# Overview

This app produces customizable visualizations of ACS statistics by CCA. The app can produce maps, bar plots, and tables of most ACS variables[[1]](#footnote-1) by CCA. To do this, it takes one of two types of inputs: pre-loaded and aggregated data from a file, or ACS data downloaded in the app via a census API.

Table 1: Inputs and outputs

|  |  |  |
| --- | --- | --- |
| Inputs | 🡪 | Outputs |
| Pre-loaded and aggregated data found in CCA Statistics.csv | Map |
| Bar plot |
| Raw data downloaded from a census API and aggregated in the app |
| Table |

# User Interface

The user selects the data to be represented on the left-hand pane. The graphic can be customized in the right-hand pane. The graphic is presented in the middle pane.

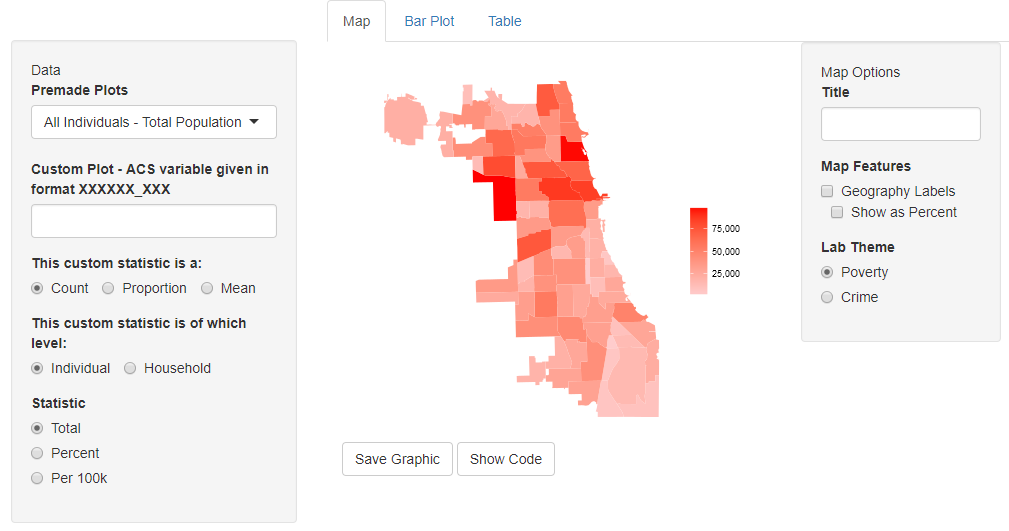


Figure 1: Example of a map

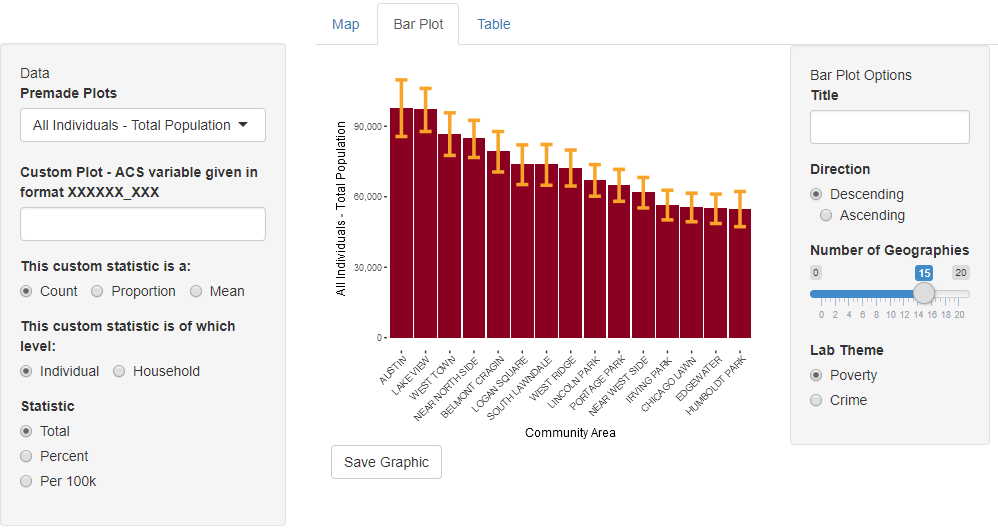


Figure 2: Example of a bar plot

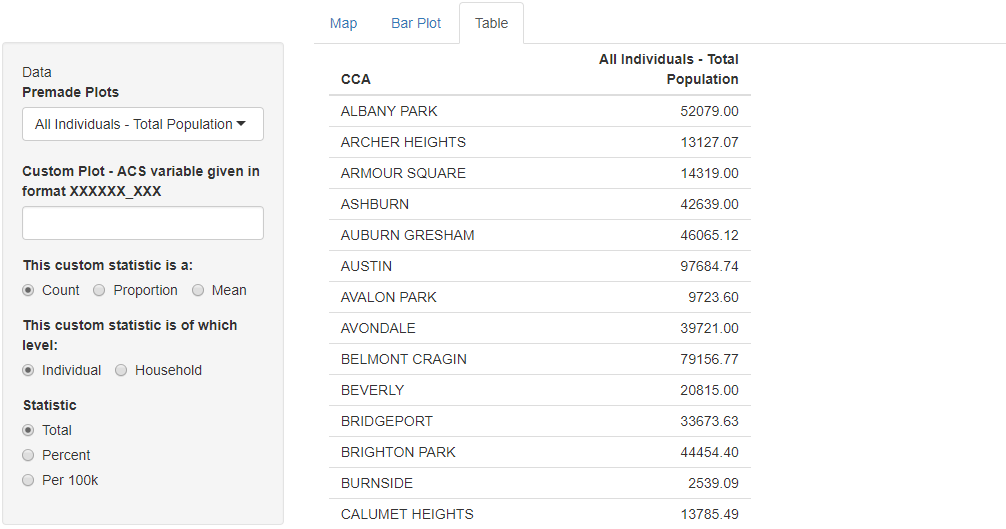


Figure 3: Example of a table

# Code

The app is organized according to the table 1, operating on two “tracks” depending on the input given. When a user selects a premade plot (the top left-most menu in the UI example above), the app simply draws a graphic of this variable from a preexisting dataset that includes the CCA and the statistic. When a user inputs an ACS variable to produce a custom plot from (the text entry box in the upper-left), the app downloads that data via a census API, aggregates it to CCA, and then draws a graphic.

Ultimately, the following files are loaded by the app:

*For premade plots*

* *CCA Statistics.csv*: Preselected statistics by CCA.
* *CCA Statistics Fortified.RData*: The above statistics in a format that can be read by ggplot2.
* *Preselected Variables.csv:* Only the names of the above variables, for labeling purposes.

*For custom plots*

* *2\_Aggregation Function.R:* The function that aggregates tract-level data downloaded via the census API to CCA level. This code uses the file *Tract to CCA Aggregation Lookup.csv* to do this.

*Miscellaneous*

* *Themes.R:* Includes ggplot themes for maps and barplots.

Full workflow below:

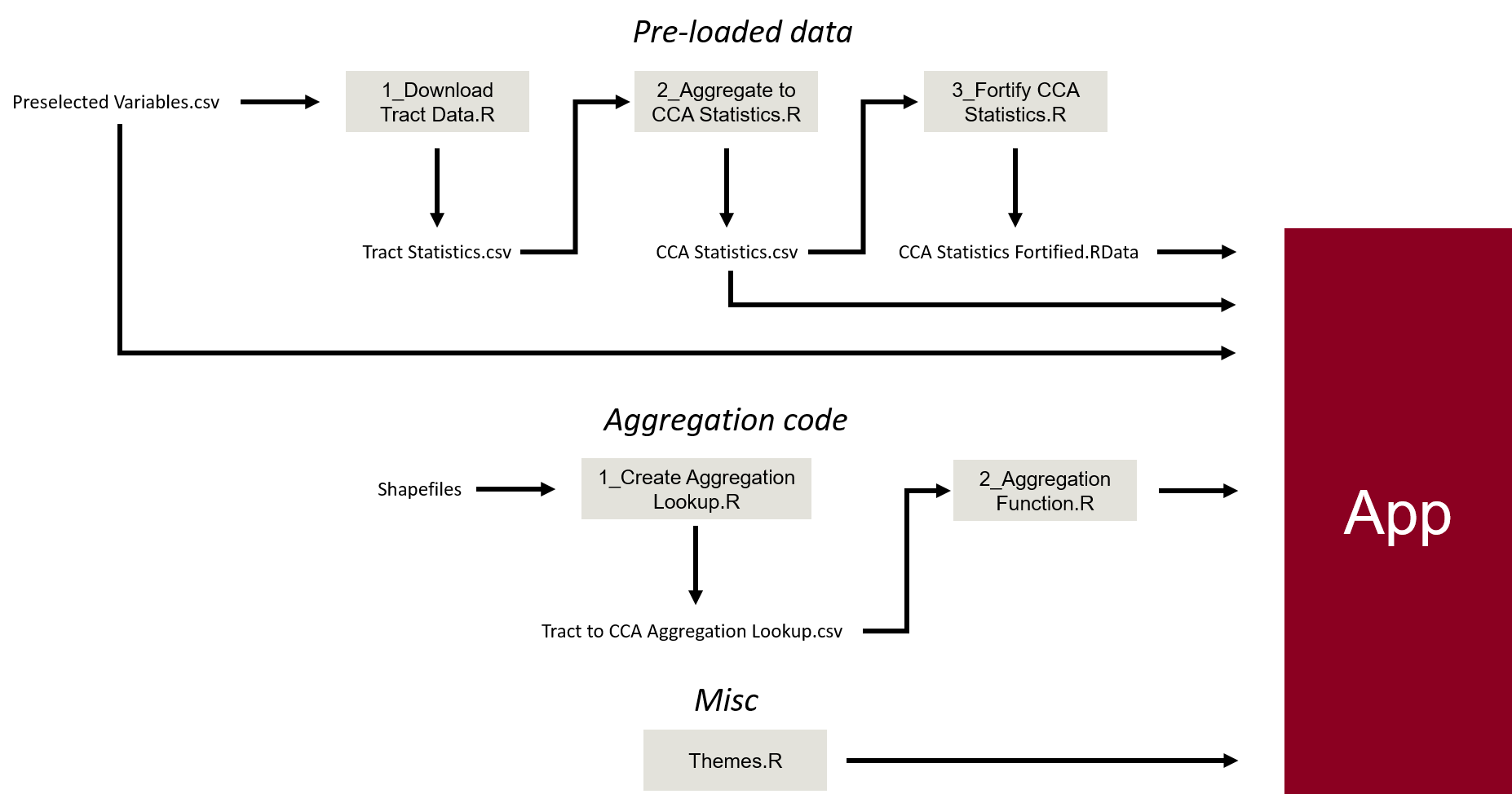


Figure 4: Code workflow

# Current Limitations

1. The census API for downloading ACS data, the R package acs, cannot load variables unless they conform to a specific 9-character format. Not all variables conform to this format, even if they are available at the tract level.
2. Margins of error cannot be aggregated accurately without individual-level data. We will have to use an approximation, but have not come up with one yet.
3. Cannot transform variables in current version
4. Various incomplete functionalities (see below)

# To Do before Launch

Required:

1. Get this thing on gitlab: **1 hour, Isaac**
2. Host online (?): **3-4 hours**
3. ~~Write a one-page readme on use:~~ **~~1 hour, Isaac~~**
4. Write a supplemental one- or two-pager explaining our aggregation method and link to it in the app: **1-2 hours, Isaac**
5. Address margin of error calculation: **5-10 hours, statistical and programming expertise**
6. Add ability to transform statistics from a total to a percent/per-capita number: **5-10 hours, R and ACS expertise**

Wish-list:

1. Improve plot customization options, functionality (titles, axis labels, showing statistic as a percent, etc.): **10 hours, R Shiny expertise**
2. Add a colorbrewer pallet for each lab: **2-3 hours, R expertise**
3. Fix save button (otherwise plots can be saved by right-clicking): **1 hour, R expertise**
4. Clean UI text, warnings: **2-3 hours, R Shiny expertise**
5. Improve and validate the Show Code functionality: **2-3 hours, R expertise**

1. See Limitation 1 [↑](#footnote-ref-1)