# Group 5: Food Deserts in the United States of America

Cole Barcia  
Marie Deshommes  
Melissa Medina  
Mike Lubin  
Pablo Barbatto

## Theme:

Food Deserts in the United States. Food Deserts refer to a phenomenon where, due to several factors, there is a lack of immediately available fresh food sources to communities in this country. By our definition a food desert can be defined as a geographic area where a person must travel more than half a mile to reach a supermarket in an urban area or more than ten miles in a rural area. This limited access can be directly correlated to the median income of the community. To put it succinctly: poorer communities have greater difficulties accessing sources for fresh food.

## Coding Approach:

Initially we intended to use Python to read our CSVs and then load them into MySQL. Once loaded into MySQL we would then use Flask API to call upon the database and merge the results to display in a JSONified format. Following a successful API call we could then use the JSONified results to load into JavaScript to generate mapped images and charts which could then be displayed in a website for users to interact with.

## Data Munging:

Starting with the Food Access Research Atlas (FARA) provided by the Economic Research Service through the United States Department of Agriculture we downloaded their dataset as a CSV file. Given that the FARA dataset used geocoding identifiers specific to the Census Tracts used un the US Census we had to tract down another sources to translate US Census Tracts to latitudes and longitudes that would allow us to map the data in a meaningful way. The file we found was in TXT formats which we imported into CSV to read into our Python file.

Using Pandas through a Python file we loaded the CSVs into dataframes. Once we had the dataframes we merged the two dataframes with a left join in Python for one, easy to use, combined dataframe. This final dataframe was more than 73k rows and the size was causing us difficulty when we tried to call on the entire data through our RESTful API so we were forced to revise the process and devise a way to break up our data.

Given that the US Census also groups their geolocations by region we decided to use these previously defined regions to break the API calls into discrete pages. We created four variables (Northeast, Midwest, South, West) that consisted of lists of the states that fell into each region. At this time we decided to exclude data from Alaska and Hawaii and focus on the lower 48 states, this had the benefit of shrinking the dataset by a modest amount and providing a more concise visual space to map.

Using .LOC we instructed the data to look in the variables, match the state in the combined dataframe, and return the region name where it was found. This result was dropped into a column called Region that we added to the combined dataframe. In this way we were able to create five distinct pages (four regions plus a value for Alaska and Hawaii combined) to call through our API in smaller amounts that would load much quicker.

Finally, we loaded this combined dataframe as a database into MySQL and moved onto the Flask RESTful API.

In our app.py file we loaded a connection to MySQL to call upon our database and created an API. The homepage returns a render\_template of our index.html file and the API returns results in JSON. The API call takes the value specified in the URL call and tells MySQL to return the data where the Region column value matches the region value used in the URL.

This data is represented on the index page in the form of a street map base with heatmap and markercluster overlays that the user can select or deselect via the layers icon in the top right corner of the map. When zoomed in to a sufficient degree for the individual map points to become visible the user can, with the markercluster overlay enabled, view tooltip data that gives relevant information about that point.

In order to demonstrate a different method of interacting with our dataset we used pandas to trim down a copy of the database to the most relevant columns. In Pandas we grouped this data by State and then by County (this allowed us to avoided combining data for the wrong states based on county name; for example matching Orange County, California with Orange County, Florida). This dataset was exported as a CSV and then transformed into a JSON file using an online web source.

We could then call upon the jsonified data directly in JavaScript files to create the charts.

## Final Visualization:

To display our project we decided that we discuss our coding approach and house the data in a HTML file to create a website with an infinite scroll that would walk through each of our maps and charts with captions. Additionally, we’d have an extra page from a floating header that would take a user to a dashboard page with al of our visualized information in an easy access format. When we attempted to implement this we found that layering maps that were making the same API call on the same page returned major errors. Due to the time constraints around the project we were forced to show our data in a different way, mainly by including more layers on the default map and relegating our charts to a separate page.