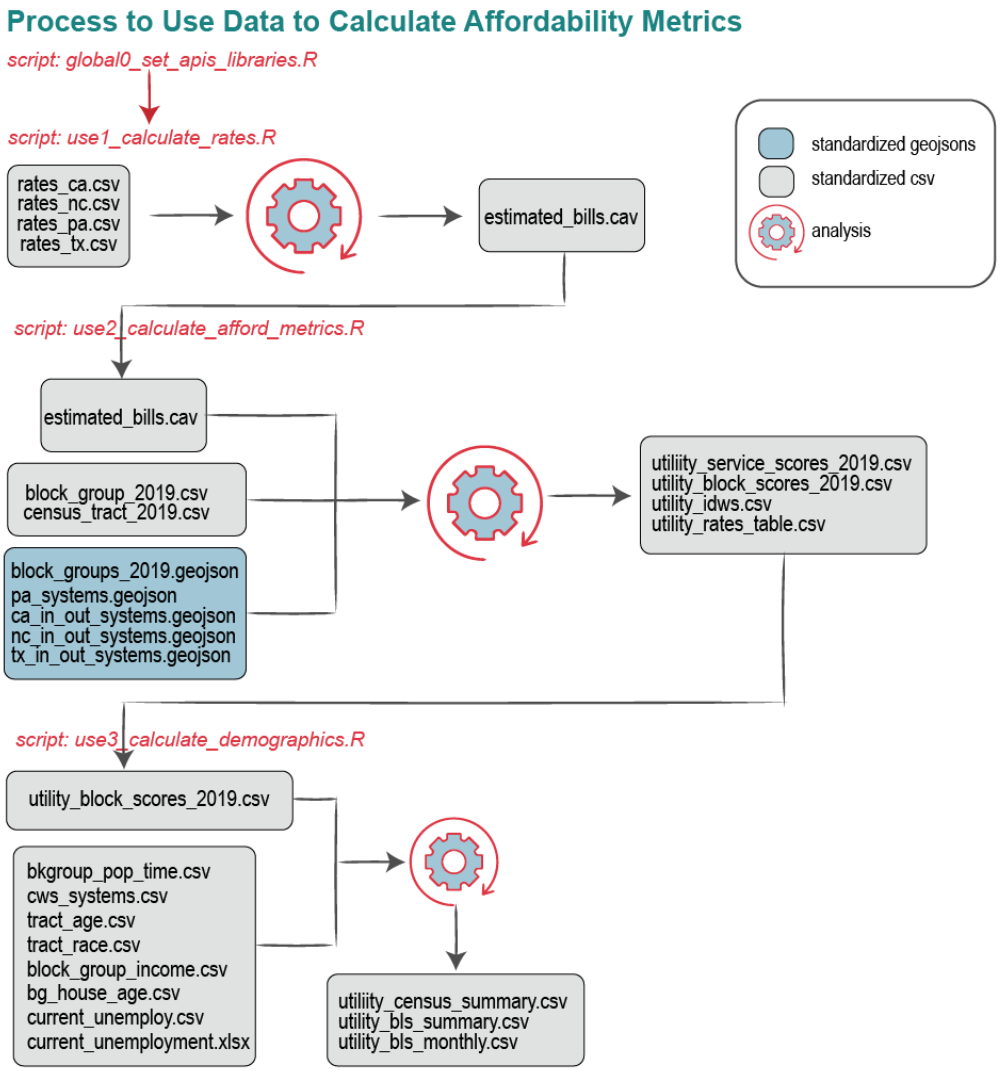


Figure 4. Flow diagram showing the process of combining data to calculate affordability metrics.

1. Calculating Rates for Each Service Area

The first step is to estimate the monthly and annual bill for customers in a utility by summing the service charge, volume charge, zone charge, and surcharges for each system. As the number of utilities increases, the time to run this code increases because we are estimating multiple bills for multiple utilities at multiple volumes. As such, we recommend users to only update the bills for those with new

rates. This can be done by adding a column to the rates data with the header "update_bill" and set the value to "yes" or "no". The code will only calculate new bills for "yes" and append them to the pre-existing database. The full code should probably be run annually.

The script calculates the mean bill when a utility charges different rates within the service area. The one exception are inside and outside rates for each type of water service (drinking water, wastewater, and stormwater). We tally the number of water services provided in the service area because we are only including those utilities with both drinking water and wastewater services for comparing affordability metrics. The "total" service consists of water and sewer only (not stormwater) and is a remnant of an earlier version of the dashboard.

Code location: code\use1_calculate_rates.R

The code does the following:

(1) Pulls all state rates data and estimates the drinking water, wastewater, and stormwater monthly bills for 0 to 16,000 gallons of use at 1,000 gallon increments.

Table 7: Files Needed for use1_calculate_rates.R and the Files Created by the Script

	File Name	Description
Files needed	../data/rates_data/ rates_xx.xlsx (where xx is the lower case state abbreviation)	Used to estimates the average monthly bill for drinking water, wastewater, and stormwater services within the service area of the drinking water utility.
Files created	../results/estimated_bills.csv	Estimates the bill at 1,000 gallon increments from 0 to 16,000 gallons. These data will be used to calculate affordability metrics in use2_calculate_afford_metrics.R.

2. Calculating affordability metrics for each service area

This script calculates several affordability metrics for each estimated bill and takes a long time to run as the number of utilities increase. We recommend running once a year with new census data and then updating or adding new utilities individually during the course of the year (code is set up to do that). This code attaches rates data to water systems using the pwsid and spatially intersects the water system with census block groups. The affordability metrics calculated are the traditional metric using median household income, the household burden using the 20th percentile income, the poverty prevalence, and the minimum wage hours. We calculate the minimum wage hours for inside and outside rate and use the average at the utility scale.

Code location: code\use2_calculate_afford_metrics.R

The code does the following:

(1) Calculates affordability metrics based on census data, estimated bills, and utility service areas.
   - utility_rates_table.csv

Table 8: Files Needed for use2_calculate_afford_metrics.R and the Files Created by the Script

	File Name	Description
Files needed	../data/block_groups.geojson	Used to spatially intersect block group with utility service area boundaries to identify the block groups in the service area and link to the relevant income and household data.
	../data/census/time census_tract.csv	Contains the census poverty data for affordability metrics (obtained in access1_census_data.R).
	../data/census/time block_group.csv	Contains the demographicdata needed to calculate affordability metrics (obtained in access1_census_data.R).
	../data/xx_systems.geojson Where xx geojsons exist for ca, nc, pa, tx, and or as of June 2021.	These geojsons are used to link rates data with demographic data to calculate affordability metrics and create all charts (obtained in access2_utility_data.R).
	Manually update the minimum wages for each state to reflect the year of interest (selected.year or folder.year)	Used to calcualte the Minimum Wage Hours affordability metric.
Files created	../results/ utility_service_scores_xxxx.csv where xxxx is the selected.year (census year)	Contains all estimated affordability metrics for each utility service area.
	../results/ utility_block_scores_xxxx.csv where xxxx is the selected.year (census year)	Contains all census based estimated affordability metrics for individual blocks within ther service area.
	../results/utility_idws.csv	Contains the estimated percent of households in each utility spending between 1 to 20% of their income on water services (broken into inside and outside estimates where relevant).
	../results/utility_rates_table.csv	Reshapes the estimated_bill.csv and keeps those with sufficient data to calculate affordability metrics.

3. Compiling demographic data for each service area

This code pulls the block groups and census tracts with water service areas together to create the demographic charts in the data visualization. Data are summarized at the block group, tract and county scale. None of these data are weighted by percent of overlap with service area, but represents general demographics within and surrounding the utility.

- Creates tables for census tab

- Population over time (block group)
- Unemployment (county)
- Age (tract)
- Race (tract)
- Income distribution (block group)
- Household age (block group)

Code location: code\use3_calculate_demographics.R

The code summarizes demographic characteristics within and near the utility service area.

Table 9: Files Needed for use3_calculate_demographics.R and the Files Created by the Script

	File Name	Description
Files needed	../data/census_time/ bkgroup_pop_time.csv	Time series of population change from 1990 to present (see access1_census_data.R).
	../data/census_time/ block_group_income.csv	Number of households by income bracket (see access1_census_data.R).
	../data/census_time/ tract_age.csv	Number of people in age brackets (see access1_census_data.R).
	../data/census_time/ tract_race.csv	Number of people in race and ethnic census categories (see access1_census_data.R).
	../data/census_time/ bg_house_age.csv	Number of households by decade (see access1_census_data.R).
	../data/census_time/ county_unemployment.xlsx	Annual unemployment by county from 1990 to 2019 as downloaded from the BLS.
	../data/census_time/ current_unemploy.csv	Monthly unemployment by county from 2020 to present (see access3_bls_data.R).
	../results/ utility_block_scores_xxxx.csv where xxxx is the selected.year	Contains the percent area each block group overlaps with the utility. This is used to weight the demographic data in the service area (see use2_calculate_afford_metrics.R.).
	../results/sdwis/cws_systems.csv	The population estimate is used to adjust the estimate obtained using census data. This adjustment was particularly important for geographically small systems (see access4_sdwis_data.R).
Files created	../results/ utility_census_summary.csv	Contains summaries of census data (population, age, race, income, and housing age) used to create demographic charts in the dashboard.
	../results/utility_bls_summary.csv	Contains the annual unemployment rate for counties intersecting the utility service area.
	../results/utility_bls_monthly.csv	Contains the monthly unemployment rate for counties intersecting the utility service area from 2020 onward.

4. Simplifying and saving files for a data visualization

This code simplifies geojson and csv files for the data visualization. This is how we simplified our code for the dashboard, but you may simplify and combine data in whatever fashion best suites your need. The metadata_html.xlsx provides detailed information about what each column does and its role in the dashboard. We simplified the results file to only include those data used in the dashboard.

Code location: code\use4_create_files_for_dataviz.R

Once the use4_create_files_for_dataviz.R script has run we need to create the mapbox tiles.

To create the mapbox files do the following:

1. Open an Ubuntu terminal and navigate to the directory containing your desired GeoJSON files. This can be done using the cd or 'change directory' command. Begin by typing **cd ../../mnt/c** to access Ubuntu's mounted C:// drive. Then, use the cd command to navigate to the data directory (i.e. **cd [www]/data/mapbox_sources**). You'll know you're in the right directory if typing the command ls lists your .geojson files.
2. Run the tippecanoe.sh shell script by typing **./tippecanoe.sh** into the command line. Allow the script to run - this process may take up to 30 minutes. If this script ever needs to be changed, simply open it in a text editor and change the commands listed.
3. If the script in the last step executed correctly, you should have new .mbtiles files in your directory, named after the .geojson sources. These files need to be uploaded to mapbox's tile service - Find the tilesets with a matching name in the tiling service website (https://studio.mapbox.com/tilesets/) and click its title. You will only see tilesets associated with your mapbox api token. From here, click the 'replace' button and drag the matching .mbtiles file into the upload box. The tiles should automatically upload and update for the clients.

To get a working map you will need to have created your own mapbox api token. In the index.html file search for 'mapboxgl.accessToken' = 'YOUR API KEY HERE BETWEEN QUOTATIONS'. Once that is done, you should be able to see your geojson file on the server you are running.

The easiest way to test this is to use Visual Studio Code Editor and click the 'Go Live' button in the bottom bar.

We created our dashboard using html, css, and javascript (Figure 5). We have outlined in the diagram how each of the javascript functions relate to each other.

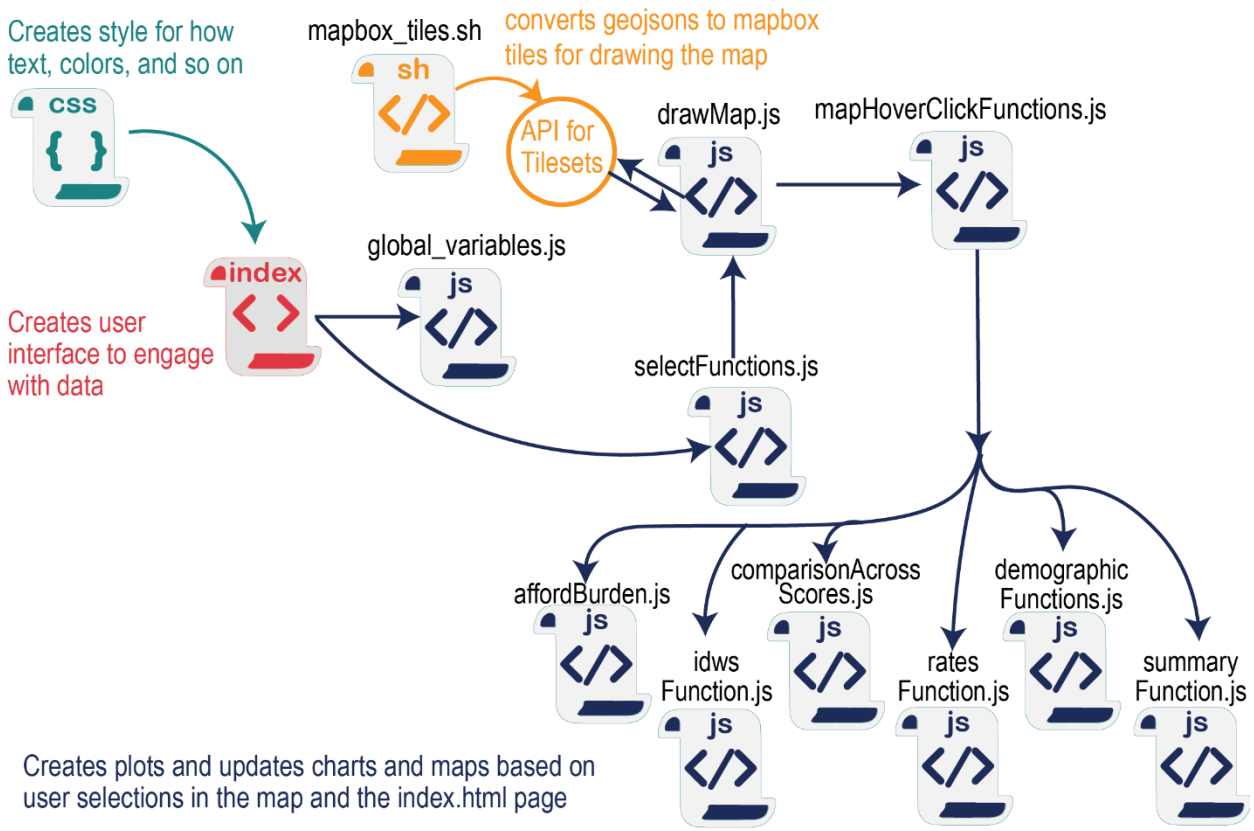


Figure 5. File structure for creating the affordability dashboard.

CSS File

..//css//main_content.css

This file contains classes for divs and stylizes how many of the elements render.

HTML Index File

The head of the file calls scripts used to:

- Holds some global variables used in several functions
- create the map (mapbox)
    - loads the initial map with geocode, county layers, municipal layer, state layer, legends, and the initial utility layer at 4,000 gallons.
- create the dashboard elements and format (bootstrap)
- create the fonts (gstatic and googeapis)
- load javascript files that contain different functions

The body sets the placement of divs and global variables used in javascript.

Contains code to draw original map and establish the geocoder.

Javascript Files

There are a series of javascript files responsible for creating the plots and maps.

(1) global_variables.js
   a. Contains the remaining global variables
   b. Loads the super simplified geojson file of utility boundaries for zooming into the map when one is selected and saves it into the global variable *gisSystemData*.
   c. Loads the csv with utility metric scores that change with volume (hh_use) and saves it into the global variable *utilityScores*
   d. Loads the csv with static utility characteristics and saves it into the global variable *utilityDetails*. This also calls the function **setDropdownValues()** located in *drawMap.js*
   e. Loads the demographic data and saves it into the global variable *demCSV* and *blsData* for the unemployment data. It then calls the **plotDemographics(selectSystem, selectVolume)** function after a timeout to allow the files to load.

(2) selectFunctions.js
   a. contains functions for the filter panel by the map. Each time a dropdown menu is engaged it calls the **drawMap()** function in the drawMap.js file.
   b. **setStateThis()** function fits the map to the state boundary and calls **drawMap()** function
      i. Returns *selectState* variable
   c. **setSizeThis()** function sets the size of the system calls the **drawMap()** function
      i. Returns *selectSize* variable
   d. **setOwnerThis()** function sets the system owner and calls the **drawMap()** function
      i. Returns *selectOwner* variable
   e. **updateNVol()** function returns the volume selected on the slider, creates the text about the current volume, removes any loaded block groups or utility layers in the map, and calls the **drawMap()** function
      i. Returns *selectVolume* variable
   f. **setSystemThis()** function returns the selected system and calls the **highlightUtility()** function
      i. returns *selectSystem* variable
   g. **setMatrixLegendThis()** function sets the legend burden chart and calls the **createMatrix()** function in the affordBurden.js file
      i. returns *matrixLegend* variable

(3) drawMap.js

a. Contains the $('button') function that turns the county and municipality layer on and off in the map.
b. **setDropdownValues()** function sets the dropdown list menu based on the *UtilityDetails* global variable and filtered by *selectState*, *selectSize*, and *selectOwner* variables. This also creates the text in the gray rectangle below the map to remind users of their selections. In the process of doing this it creates the *selCSV* global variable called in the functions that create charts.
    i. Returns *selCSV*
c. **drawMap()** function
    i. calls the setDropdownValues() function
    ii. removes the mapbox utility layers and redraws the map based on user selection: *selectState*, *selectSize*, and *selectOwner*
    iii. It loads the charts for whatever tab is open
d. Calls the **highlightUtility(selectSystem)** function in the mapHoverClickFunctions.js file

(4) mapHoverClickFunctions.js
a. Contains the functions for hovering and clicking on the utility and block group layers.
    i. Includes info boxes when the mouse moves over a utility or block groups
    ii. Includes creating highlight layers for when scrolling over block groups and makes the block group layer unclickable
    iii. Includes creating the histogram when the mouse moves over a utility
b. Contains the **highlightUtility()** function that runs whenever a selection is made:
    i. Removes any previously selected utility or block group layers
    ii. Draws block groups for the selected system (if selected) and adds it to the map
    iii. Creates text stating what system is selected with it's pwsid (or that no system has been selected)
    iv. Calls other functions to plot charts depending on what chart tab is currently opene

(5) affordBurden.js
a. **createMatrix(selectSystem, matrixLegend)** function where *selectSystem* is the utility selected by the user and the *matrixLegend* is the legend type selected by the user. The *matrixLegend* variable is the selectFunctions.js (**setMatrixLegendThis()**). This function uses the data preloaded in **global_variables.js** (*utilityScores* and *selCSV* – which is the filtered *utilityDetails* variable).
    i. Creates the affordability burden plot showing utility scores for both poverty prevalence and household burden
        1. returns plot to the div with id: matrixPlot

    ii. Creates the table counting the number of utilities, filtered utilities, and block groups for the selected utility that are in each affordability burden category.
        1. returns table to the div with id: matrixTable

iii. Creates the burden plot showing the change in household burden for the selected utility at all volumes of water.
    1. returns plot to div matrixVolPlot
iv. The function also updates the headers of plots and tables.

(6) comparisonAcrossScores.js
    a. contains the **allScores(selectSystem)** function where *selectSystem* is the utility selected by the user. This function uses the data preloaded in **global_variables.js** (*utilityScores* and *selCSV* – which is the filtered *utilityDetails* variable).
    b. Creates the box plots comparing minimum wage hours, traditional, household burden, and poverty prevalence metrics.
        i. returns the traditional and household burden boxplots to the allScoresChart div
        ii. returns the poverty prevalence boxplots to the ppiScoresChart div
        iii. returns the minimum wage boxplots to the mwhScoresChart div

(7) ratesFunction.js
    a. contains all the functions used to create plots and tables in the tab about the cost of water services.
    b. **createBoxTrace()** function that reads in whether drinking water or wastewater were selected by the user
        i. returns *selWaterType* which refers to whether the user wants to explore drinking water or wastewater rates

    c. **plotRates(selectSystem, selectVolume, selWaterType)** functions where *selectSystem* is the utility selected by the user, *selectVolume* is the volume selected by the user, and *selWaterType* is whether the user prefers to see drinking water or wastewater data. The data included in the plots are filtered based on the *selCSV* variable
        i. This function reads in several files:
            1. data/rates_metadata.csv to create the table of utilities and links to their rates online in the ratesMetaTable div.
            2. data/rates/all_rates_"+selectVolume+".csv to create the plots showing the rates and rate structures for the selected utility at this volume.
                a. The csv was too large to have all volumes in one place (slow load speeds) and we divided the csv's by volume
                b. returns plot of monthly bill to the monthBillChart div
                c. returns the boxplot showing the rate structure to the boxRateChart div
            3. data/commodity_price.csv to create the table showing the cost for 1000 gallons of water in the commodityChart div

(8) idwsFunction.js
    a. **togglePlots()** function shows the line chart or a table summarizing similar information depending on user selection

i. returns *selectPlotType* (either “table” or “plot”)

b. contains the function **plotCostBill(selectSystem, selectVolume, selectPlotType)** where *selectSystem* is the utility selected by the user, *selectVolume* is the volume selected by the user, and *selPlotType* is whether the user prefers to see a plot or a chart showing how many households spend more than some percent of their income on water services. The data included in the plots are filtered based on the *selCSV* variable.

c. reads in the following csv file: “data/IDWS/idws_” + selectVolume +”.csv” to create plots and tables showing how many households spend a certain percent of their income at the selected volume and utility.

   i. The csv was too large to have all volumes in one place and so the csv is volume specific.

   ii. Returns the plot showing the percent of households spending more than some amount of income on water services

      1. returns the plot to the costBillChart div

   iii. If the table is selected, we also read in the “data/all_utility_census_summary.csv” file to assess how many households are in each bracket. We then calculate the percent of income going to water services in that bracket.

      1. Table to the costBillChart div

(9) demographicFunctions.js

a. **plotDemographics(selectSystem)** function where *selectSystem* is the utility selected by the user. This function creates all the plots and chart titles in the demographics tab. The data are filtered based on the *selCSV* variable.

b. The census data are loaded in the global_variables.js file and saved in the *demData* variable

   i. returns the chart showing population change over time to the popTimeChart div

   ii. returns the chart showing age distribution to the ageChart div

   iii. returns the chart showing race and ethnic distribution to the raceChart div

   iv. returns the chart showing the income distribution to the incomeChart div

   v. returns the chart showing the housing age distribution to the buildChart div

c. The unemployment data are loaded in the global_variables.js and are saved as *blsData*

   i. returns the plot showing the annual unemployment rate to the annualBLSChart

      1. We show every other year to increase load speeds

d. Reads in the file “data/all_utility_bls_monthly.csv”

   i. Returns the chart showing monthly unemployment from 2020 onward to monthBLSChart. This chart only loads if data previously selected.

e. Note that very small and small utilities may not be displayed here. These utilities are often much, much smaller than a single block group and our confidence in the data is lower. An analysis at the block level or a survey of the utility (when a single neighborhood) would be more appropriate.

(10)summaryFunctions.js

a. contains all the scripts to create the summary tab in the function **createSummary()** where selectSystem is the selected utility. The function uses data preloaded in the global_variables.js: *utilityScores* and *selCSV*.
    i. Returns plot showing the percent of utilities based on their affordability burden level for each volume of water from 0 gallons to 16,000 gallons
        1. Returns plot to summaryBurdenChart div
    ii. Returns plot showing the percent of utilities at various levels of household burden for each volume of water from 0 gallons to 16,000 gallons
        1. Returns plot to summaryHBIChart
    iii. Returns plot showing the percent of utilities at various levels of poverty prevalence for each volume of water from 0 gallons to 16,000 gallons. Note that the same poverty is prevalent no matter how much water is used. That is the point to make here.
        1. Returns plot to summaryPPIChart
    iv. Returns plot showing the number of hours minimum wage earners would spend each year to pay for water services for each volume of water from 0 gallons to 16,000 gallons
        1. Returns plot to summaryWageChart
    v. Returns plot showing the percent of utilities with percent of income going to water services for the traditional metric for each volume of water from 0 gallons to 16,000 gallons
        1. Returns plot to summaryMHIChart