

1 Introduction

In partnership with Feeding San Diego and the Jacobs & Cushman San Diego Food Bank, the Data Science Alliance (DSA) developed a model to predict food assistance in San Diego County. The project aims to optimize food distribution efforts and promote data-driven decision-making. The guiding principles of Responsible Data Science — Fairness, Transparency, Privacy and Veracity — were applied at each stage of the project to ensure the ethical use of data.

In addition to the predictive model, DSA created a dashboard to provide a visual representation of the Food Banks' data to monitor "hunger hot spots" and other metrics. This allows the Food Banks to quickly assess the current status of a particular area to better serve the population in need.

The dashboard features three tabs. The first tab displays both historical and forecast data on food distribution, covering various timeframes and geographic levels, as well as the average number of meals distributed to a person experiencing food insecurity in a zip code. The second tab presents socioeconomic information at both county and zip code levels, providing valuable context for food banks to make informed decisions. The third tab includes information about the data used in the dashboard as well as an explanation of the methodologies used to produce the forecasts and the food insecurity estimation.

2 Problem Statement

The primary challenge addressed by DSA, in collaboration with Feeding San Diego and the Jacobs & Cushman San Diego Food Bank, is the optimization of food distribution efforts across San Diego County. The goal is to ensure efficient allocation of resources to areas experiencing food insecurity. This involves accurately predicting food assistance distribution and effectively visualizing the data to identify "hunger hot spots" and other critical metrics. The project aims to facilitate data-driven decision-making in addressing food insecurity, while adhering to the principles of Responsible Data Science: Fairness, Transparency, Privacy, and Veracity. The development of a predictive model and an insightful dashboard seeks to enhance the Food Banks' ability to respond proactively to the needs of the community, ensuring that those in need receive timely and adequate food assistance.

3 Scope of the Project

The scope of this project is defined by a series of targeted efforts aimed at addressing food insecurity in San Diego County. At its core, the project focuses on predictive modeling, identification of critical need areas, interactive data visualization, and stakeholder collaboration.

This section concisely presents these efforts, highlighting the technical approaches and the commitment to continuous improvement that underpin the project's objectives.

3.1 Development of a Predictive Model

The foundation of this initiative is the development of a predictive model tailored to forecast food distribution assistance across San Diego County. This model uses historical food bank distribution data as well as economic factors such as unemployment and inflation rate, providing forecasts for various timeframes and geographic levels. This feature is vital for identifying areas with increasing need.

3.2 Definition of Hunger Hot Spots

A critical component of our approach is the identification and definition of “hunger hot spots” within the county. Through the establishment of specific criteria and indicators, these hot spots are clearly delineated and regularly updated on the dashboard, ensuring focused monitoring and response to these critical areas.

3.3 Creation of a Dynamic Dashboard

Our project encompasses the creation of an interactive, user-friendly dashboard. This platform features three tabs: one displaying historical and forecasted food distribution data as well as “hunger hot spots”, another presenting socioeconomic data at the county and zip code levels, and a third offering details of the data used and methodologies behind our models.

3.4 Technical Details

All data collection, cleaning, and processing tasks are executed using Python. The models are developed using both Python and R, leveraging the strengths of each programming language for optimal results. All data is securely stored in a private GitHub repository and in DSA's Google Drive account, ensuring organized collaboration and data integrity. The dashboard was developed using Tableau and is hosted on DSA's website, protected by password access, ensuring data security and controlled accessibility.

3.5 Stakeholder Engagement

This project thrives on collaboration with Feeding San Diego and the Jacobs & Cushman San Diego Food Bank. We are committed to providing comprehensive support to these organizations, enabling them to effectively use the dashboard and interpret the model outputs.

3.6 Monthly Dashboard Update, Continuous Monitoring and Improvement

The dashboard is updated every month with the most recent data available. Acknowledging the dynamic nature of data and community needs, we are dedicated to the ongoing monitoring and refinement of the model and dashboard, ensuring they remain relevant and effective in addressing the evolving challenges of food insecurity.

4 Project Workflow

This section outlines the project workflow, detailing the journey from data collection to dashboard deployment. It covers the acquisition of private and public data, their transformation through rigorous processing, and the presentation in an interactive dashboard. Each step is designed to ensure integrity, security, and utility, setting the foundation for a robust data-driven response to food insecurity.

4.1 Data Collection

Private Data: Sensitive data are collected from local entities such as Feeding San Diego, Jacobs & Cushman San Diego Food Bank, San Diego Gas & Electric (SDGE), and the County of San Diego Health & Human Services Agency. These datasets are received via email, stored on a secure Google Drive, and transferred to the project's private GitHub repository.

Public Data: Additional data are sourced from public agencies, including the Bureau of Labor Statistics, Zillow, U.S. Census Bureau, and the San Diego Association Of Governments (SANDAG).

4.2 Data Processing and Model Development

The processing and analysis of both private and public datasets are conducted through a central Jupyter notebook, *main.ipynb*. This main script executes several Python scripts sequentially to handle different aspects of the data processing and model development:

- *Import_socioeconomics.py*: Imports and processes socioeconomic data.
- *Import_FoodBanks.py* and *Create_FoodBank_data.py*: Process data received from the Food Banks.
- *Import_ACS.py*: Imports and processes data from the American Community Survey.
- *Generate_Forecasts.py*: Generates forecast data for food distribution.
- *Create_the_ZipCounty_tabs.py*, *Create_the_data_tab2s.py*, and *Create_the_data_forecasts.py*: Prepare data for the dashboard.

4.3 Output Generation

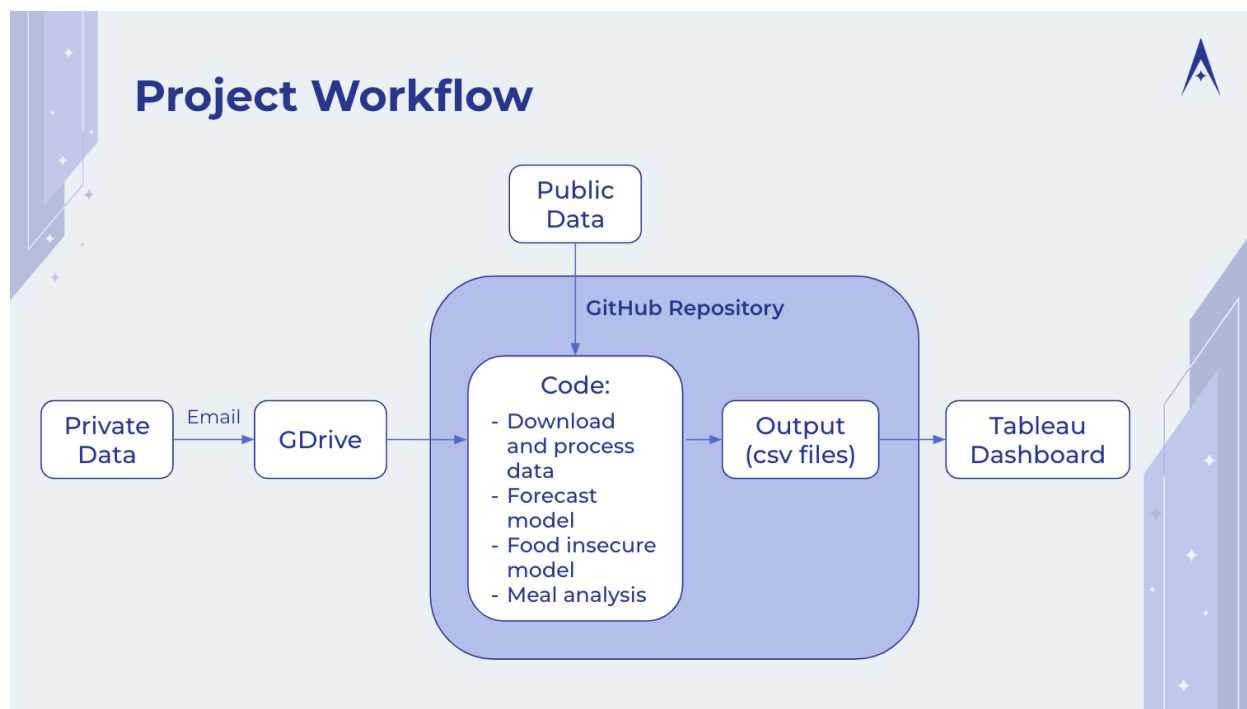
The scripts compile the processed data into several CSV files, which serve as the primary data source for the subsequent visualization stage.

4.4 Visualization and Dashboard Development

The CSV files are linked to a Tableau workbook, which is used to build an interactive dashboard. This Tableau workbook is then published to Tableau Public, albeit with restricted access to ensure data privacy.

4.5 Dashboard Deployment

The finished dashboard is embedded into DSA's website. Access to the dashboard is secured with a password, ensuring that only authorized personnel can view the information it contains.



5 Datasets

In this section, we explore the datasets that form the backbone of our analysis. The starting point of January 2018 ensures a comprehensive view while avoiding anomalies due to the COVID-19 pandemic. Each dataset—from labor statistics to housing data and beyond—is meticulously collected, processed, and incorporated into our models and dashboard. This timeframe allows us to leverage a significant period of stability in the data to establish baselines and trends that inform our predictive capabilities and understanding of food insecurity dynamics.

We store all data in a private GitHub repository under DSA's administration.

5.1 San Diego County, Zip Codes, Cities, Districts

The starting point is San Diego County and the physical makeup of it in which people reside. The list of zip codes, cities, and districts that make up San Diego county dictate every proceeding step of this project.

5.1.1 Data collection

From <https://www.unitedstateszipcodes.org/zip-code-database> (using the Free option, "Excel Format (data only)" selection), we collect the county, zip codes, and cities.

From <https://sdmg.senate.ca.gov/zipcodedirectory>, we collect the congressional districts and their associated zip codes.

5.1.2 Scope

The county, zip codes, cities, and districts data are up-to-date as of January 2024.

5.1.3 Data processing/Preprocessing

Because many PO Boxes are recorded as having their own zip code, we squash all such zip codes (and "UNIQUE" single-buildings with their own zip codes) into their larger zip code area. An additional number of zip codes are decommissioned and/or have very little data that can be attributed to them—be it from the Bureau of Labor Statistics or other socioeconomic measures or from the food banks and CalFresh. To get around this and reduce the number of null or zero values being carried on throughout this project, those zip codes are also squashed into the most appropriate zip code surrounding/nearest them.

We determine what is the most appropriate zip code by looking at Tableau's map and selecting the zip code that overlaps the PO Box. In the cases where Tableau did not display a clear answer, we used zipdatamaps.com to identify the zip code that overlapped the PO Box.

A Python dict of these to-be-removed zip codes and their corrected zip codes are saved as the file “data/zipcode_relabeling.csv” and then applied (or “mapped”) throughout the project via the Pandas library. This mapping can be found anywhere the variable “mapping” is used.

We apply this mapping in later steps of the project rather than removing the PO Box zips from the start in order to continue capturing any meal distributions attributed to those zips, and for consistency across all socioeconomic data captures.

5.2 Bureau of Labor Statistics

The U.S. Bureau of Labor Statistics “is the principal fact-finding agency for the Federal Government in the broad field of labor economics and statistics.” (from [bls.gov](https://www.bls.gov))

5.2.1 Data collection

From <https://api.bls.gov/publicAPI/v2/timeseries/data>, we collect Consumer Price Index (CPI), Electricity Prices, Unemployment Rate, and Gas Prices data.

This data is collected and processed in “pipeline/import_socioeconomics.py” and saved out into several CSVs:

- data/BLS_CPI/BLS_CPI.csv
- data/BLS_ELECTRICITY/BLS_ELECTRICITY.csv
- data/BLS_UNEMPLOYMENT/BLS_UNEMPLOYMENT.csv
- data/BLS_GASOLINE/BLS_GASOLINE.csv

5.2.2 Scope

The data we collect is from January 1st, 2020 through the present. All data is recorded at the county level.

Electricity Prices, Unemployment Rate, and Gas Prices are recorded monthly. The CPI data is only recorded every other month on the odd-numbered months.

5.2.3 Data processing/Preprocessing

Minimal preprocessing is done to the data. We use the Python `json` library to pull and parse the raw data from BLS’s website, and load it into a Pandas DataFrame. The date column is converted to a datetime type and then used to sort the DataFrame chronologically.

To fill in every other month of the CPI data, we use Python's Pandas library to resample and interpolate. Resampling uses the first of every month with `.resample(rule="MS").` Interpolation thus uses time as the index with `.interpolate(method='time').`

5.3 Zillow

"Zillow is an online real estate marketplace for finding and sharing information about homes, real estate, and mortgages." (from [Netify](#))

5.3.1 Data collection

From

https://files.zillowstatic.com/research/public_csvs/zori/County_zori_sm_month.csv?t=1673666128, we collect the Rent Prices data.

This data is collected and processed in "pipeline/Import_socioeconomics.py" within the function `zillow_rent()`, and saved out as the CSV file "data/ZILLOW_RENT/ZILLOW_RENT.csv".

5.3.2 Scope

The earliest records from this Zillow source are from January 1st, 2015, and are recorded monthly through the present. All data is recorded at the county level.

5.3.3 Data processing/Preprocessing

We use the Python `requests` library to pull and parse raw data from zillowstatic's website, and load it into Pandas DataFrames. It is then filtered for row(s) only within the San Diego region (county), and columns that are dates. The DataFrame is then melted into two columns of dates and rent prices. The date column is converted to a datetime type and then used to sort the DataFrame chronologically.

5.4 U.S. Census Bureau: American Community Survey

The American Community Survey (ACS) from the U.S. Census Bureau "is the premier source for detailed population and housing information about our nation." (from [census.gov](#))

5.4.1 Data collection

From <https://api.census.gov/data/2021/acs/acs5>, we collect the Population, Veteran Population, White, Black or African American, American Indian and Alaska Native, Asian, Native Hawaiian and Other Pacific Islander, Other Race, Two or More Races, Not Hispanic or Latino Origin,

Hispanic or Latino Origin, and Renter Population data. The corresponding codes are B01003_001E, B21001_002E, B02001_002E, B02001_003E, B02001_004E, B02001_005E, B02001_006E, B02001_007E, B02001_008E, B03003_002E, B03003_003E, and B25008_003E.

From <https://api.census.gov/data/2021/acs/acs5/subject>, we collected the Population in Poverty, Population with Disability, Senior Population (60y+), Households Receiving SNAP, and Median Income data. The corresponding codes are S1701_C01_040E, S1810_C02_001E, S0102_C02_001E, S2201_C03_001E, and S1903_C03_001E.

From <https://api.census.gov/data/2021/acs/acs5/profile>, we collected the Youth Population data. The corresponding code is DP05_0019E.

This data is collected and processed in “pipeline/Import_ACS.py” within the functions `acs2021_zips()` and `acs2021_county()`. Additional processing steps are performed within these functions before the ACS data is saved as CSV files (see #5.3.2).

5.4.2 Scope

The data was recorded by ACS on December 8th, 2022, and measured on either a zip code level or county level.

5.4.3 Data processing/Preprocessing

We use the Python `requests` library to pull and parse the raw data from ACS’s website, and load it into Pandas DataFrames. The zip codes we pull from ACS are from a list saved internally, called “zipcodes_complete.csv”, not all of which are in ACS’s database—such zip codes are printed out in the console and then the code moves on to try pulling the next zip code.

Some values are listed as “-666666666” or null values. They are replaced with 0.

5.5 SDGE: LIHEAP

The Low-Income Home Energy Assistance Program (LIHEAP) from San Diego Gas & Electric (SDGE) is a financial assistance program for “[f]amilies that are struggling financially or in a crisis situation...toward their past-due energy bill”. (from [sdge.com](https://www.sdge.com))

5.5.1 Data collection

SDGE emails DSA a monthly LIHEAP report .XLSX file, from which we use the `Pledge_Amount` and `CV_EnergyAssistance[Post Code (Service Address)]` data.

This data is collected and processed in “pipeline/Import_socioeconomics.py” within the function `liheap()`, and saved out as the CSV file “data/LIHEAP/SDGE_LIHEAP.csv”.

5.5.2 Scope

The data is recorded monthly by SDGE at the zip code level. Only the most recent month of data is used in the dashboard each time it is updated.

5.5.3 Data processing/Preprocessing

The downloaded file’s zip codes are recorded with 9 digits, so the dash and 4 ending digits are removed. We then use our `zipcodes_complete.csv` file to only save LIHEAP records whose zip codes are in our list. All rows with the same zip code are added together to get a sum total of LIHEAP pledges for that zip code. The columns are lastly renamed to the more recognizable (and shorter) Zip Code and LIHEAP Request’.

5.6 SANDAG

The San Diego Association Of Governments (SANDAG) is a “planning organization and a council of governments” (from sandag.org) which records measures such as unemployment rates.

5.6.1 Data collection

From https://opendata.sandag.org/Economy/Unemployment_Wide/wr4p-hajy, we use the unemployment rates for all zip codes, specifically for the most recent month that SANDAG has recorded.

This data is collected and processed in “pipeline/Import_socioeconomics.py” within the function `sandag_unemployment()`, and saved out as the CSV file “data/SANDAG_UNEMPLOYMENT/SANDAG_UNEMPLOYMENT.csv”.

5.6.2 Scope

The data was recorded monthly by SANDAG between March 2020 and February 2023, though we only use the most recent (February 2023) row. All data is recorded at the zip level.

5.6.3 Data processing/Preprocessing

The downloaded file is organized with dates as the rows and zip codes as the columns. For the purpose of the Socioeconomics tab, we only need the most recent month, so just the bottom row is extracted from the downloaded file.

Zip code columns are named with the convention “<zip number>_ <zip name>” (i.e. “91950_National City”), so the column names are truncated to just the zip number in order to match them with our existing style convention.

Lastly, the DataFrame is transposed so it becomes a 2-column DataFrame of zip codes and unemployment values.

5.7 Feeding San Diego

Feeding San Diego (FSD) is a food bank founded in 2007, “leading hunger-relief and food rescue organization in San Diego County” (from [feedingsandiego.org](https://www.feedingsandiego.org)) as a part of the Feeding America organization.

5.7.1 Data collection

FSD emails Data Science Alliance a monthly report of food donated in an .XLSX file, from which we use the zip code row headers and Gross Weight column data.

This data is collected and processed in “pipeline/Import_FoodBanks.py” within the functions FSD_process_new_month() and FSD_import(), and saved out as the CSV file “data/FEEDING_SD_FOOD/FSD_FOOD.csv”.

5.7.2 Scope

The data is recorded monthly by FSD (starting in January 2018 to current), at the zip code level.

5.7.3 Data processing/Preprocessing

Any empty whitespace rows and columns are removed first.

The downloaded file is organized with zip code headers rows, proceeded by rows that correspond with the last 4 digits of the 9 digit zip codes they belong to. Only the 5 digit zip code header rows are kept, accomplished by filtering for rows that start with “9”.

Because each month's data is appended (as a row) to an existing file of the past FSD monthly data, we prepare the new month's column-wise data to be turned into a row. The date—in the form of "YYYY FullMonthName"—is appended at the top. Then the date and zip codes are set as the index, and the DataFrame is transposed (so the columns are now the date and zip codes, and the values are the Gross Weight of food).

The download file uses commas in place of periods, so this is changed to use periods. Lastly, a sum total of the entire month's total weight is appended as a 'FSD Total' column to the end.

The new month's data is appended to the existing CSV of past FSD monthly pounds data, and any null values are replaced with 0.

5.8 Jacobs & Cushman San Diego Food Bank

Jacobs & Cushman San Diego Food Bank (SDFB) is a food bank and "the largest hunger-relief organization in San Diego County" (from sandiegofoodbank.org).

5.8.1 Data collection

SDFB emails Data Science Alliance a monthly report of food donated in an .XLSX file, from which we use the zip code and Agency Total data.

This data is collected and processed in "pipeline/Import_FoodBanks.py" within the function `SDFB_import()`, and saved out as the CSV file "data/SDFB_FOOD/SDFB_FOOD.csv".

5.8.2 Scope

The data is recorded monthly by SDFB (starting in January 2018 to current), at the zip code level.

5.8.3 Data processing/Preprocessing

For each new month's data, only the two columns corresponding to zip codes and pounds of food are kept, and the rest of the downloaded file's columns are dropped. All rows with the same zip code are added together to get a sum total of pounds for that zip code. The DataFrame is pivoted so that the index is the date, columns are zip codes, and values are the pounds of food.

We then use our `zipcodes_complete.csv` file to only save SDFB columns whose zip codes are in our list. Lastly, a sum total of the entire month's total weight is appended as a SDFB Total' column to the end.

That DataFrame representing the new month of data—now just a single row—is appended to the existing CSV of past SDFB monthly pounds data, and any null values are replaced with 0.

5.9 San Diego Health & Human Services Agency: CalFresh

Federally known as the Supplemental Nutrition Assistance Program (SNAP), CalFresh “is the largest food program in California and provides an essential hunger safety net” (from cdss.ca.gov).

5.9.1 Data collection

CalFresh emails Data Science Alliance a monthly report of dollars put towards the program in an .XLSX file, from which we use the Zip Code and total dollars for the month’s data.

This data is collected and processed in “pipeline/Import_CalFresh.py” and “pipeline/calfresh_update.ipynb”, and saved out as the CSV file “data/CALFRESH/CALFRESH_MEALS.csv”.

5.9.2 Scope

The data is recorded monthly by SDFB (starting in January 2020 to current), at the zip code level.

5.9.3 Data processing/Preprocessing

Zip codes labeled as “Other*” are removed, and the whole DataFrame is transposed so the Zip Codes are the columns and the CalFresh dollars are the values. That DataFrame representing the new month of data—now just a single row—is appended to the existing XLSX of past CalFresh monthly dollars data.

To make the dollar amounts comparable to pounds of food from FSD and SDFB, we must turn the dollar amounts into amounts of meals. The costs of meals were historically determined at the end of the year in 2017 through 2021, so a new DataFrame with those dates and known meal costs are created. To fill in the monthly gaps between the December values, we use interpolation with ‘pchip’ method. To fill in the monthly gaps between then and the current month, we use the Consumer Price Index (CPI).

The rate of change in CPI is used to calculate the change of meal costs. This is accomplished by finding the monthly % change in CPI, and multiplying that % by each month’s meal cost to get the next month’s meal cost. The CalFresh dollars DataFrame is then divided by the cost of meals, giving us the CalFresh meals per month per Zip Code.

5.10 Calpads

The California Longitudinal Pupil Achievement Data System (CALPADS) data for grades K–12 is “the starting point for determining...supplemental and concentration grant calculations,” of which we are most interested in the “free or reduced-price meal (FRPM) eligibility” (from [cde.ca.gov](https://www.cde.ca.gov)).

5.10.1 Data collection

From <https://www.cde.ca.gov/ds/ad/filescupc.asp> for each school year, we collect CALPADS Unduplicated Pupil Count (UPC), which is “an unduplicated count of students who are FRPM eligible (including foster youth students) or EL eligible. This count corresponds to the ‘Total Unduplicated FRPM/EL Eligible Count’ provided in CALPADS certification report 1.17: *FRPM/English Learner/Foster Youth - Count*.”

This data is collected and processed in “pipeline/Import_CalFresh.py” within the functions `calpads()` and `calpads_calendar()`, and saved out as the CSV file “data/CALPADS/CALPADS_MEALS.csv”.

5.10.2 Scope

The data we use was recorded yearly by CALPADS between Fall of 2019 and Spring of 2023 for each school year. The first school year we use is 2019-20 to match with our starting 2020 CalFresh data, and then the 2020-21, 2021-22, and 2022-2023 school years.

5.10.3 Data processing/Preprocessing

Each school within the San Diego county were matched with their associated zip code from Google maps and the dataset <???. That table—still containing the CALPADS Unduplicated Pupil Count (UPC)—is then summed up by each zip code per school year for a total of “students who are eligible for school meals.” The proceeding steps rely on several assumptions we explain as well.

We assume these students are eligible for 2 meals a day (e.g. breakfast and lunch). We assume students eat at school 20 days per month. We assume students are in school 9 of the 12 months of the year, wherein they are not receiving school meals during the summer months (June – August). To implement this math, each zip code’s total “students who are eligible for school meals” are multiplied by $\times 2 \times 20$ to represent one month’s worth of meals. We then assign these meal counts by month.

- With the school year of 2019-20, we associate the months January – May 2020.

- With the school year of 2020-21, we associate the months September 2020 – May 2021.
- With the school year of 2021-22, we associate the months September 2021 – May 2022.
- With the school year of 2022-23, we associate the months September 2022 – May 2023 and September 2023 – January 2024 because we do not yet have the CalPads data for 2023-24.

6 Deployment: Dashboard Design and Functionality

This section delves into the deployment phase of the project, focusing on the design and functionality of the dashboard constructed in Tableau. Below we provide insight into the rationale behind choosing Tableau as our primary visualization tool, the purpose and development process of each dashboard tab, and the technical nuances that underpin its interactivity and user experience.

6.1 Why Tableau

Tableau is selected for its advanced data visualization capabilities, allowing for the creation of an interactive and user-friendly dashboard that can effectively display complex datasets in a comprehensible manner. Its ability to handle large volumes of data and provide real-time updates makes it an ideal tool for monitoring food distribution metrics and identifying hunger hot spots within San Diego County.

6.2 “Food Bank Distribution” tab

6.2.1 Purpose

The “Food Bank Distribution” tab is dedicated to presenting the combined efforts of Feeding San Diego and the Jacobs & Cushman San Diego Food Bank. It shows the volume of food distributed across various regions and allows users to track changes over time, identify areas of high demand, and plan for future food distribution needs.

6.2.2 Development

After importing and cleaning the FSD and SDFB data (#3.6 and #3.7), the two are added together for a “Both Food Banks” data set. This is done simply by adding together the pounds of food per zip code in “pipeline/Import_FoodBanks.py” within the function `both_import()`, and saved out as the CSV file “data/BOTH_FOOD/BOTH_FOOD.csv” and saved out with FSD and SDFB in the excel file “data/FOOD_EXCEL_FILES/FOOD_BANKS_DATA.xlsx”.

The food bank data is then processed in “pipeline/Create_FoodBank_data.py”. In the function `data_region_and_FB_city_xlsx()`, all zip codes and their zip name are saved out in the CSV file “data/OUTPUT/data_region.csv”. In the same function, we create an excel file for viewing the data by city, in which the columns cities and rows are months. A tab for each of the three food bank options—FSD, SDFB, and both—are saved in this excel file

“data/FOOD_EXCEL_FILES/FOOD_BANKS_DATA_BYCITY.xlsx”. This is done by melting the DataFrames from “data/FOOD_EXCEL_FILES/FOOD_BANKS_DATA.xlsx”, merging with our “data/SD_zipcode_name.csv” DataFrame which matches zip codes with zip names (aka city names). Quickly, the pounds of food are rounded to the 3rd decimal place, all null values are filled with 0, and the typoed city name “San Diego/” is corrected to “San Diego”. The DataFrames are pivoted by city names as the columns and dates as the rows, aggregating pounds of food by summing them together. Lastly, a column for totals is appended to the end of each DataFrame.

The food bank data is also processed in the function `FB_district_xlsx()`, where we create an excel file for viewing the data by district, in which the columns are districts and rows are months. A tab for each of the three food bank options are saved in this excel file

“FOOD_EXCEL_FILES/FOOD_BANKS_DATA_BYDISTRICT.xlsx”. This is done by melting the DataFrames from “data/FOOD_EXCEL_FILES/FOOD_BANKS_DATA.xlsx”, and merging with our “OUTPUT/data_district.csv” DataFrame which matches zip codes with congressional districts. Quickly, the pounds of food are rounded to the 3rd decimal place and all null values are filled with 0. The DataFrames are pivoted by districts as the columns and dates as the rows, summing the pounds of food together as they’re grouped. Lastly, a column for totals is appended to the end of each DataFrame.

The above steps could belong in a section of its own as #3.9 because they create CSVs and excel files that only lead up to this point. Now we get into the CSV file that is used for Tableau’s Food Bank Distribution tab: “OUTPUT/data.csv”.

This is created in the functions `data()`, `with_calfresh()`, and `data_county()`. In `data()`, the three DataFrames from “data/FOOD_EXCEL_FILES/FOOD_BANKS_DATA.xlsx” are melted together and merged with our “data/SD_zipcode_name.csv” DataFrame to match zip codes with zip names. All null values are filled with 0, the pounds of food are rounded down to nearest whole numbers, and the typoed city name “San Diego/” is corrected to “San Diego”. We next merge Food Insecurity data from

“data/FOOD_INSECURITY/FOOD_INSECURITY_ESTIMATION_ZIP.csv” into the DataFrame, and add a column ‘LbsByInsucure’ (pounds of food per insecure populace) by dividing the total food pounds by insecure population. Another column which is a duplicate of the date column is added, which will allow Tableau to sort by fiscal years (in addition to calendar years). Finally, we

merge the Congressional Districts data from “data/SD_congressional_districts.csv” into the DataFrame, and fill any nulls with 0s.

In `with_calfresh()`, the recently-made “OUTPUT/data.csv” DataFrame is merged with “data/CALFRESH/CALFRESH_MEALS.csv” DataFrame, from which only the `CalfreshMeals` column is retained. After all null values are filled with 0, we add a `Meals` column by dividing the total food pounds by a designated unit (default = 1.2 lb/meal) and then adding `CalfreshMeals` to that quotient. Another column ‘`MealsByInsecure`’ (meals per insecure populace) is added to the DataFrame by dividing `Meals` by the food insecure population. The DataFrame is then merged with “data/FOOD_INSECURITY/DATA_FOR_ESTIMATION/ZIP_ALL.csv” for the poverty population column. A column is added to the DataFrame for ‘`MealsByPoverty`’ (meals per poverty populace) by dividing `Meals` by the poverty population. All null values and all infinity values are replaced with 0.

In `data_county()`, the “OUTPUT/data.csv” DataFrame is summed up by zip code so there is only one row for the entire San Diego county per month.

6.2.3 Discussion

The "Food Bank Distribution" tab serves as a strategic tool, providing valuable insights into food distribution patterns by Feeding San Diego and the Jacobs & Cushman San Diego Food Bank. The data visualized here is crucial for assessing the effectiveness of current distribution networks and for identifying areas where resources may be underutilized or in high demand. Use cases for this tab include resource allocation planning, impact assessment of food bank programs, and immediate response planning to emerging needs. Stakeholders can leverage this tab to optimize distribution routes, prepare for seasonal variations in food assistance needs, and coordinate with local agencies for emergency response.

6.3 “Socioeconomic Data” tab

6.3.1 Purpose

This tab aims to provide a comprehensive view of the socioeconomic factors that influence food insecurity. By combining data from various sources, including ACS, CalFresh, and LIHEAP, the dashboard can highlight correlations between socioeconomic conditions and food assistance needs.

6.3.2 Development

After importing and cleaning the ACS data (#5.3), this dataframe was merged with a few additional data sources in “pipeline/Import_ACS.py” within the functions `acs2021_zips()` and

acs2021_county() to work towards the Socioeconomic Data tab's data sources. These CSVs created are "data/ACS_SOCIOECONOMIC/acs2021_zip.csv" and "data/OUTPUT/data_acs2021_county.csv". The additional columns are San Diego Gas & Electric's LIHEAP Request (#5.4), SANDAG unemployment rate (#5.5), and two columns of predicted food insecurity from "data/FOOD_INSECURITY/FOOD_INSECURITY_ESTIMATION_ZIP.csv".

The CSV files that are used for Tableau's Socioeconomic Data tab are created in the file "pipeline/Create_the_ZipCounty_tabs.py". Those CSV files are:

- data/OUTPUT/data_tab2_varsall.csv
- data/OUTPUT/data_tab2_vars.csv
- data/OUTPUT/data_tab2_race.csv
- data/OUTPUT/data_tab2_ethni.csv

In the function data_tab2_varsall(), the "data/ACS_SOCIOECONOMIC/acs2021_zip.csv" DataFrame is merged with our "data/SD_zipcode_name.csv" for its zip names. Ten columns (those to do with race, ethnicity, and poverty population of each zip code) are divided by the total population for each zip code, changing the unit of measurement to percentages for these ten columns. This DataFrame is then merged with "data/CALFRESH/CALFRESH_MEALS.csv" for the CalFresh dollars and meals amounts. Lastly, all null values and all infinity values are replaced with 0.

In the function data_tab2_vars(), we read in the recently-made "data/OUTPUT/data_tab2_varsall.csv" and drop the nine race and ethnicity related columns. The function data_tab2_race() does the same but retains only the seven race related columns ('White', 'Black or African American', 'American Indian and Alaska Native', 'Asian', 'Native Hawaiian and Other Pacific Islander', 'Other Race', and 'Two or More Races'). The function data_tab2_ethni() does the same but retains only the two ethnicity related columns ('Not Hispanic or Latino Origin' and 'Hispanic or Latino Origin').

6.3.3 Discussion

The "Socioeconomic Data" tab presents a rich layer of context, correlating economic indicators with food insecurity trends. This information is instrumental for users to identify the socioeconomic drivers behind food assistance demand. Use cases include the development of targeted assistance programs, informing policy decisions to mitigate food insecurity, and conducting longitudinal studies on the impacts of economic changes on food distribution needs. It can also aid in tailoring educational campaigns and advocacy efforts to address the root causes of hunger in various demographics.

6.4 “Info Tab”

6.4.1 Purpose

The purpose of the "Info Tab" is to provide transparency regarding the data sources, methodologies, and assumptions used in the dashboard. It serves as a reference point for users to understand the context and limitations of the data presented.

6.4.2 Development

The development of the "Info Tab" involves compiling detailed documentation on data sources, processing steps, and any analytical methods applied to the data. This information is presented in a clear and accessible format within the dashboard.

6.4.3 Discussion

The "Info Tab" is a repository of transparency, documenting the data's origins, unit of measurement, the analytical methodologies applied, and the assumptions underpinning the dashboard's information. This tab is valuable for data validation, peer review, and ensuring the ethical use of information. Users can utilize this tab to understand the data lifecycle, replicate study results, or extend the analysis for further research. It also serves as an educational resource for training stakeholders on interpreting dashboard data and understanding the significance of various data points in the broader scope of food security initiatives.

6.5 Technical Details: Tableau Parameters and Calculated Fields

This section in the documentation aims to provide an explanation of the calculated fields that have been created in the Tableau dashboard. The codes for each field can be found in the Tableau workbook file. The purpose is to elucidate the reasoning behind the creation of each field, ensuring that whoever uses the dashboard or makes further edits gains a comprehensive understanding of each field.

The format of each entry is as follows:

- Parameter/Calculated field name
- Screenshot of the field in the workbook
- The rationale

Not all of the created fields shown in the workbook were ultimately utilized in the final version of the dashboard. Some of them were specifically designed for testing purposes. This document only includes the fields that were implemented and deemed relevant for the final version of the dashboard.

6.5.1 Parameter

6.5.1.1 Measure Selection

Edit Parameter [Measure Selection]

Name
Measure Selection

Properties

Data type
String

Display format
Food (lbs)

Current value
Food (lbs)

Value when workbook opens
Current value

Allowable values

☐ All
☒ List
☐ Range

Value	Display As
Food (lbs)	Food (lbs)
Pounds of Food Distri...	Pounds of Food Distri...
Click to add	

☒ Fixed
☐ When workbook opens

Add values from

Remove Selected

Cancel

OK

This parameter has been created to incorporate a filter for the map and dashboard. Users can select either 'Food (lbs)' or 'Pounds of Food Distributed by Food Insecure' to display relevant information.

6.5.1.2 Select Congressional District

Edit Parameter [Select Congressional District]

Name
Select Congressional District

Properties

Data type
Integer

Display format
48

Current value
48

Value when workbook opens
Current value

Allowable values

☐ All
☒ List
☐ Range

Value	Display As
48	48
49	49
50	50
51	51
52	52
Click to add	

☒ Fixed
☐ When workbook opens

Add values from

Remove Selected

Cancel

OK

This parameter, along with the calculated fields 'sync dis' and 'District' (section 2-15), is created to synchronize the selection of the congressional district from 'data.csv' with the forecasted district data. This parameter is added to 'Map-District-cy' sheet.

Select Congressional District

48

The filter in 'Foodbank-dis-CY' is created from this parameter, instead of a map filter like in other dashboards.

In addition to the group of calculations consisting of, 'Dis-CY-Real-Both', 'Dis-CY-Real-SDFB', 'Dis-CY-Real-FSD', 'Dis-CY-Fore-Both', 'Dis-CY-Fore-SDFB', 'Dis-CY-Fore-FSD' (section 2-16), 'Dis-CY-Fore-Both-Display', 'Feeding San Diego', and 'Jacob & Cushman SDFB' (section 2-17), these new added entries have been developed to address a display issue in the 'Calendar Year

Forecast' table at the District level on the 'Foodbank-dis-CY' dashboard. Currently, Tableau only sums the forecast data, whereas this table requires the summation from January to December of the calendar year, incorporating both the 'data.csv' and the 'data_forecast_district.csv' datasets.

6.5.2 Calculated Field

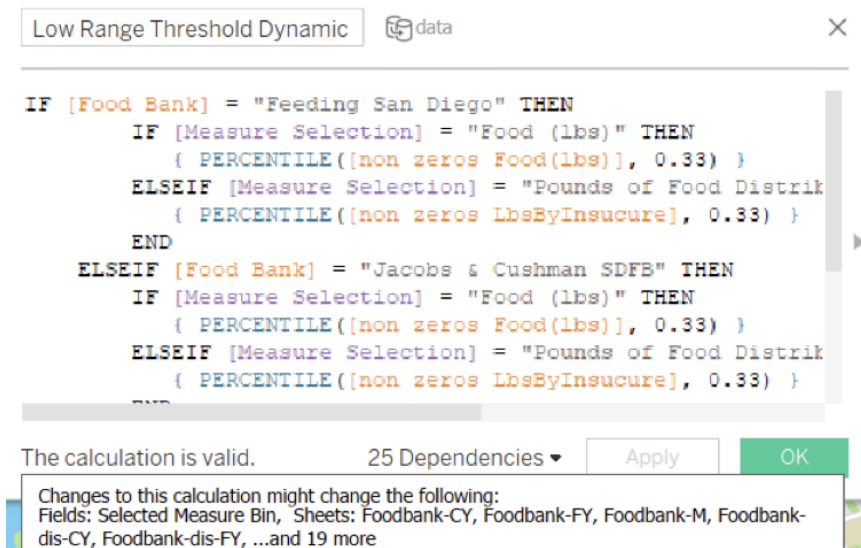
6.5.2.1 Selected Measure



The 'Selected Measure' calculated field is designed to determine the appropriate measure to display based on the selected options. The rationale behind this calculation is as follows:

If the 'Food Bank' selection filter is set to 'Feeding San Diego', the calculation checks the 'Measure Selection' parameter and read the [Food(lbs)] column from 'data.csv' if 'Food (lbs)' is selected or read the [LbsByInsucure] column if 'Pounds of Food Distributed by Food Insecure Population' is selected. The same logic applies to the other two food bank options (Jacob and both).

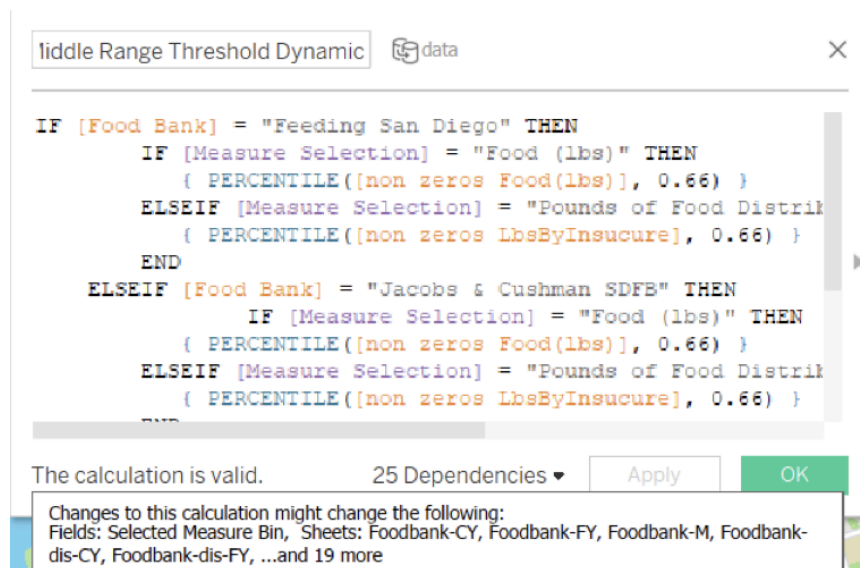
6.5.2.2 Low Range Threshold Dynamic



The 'Low Range Threshold Dynamic' calculated field serves the purpose of dynamically determining the low range threshold value based on the selected options. The rationale behind this calculation can be summarized as follows:

If the 'Food Bank' map filter is set to 'Feeding San Diego', the calculation checks the 'Measure Selection' parameter and returns the 33rd percentile value of the non-zero values in the '[Food(lbs)]' measure if 'Food (lbs)' is selected, or the 33rd percentile value of the non-zero values in the '[LbsByInsecure]' measure if 'Pounds of Food Distributed by Food Insecure Population' is selected. Similar rationale applies to other food bank options.

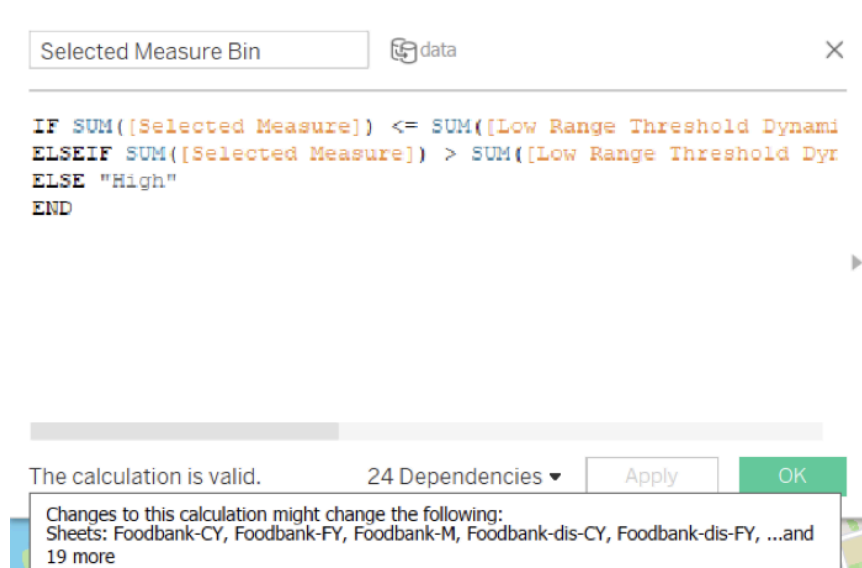
6.2.2.3 Middle Range Threshold Dynamic



The 'Middle Range Threshold Dynamic' calculated field is designed to dynamically determine the middle range threshold value based on the selected options. The rationale behind this calculation can be summarized as follows:

If the 'Food Bank' map filter is set to 'Feeding San Diego', the calculation checks the 'Measure Selection' and returns the 66th percentile value of the non-zero values in the '[Food(lbs)]' measure if 'Food (lbs)' is selected, or the 66th percentile value of the non-zero values in the '[LbsByInsucure]' measure if 'Pounds of Food Distributed by Food Insecure Population' is selected.

6.2.2.4 Selected Measure Bin



The 'Selected Measure Bin' calculated field serves the purpose of categorizing the selected measure into bins based on predefined thresholds. The rationale behind this calculation can be summarized as follows:

- If the sum of the 'Selected Measure' is less than or equal to the sum of the 'Low Range Threshold Dynamic', the calculated field assigns the value 'Low' to indicate that the measure falls within the low range.
- If the sum of the 'Selected Measure' is greater than the sum of the 'Low Range Threshold Dynamic' but less than or equal to the sum of the 'Middle Range Threshold Dynamic', the calculated field assigns the value 'Middle' to indicate that the measure falls within the middle range.
- If the sum of the 'Selected Measure' exceeds the sum of the 'Middle Range Threshold Dynamic', the calculated field assigns the value 'High' to indicate that the measure falls within the high range.

By comparing the sum of the 'Selected Measure' against the dynamically calculated thresholds, this calculated field facilitates the grouping of the measure into distinct bins, allowing for easier analysis and visualization of the data.

6.2.2.5 Food Bank Index

Food Bank Index

data

×

```

IF [Food Bank] = "Both" THEN "Both"
ELSEIF [Food Bank] = "Feeding San Diego" THEN "Feeding San D:
ELSEIF [Food Bank] = "Jacobs & Cushman SDFB" THEN "Jacobs & (
END

```

The calculation is valid.

7 Dependencies ▾

Apply

OK

Changes to this calculation might change the following:
Fields: Sync food bank , Sheets: Foodbank-CY, Foodbank-FY, Foodbank-M, Map-County-cy,
Map-County-fis, ...and 1 more

The 'Food Bank Index' calculated field is designed to assign an index value to each food bank based on the selected 'Food Bank' option. The rationale behind this calculation can be summarized as follows:

- If the 'Food Bank' is set to 'Both', the calculated field assigns the value 'Both' to indicate that data from both 'Feeding San Diego' and 'Jacobs & Cushman SDFB' food banks should be included.
- If the 'Food Bank' is set to 'Feeding San Diego', the calculated field assigns the value 'Feeding San Diego' to indicate that only data from the 'Feeding San Diego' food bank should be included.
- If the 'Food Bank' is set to 'Jacobs & Cushman SDFB', the calculated field assigns the value 'Jacobs & Cushman SDFB' to indicate that only data from the 'Jacobs & Cushman SDFB' food bank should be included.

The 'Food Bank Index' calculated field acts as a categorization or index that represents the selected 'Food Bank' option. It establishes a connection to the 'Sync food bank' (see below) calculated field by providing the conditionals that determine which food bank(s) should be synchronized in the filter.

6.2.2.6 Sync food bank

 data ×

```
IF [Food Bank Index] = "Both" THEN
  [Foodbank] IN ('Feeding San Diego', 'Jacobs & Cushman SDI
ELSEIF [Food Bank Index] = 'Feeding San Diego' THEN
  [Foodbank] = 'Feeding San Diego'
ELSEIF [Food Bank Index] = 'Jacobs & Cushman SDFB' THEN
  [Foodbank] = 'Jacobs & Cushman SDFB'
END
```

The calculation is valid. 6 Dependencies ▾

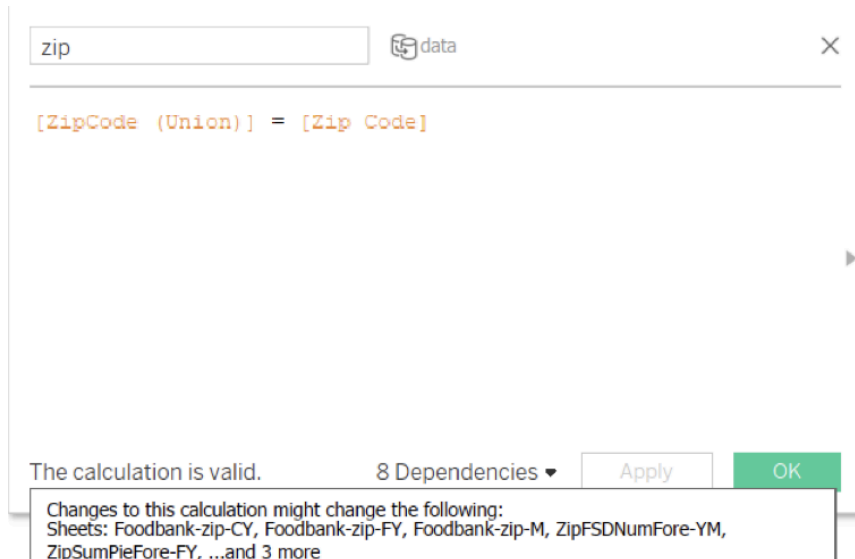
Changes to this calculation might change the following:
Sheets: Foodbank-CY, Foodbank-FY, Foodbank-M, Map-County-cy, Map-County-fis, ...and 1 more

The 'Sync food bank' calculated field serves the purpose of synchronizing the food bank filter between two datasets to prevent null values. The rationale behind this calculation can be summarized as follows:

- If the 'Food Bank Index' is set to 'Both', the calculated field checks if the 'Foodbank' is either 'Feeding San Diego' or 'Jacobs & Cushman SDFB'.
- If the 'Food Bank Index' is set to 'Feeding San Diego', the calculated field checks if the 'Foodbank' is specifically 'Feeding San Diego'.
- If the 'Food Bank Index' is set to 'Jacobs & Cushman SDFB', the calculated field checks if the 'Foodbank' is specifically 'Jacobs & Cushman SDFB'.

By implementing these conditional statements, the 'Sync food bank' calculated field harmonizes the food bank filter across datasets, ensuring that only relevant data is included based on the selected 'Food Bank Index'. This helps prevent null values and provides consistency in the filtered data.

6.2.2.7 Zip



The 'zip' calculated field serves the purpose of comparing the values between the '[ZipCode (Union)]' and '[Zip Code]' fields. The rationale behind this calculation can be summarized as follows:

The code `'[ZipCode (Union)] = [Zip Code]'` compares the values of the '[ZipCode (Union)]' field with the '[Zip Code]' field. If the values in both fields match, the calculated field will return 'True', indicating that the zip code is the same in both fields. Conversely, if the values do not match, the calculated field will return 'False'. A filter is created with only the 'True' value of this field. It allows us to establish a connection between these fields and perform further analysis or filtering based on zip code consistency or discrepancies.

6.2.2.8 CurrentFisYear

data
×

```
STR(YEAR(DateFis))
```

The calculation is valid. 4 Dependencies ▾

Changes to this calculation might change the following:
Sheets: FY_Date, Foodbank-dis-FY, Foodbank-reg-FY, Foodbank-zip-FY

The 'CurrentFisYear' calculated field is created to extract the fiscal year from the 'DateFis' field and represent it as a string value. This value is displayed as texts in the dashboard.

6.2.2.9 LabelForecast_zip_Month, LabelForecast_Reg_Month, LabelForecast_Dis_Month

data
×

```
IF DATETRUNC('day',[Date (Union1)])<= TODAY() THEN 'Actual'  
ELSEIF DATETRUNC('day',[Date (Union1)])> TODAY() THEN 'Forec'  
END
```

The calculation is valid. 4 Dependencies ▾

Changes to this calculation might change the following:
Sheets: CountyTrend-YM, Foodbank-M, Foodbank-reg-M, RegTrend-YM

All three fields serve similar purposes. The only difference is the data used (e.g., LabelForecast_zip_Month reads zip-level data). These fields are used in color-coding graphs

that plot both the forecast and the historical data with month as the unit of the x-axis. Here is an explanation using of 'LabelForecast_Reg_Month'.

The 'LabelForecast_Reg_Month' calculated field is created to label data points as either 'Actual' or 'Forecast' based on a comparison between the date in the '[Date (Union1)]' field and the current date (in the format of Month-Year). The rationale behind this calculation can be summarized as follows:

- If the date in the '[Date (Union1)]' field is on or before the current date (as determined by 'TODAY()'), the calculated field assigns the label 'Actual' to indicate that the data point corresponds to actual observed values.
- If the date in the '[Date (Union1)]' field is after the current date, the calculated field assigns the label 'Forecast' to indicate that the data point corresponds to a forecasted or predicted value.

6.2.2.10 LabelForecast_Reg_Year



All three fields serve similar purposes. The only difference is the data used (e.g., LabelForecast_zip_Year reads zip-level data). These fields are used in color-coding graphs that plot both the forecast and the historical data with year as the unit of the x-axis. Here is an explanation using of 'LabelForecast_Reg_Year'.

The 'LabelForecast_Reg_Year' calculated field is designed to label data points as either 'Actual' or 'Forecast' based on the year value extracted from the '[Date (Union1)]' field. The rationale behind this calculation can be summarized as follows:

- If the year extracted from the '[Date (Union1)]' field falls between 2018 and 2022 (inclusive), the calculated field assigns the label 'Actual' to indicate that the data point corresponds to actual observed values within that time period.
- If the year extracted from the '[Date (Union1)]' field is 2023 or greater, the calculated field assigns the label 'Forecast' to indicate that the data point corresponds to a forecasted or predicted value for future periods.

Note: Here the year is hard coded. When entering the year 2024, manual changing of this field is required. This may be automated by extracting the year using the [YEAR ()] function.

6.2.2.11 YearFore_Both_C, YearFore_Both_Dis, YearFore_Both_Zip;
YearFore_Separate_C, YearFore_Separate_Dis, YearFore_Separate_C

 data ×

```
SUM(IF [FoodBank (Union1)] = "Jacobs & Cushman SDFB"
OR [FoodBank (Union1)] = "Feeding San Diego"
THEN [Food(lbs) (Union1)] END)
```

The calculation is valid.

8 Dependencies ▼

Apply

OK

Changes to this calculation might change the following:
Sheets: CountyTrend-CY, CountyTrend-FY, Foodbank-CY, Foodbank-FY, Foodbank-reg-CY,
...and 3 more

 data ×

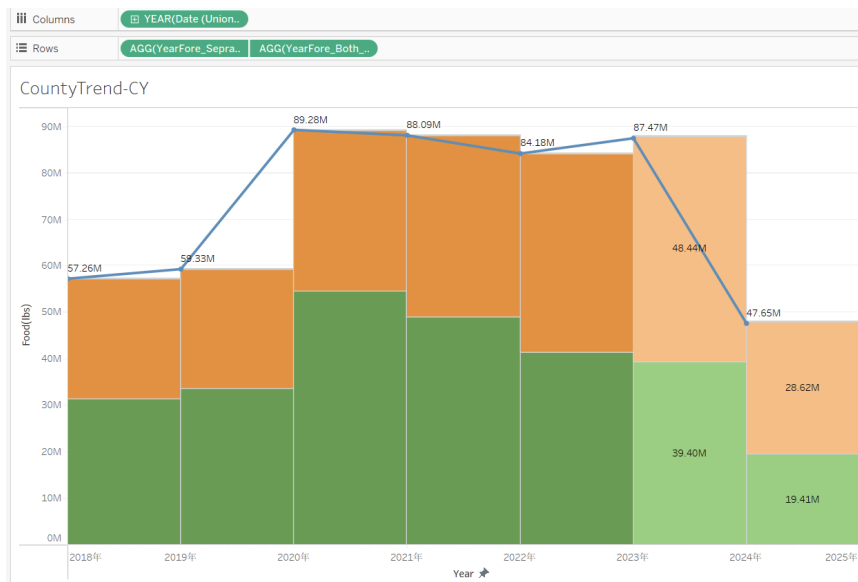
```
SUM(IF [FoodBank (Union1)] = "Both" THEN [Food(lbs) (Union1)
```

The calculation is valid. 8 Dependencies ▼

Changes to this calculation might change the following:
Sheets: CountyTrend-CY, CountyTrend-FY, Foodbank-CY, Foodbank-FY, Foodbank-reg-CY,
...and 3 more

The "YearFore_Both_C" calculated field is created to calculate the sum of the '[Food(lbs) (Union1)]' values only for the rows where the '[FoodBank (Union1)]' field is equal to "Both". The "YearFore_Seprate_C" calculated field is created to calculate the sum of the '[Food(lbs) (Union1)]' values only for the rows where the '[FoodBank (Union1)]' field is equal to either "Jacobs & Cushman SDFB" or "Feeding San Diego".

These two fields are used together as a dual axis to generate a plot that plots and color-codes the two food banks as stacked bar graphs. Additionally, a connected dot-plot representing the total value is overlaid on top of the bars:



The pairs of fields at the zip-level and district-level follow the same rationale but retrieve data from different datasets.

6.2.2.12 ForecastYear

ForecastYear

data

```
'1/' + STR(YEAR(TODAY())) + '- 12/' + STR(YEAR(TODAY()))
```

The calculation is valid.

5 Dependencies

Apply

OK

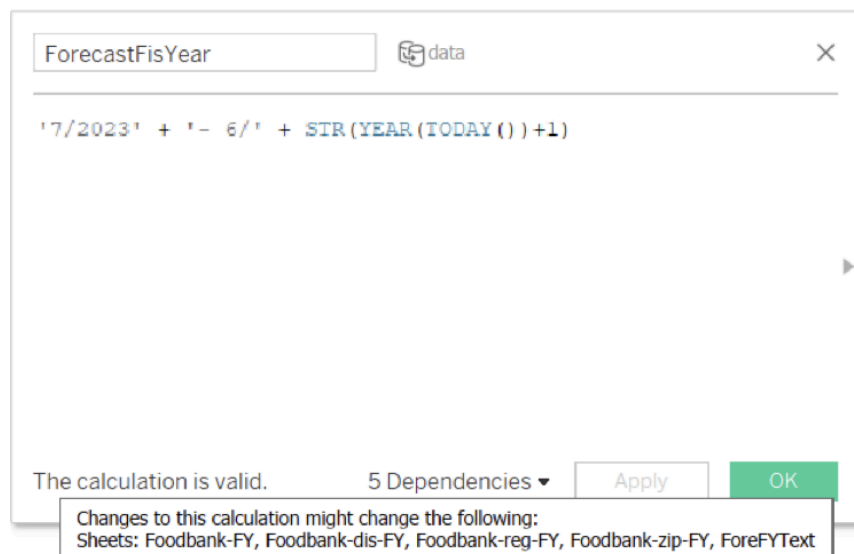
Changes to this calculation might change the following:
Sheets: Foodbank-CY, Foodbank-dis-CY, Foodbank-reg-CY, Foodbank-zip-CY, ForeYText

The rationale behind the "ForecastYear" calculated field is to generate a textual representation of the forecast year. The code concatenates different components to form the forecast year representation. Here's a breakdown of the code:

- '1/': This part represents the start month of the forecast year, which is January.
- STR(YEAR(TODAY())): This extracts the current year using the YEAR(TODAY()) function and converts it to a string using STR(). It ensures that the year dynamically adjusts based on the current date.
- '- 12/': This part represents the end month of the forecast year, which is December.

By combining these components, the calculated field "ForecastYear" will display a string representation of the forecast year in the format "1/YYYY - 12/YYYY", where "YYYY" represents the current year.

6.2.2.13 ForecastFisYear



The rationale behind the "ForecastFisYear" calculated field is to generate a textual representation of the forecast fiscal year. The "ForecastFisYear" calculated field is designed to generate a textual representation of the forecast fiscal year.

- '7/2023': This part represents the start month and year of the forecast fiscal year, which is July 2023. The value is hardcoded to '7/2023' in this example.
- '- 6/': This part represents the end month of the forecast fiscal year, which is June.
- STR(YEAR(TODAY())+1): This extracts the current year using the YEAR(TODAY()) function, adds 1 to it to get the next year, and converts it to a string using STR(). It ensures that the year dynamically adjusts based on the current date.

By combining these components, the calculated field "ForecastFisYear" will display a string representation of the forecast fiscal year in the format "7/2023 - 6/YYYY", where "YYYY" represents the next calendar year.

Note: The year '2023' is hardcoded in this calculation. Therefore, when transitioning to the next calendar year, it needs to be manually updated. However, to automate this process, the YEAR() function can be utilized. By incorporating the YEAR() function, the forecast fiscal year will dynamically adjust based on the current date, eliminating the need for manual updates.

6.2.2.14 ForecastMonth



The "ForecastMonth" calculated field is designed to generate a textual representation of the forecast month.

- `DATEADD('month', 1, TODAY())`: This part retrieves the name of the month for the next month relative to the current date. It uses the DATEADD function to add 1 month to the current date (TODAY()) and then extracts the month name using the DATENAME function.
- `STR(YEAR(TODAY()))`: This extracts the current year using the YEAR function and converts it to a string using the STR function.

By combining these components with appropriate formatting, the calculated field "ForecastMonth" will display a textual representation of the forecast month in the format "Month Year". For example, "June 2023".

6.2.2.15 sync dis and District

data
×

[Select Congressional District]

The calculation is valid. 49 Dependencies ▾

Apply OK

Changes to this calculation might change the following:
Fields: Dis-CY-Fore-Both, Dis-CY-Fore-Both-Display, Dis-CY-Fore-FSD, Dis-CY-Fore-SDHB,
Dis-CY-Real-Both, ...and 8 more, Sheets: DisBothNumFore-FY, DisBothNumFore-Y,
DisBothNumFore-YM, DisBothNumReal-FY, DisBothNumReal-Y, ...and 31 more

[Sync dis] value is equal to the selection of the [Select Congressional District] parameter. The output is a numeric number from 48 to 52 based on selection.

data
×

[CongressionalDistricts] = [sync dis]

The calculation is valid. 31 Dependencies ▾

Apply OK

Changes to this calculation might change the following:
Fields: Dis-CY-Fore-Both-Display, Dis-CY-Real-Both, Dis-CY-Real-FSD, Dis-CY-Real-SDHB,
Feeding San Diego, ...and 1 more, Sheets: DisBothNumFore-Y, DisBothNumReal-FY,
DisBothNumReal-Y, DisBothNumReal-YM, DisFSDNumFore-Y, ...and 20 more

The [District] calculated field produces a Boolean value, either true or false. When the [CongressionalDistricts] from the 'data_district.csv' matches the [sync dis] (the selected value),

[District] is set to true. To ensure that only the selected districts are included and to avoid null values, a filter is applied to connected sheets based on the true value of [District]. The parameter [Select Congressional District] along with these two calculated fields enable us to synchronize the district selection across different data sources, including data.csv, data_district.csv, and data_forecast_district.csv.

6.2.2.16 Dis-CY-Real-Both, Dis-CY- Real -SDFB, Dis-CY-Real-FSD, Dis-CY-Fore-Both, Dis-CY-Fore-SDFB, Dis-CY-Fore-FSD

Dis-CY-Real-FSD
data

```

IF [District] = TRUE
  AND [Food Bank] = 'Feeding San Diego'
  AND DATEPART('year', [Date]) = DATEPART('year', TODAY())
THEN [Food(lbs)]
END

```

The calculation is valid.
4 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:
Fields: Feeding San Diego, Sheets: DisFSDNumFore-Y, DisSumPieFore-Y, Foodbank-dis-CY

Dis-CY-Real-SDFB
data

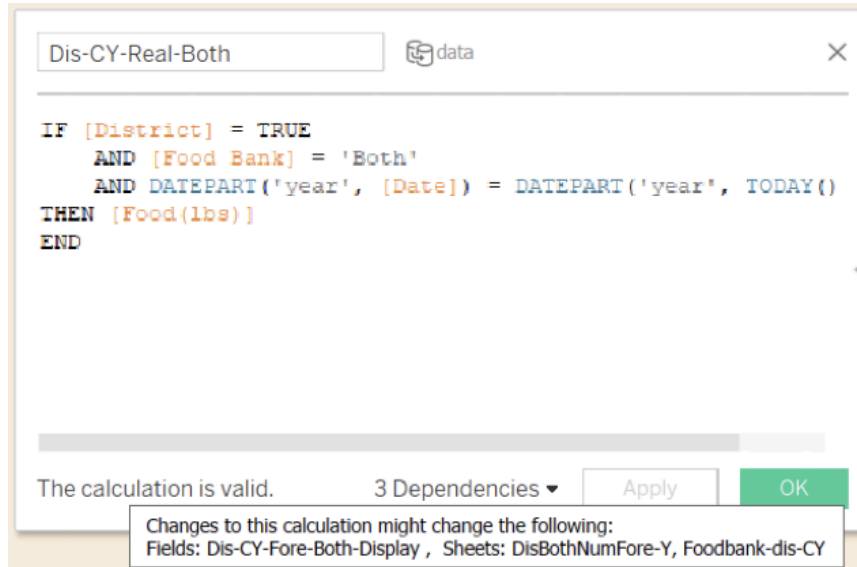
```

IF [District] = TRUE
  AND [Food Bank] = 'Jacobs & Cushman SDFB'
  AND DATEPART('year', [Date]) = DATEPART('year', TODAY())
THEN [Food(lbs)]
END

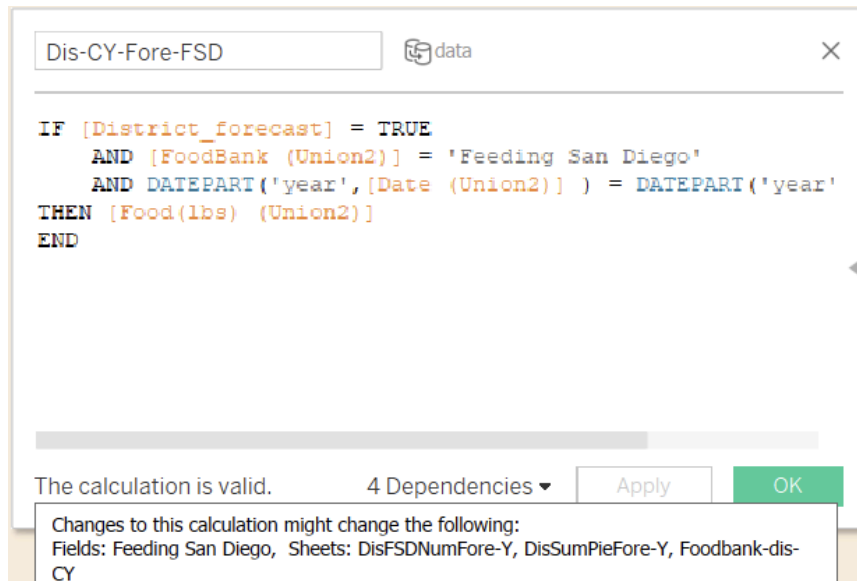
```

The calculation is valid.
4 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:
Fields: Jacob & Cushman SDFB, Sheets: DisSDFBNumFore-Y, DisSumPieFore-Y, Foodbank-dis-CY



These three fields, [Food(lbs)], are selected from 'data.csv' based on the food bank (FSD, SDFB, or both) and the current calendar year. The selection includes data from January up to the most recent month with available food delivery data.



Dis-CY-Fore-SDFB
data
X

```

IF [District_forecast] = TRUE
  AND [FoodBank (Union2)] = 'Jacobs & Cushman SDFB'
  AND DATEPART('year', [Date (Union2)] ) = DATEPART('year'
THEN [Food(lbs) (Union2)]
END

```

The calculation is valid.
4 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:
Fields: Jacob & Cushman SDFB, Sheets: DisSDFBNumFore-Y, DisSumPieFore-Y,
Foodbank-dis-CY

Dis-CY-Fore-Both
data
X

```

IF [District_forecast] = TRUE
  AND [FoodBank (Union2)] = 'Both'
  AND DATEPART('year', [Date (Union2)] ) = DATEPART('year'
THEN [Food(lbs) (Union2)]
END


```

The calculation is valid.
3 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:
Fields: Dis-CY-Fore-Both-Display , Sheets: DisBothNumFore-Y, Foodbank-dis-CY

In these three fields, [Food(lbs)], are selected from the 'Forecast_District' union, instead of 'data.csv', based on the food bank (FSD, SDFB, or both) and the current calendar year. The selection includes data starting from the most recent forecast to December of the current calendar year.

6.2.2.17 Feeding San Diego, Jacob & Cushman SDFB, Dis-CY-Fore-Both-Display

 data ✕

$$\text{SUM}([Dis-CY-Fore-FSD]) + \text{SUM}([Dis-CY-Real-FSD])$$


The calculation is valid.

3 Dependencies ▼

Apply

OK

Changes to this calculation might change the following:
Sheets: DisFSDNumFore-Y, DisSumPieFore-Y, Foodbank-dis-CY

 data ✕

$$\text{SUM}([Dis-CY-Fore-SDFB]) + \text{SUM}([Dis-CY-Real-SDFB])$$

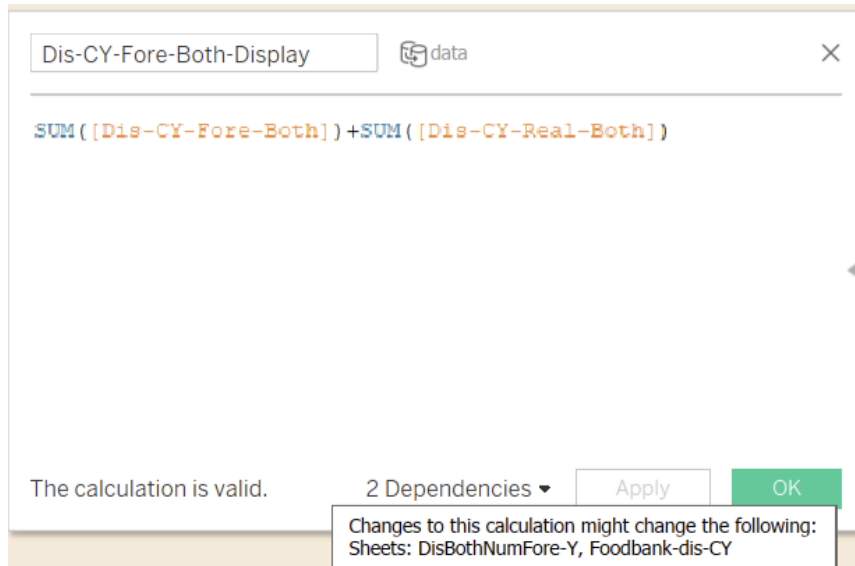
The calculation is valid.

3 Dependencies ▼

Apply


OK

Changes to this calculation might change the following:
Sheets: DisSDFBNumFore-Y, DisSumPieFore-Y, Foodbank-dis-CY



These fields are designed to present the calendar year food forecast for each food bank in the 'Foodbank-dis-CY' dashboard. They accomplish this by combining the actual delivery data from 'data.csv' for each month with the corresponding forecasted data from the 'Forecast_District' union. This integration allows for the comprehensive display of the food forecast from January to December for each food bank.

6.2.2.18 District_forecast, District_trend



×

`[CongressionalDistrict] = [sync dis]`

The calculation is valid.
25 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:

Fields: Dis-CY-Fore-Both, Dis-CY-Fore-Both-Display, Dis-CY-Fore-FSD, Dis-CY-Fore-SDFB, District_forecast & District_trend (Combined), ...and 2 more, Sheets: DisBothNumFore-FY, DisBothNumFore-Y, DisBothNumFore-YM, DisFSDNumFore-FY, DisFSDNumFore-Y, ...and 13 more


×

`[CongressionalDistricts (Union2)] = [sync dis]`

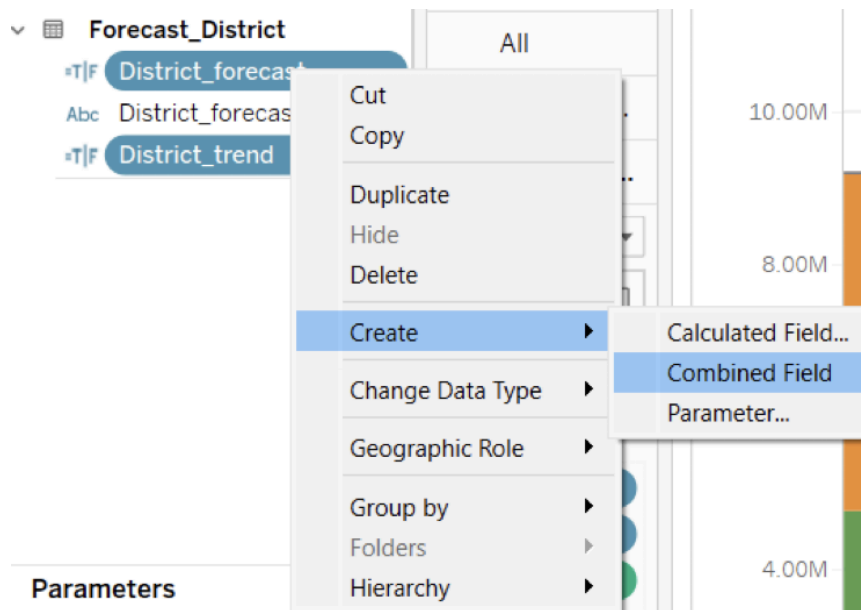
The calculation is valid.
7 Dependencies ▼
Apply
OK

Changes to this calculation might change the following:

Fields: District_forecast & District_trend (Combined), Sheets: DisTrend-CY, DisTrend-FY, DisTrend-YM, Foodbank-dis-CY, Foodbank-dis-FY, ...and 1 more

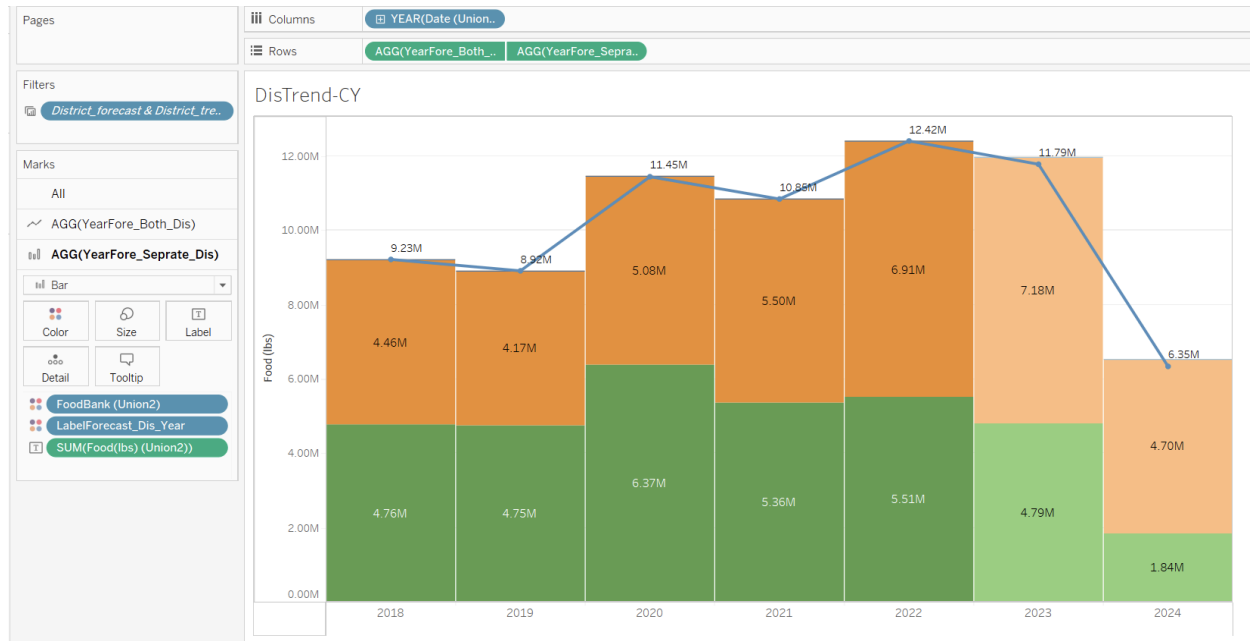
The 'Forecast_District' union merges data from 'data.csv' and 'data_forecast_district.csv'. The [CongressionalDistrict] column represents the district in 'data_forecast_district.csv', while [CongressionalDistricts (Union2)] represents the district column of the combined union. These two fields are created to synchronize district selection. This synchronization could be avoided if the district column names were consistent between 'data.csv' and 'data_forecast_district.csv'.

To ensure that both real and forecasted data are visible in the plot, a combined field is created using the following steps. This combined field is then applied as a filter in the 'DisTrend-CY', 'DisTrend-YM', 'DisTrend-FY' sheets.

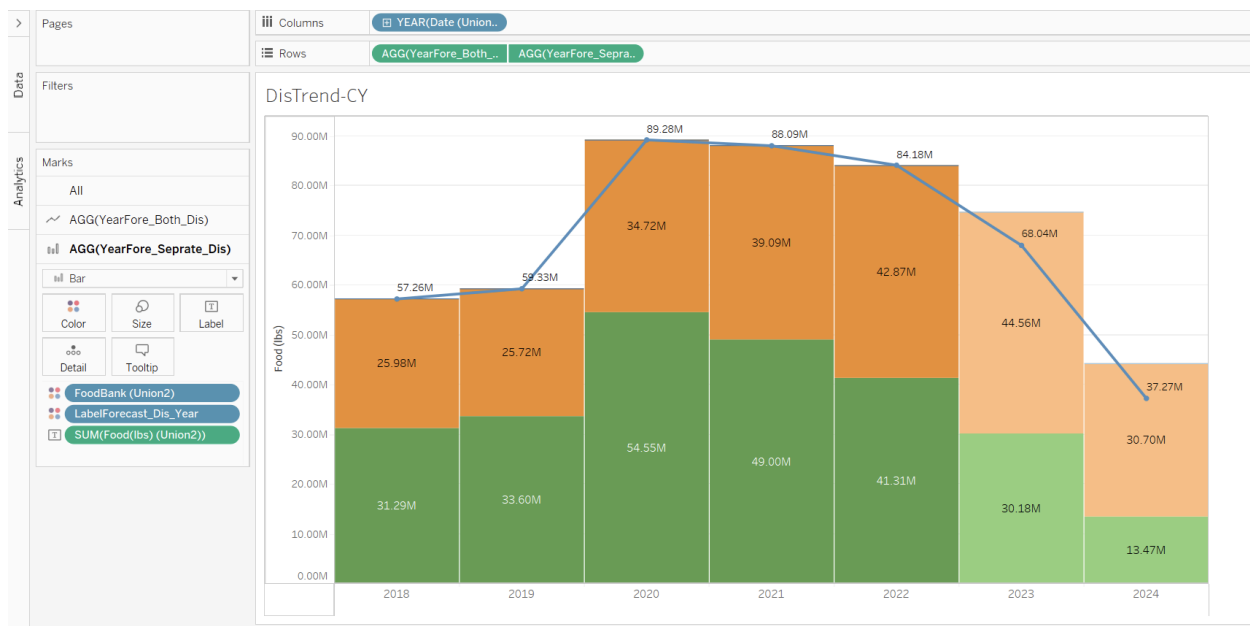


Comparing the plot with and without filter, we could see if we don't apply the filter, the plot will plot food (lbs) from all districts instead of the selected district.

With combined fields filter:



Without combined fields filter:



Sections 6.2.2.15 to 6.2.2.18 were collectively created to address the display issue mentioned in the parameter section. This display issue did not occur at the county, region, and zip code levels because the column names remained consistent. It is possible that this issue could be avoided

by modifying the code responsible for generating the 'data_forecast_district.csv' file to ensure that the district column's name aligns with the column names in both 'data.csv' and 'data_district.csv'.

7 Modeling and Analyses

This section provides an in-depth look at the modeling and development processes central to our project, aimed at addressing food insecurity in San Diego County. It encompasses a detailed examination of our forecasting model using the Prophet algorithm, a versatile tool for predicting future food bank distribution, and an exploration of our food insecurity estimation model, inspired by Feeding America's methodology. This section also defines "hunger hot spots" by employing specific metrics to identify areas most impacted by food insecurity, thereby guiding targeted intervention strategies. These components collectively demonstrate our commitment to leveraging advanced data analytics for effective planning and decision-making in combating food insecurity.

7.1 Forecasting model

We use the Prophet algorithm to predict food banks food distribution in the coming months and year. Developed by Facebook's Core Data Science team, Prophet is a user-friendly tool that works with time-based data to generate forecasts using historical information. It is available in both R and Python programming languages.

Prophet excels in handling non-linear trends and seasonal patterns such as yearly, weekly, and daily fluctuations, as well as the impact of holidays. It is particularly useful for data with strong seasonal effects and a few years of historical data. The model also allows for the use of exogenous variables that may influence the behavior of the variable of interest. Prophet can cope with missing data, shifts in trends, and typically deals well with outliers. The tool offers a fast, fully automated forecasting process that can be fine-tuned by data scientists and analysts. Prophet allows users to modify and improve forecasts using human-understandable parameters, making it adaptable and powerful for generating accurate predictions.

7.1.1 Methodology

Our methodology revolves around the use of the Prophet algorithm for predicting food bank distribution over the next 16 months, based on historical data starting in January 2018 and ending in the most recent available month. This time frame ensures a comprehensive view of past trends and seasonal variations. The model is designed to produce estimates for different geographic levels: zip code, city, congressional district, and county. Each food bank's data (Feeding San Diego, Jacobs & Cushman San Diego Food Bank, and their combined data) is

analyzed separately to capture distinct distribution patterns. We set a threshold for model accuracy, considering only those forecasts with an R-squared higher than 0.5 and excluding areas with no distribution in the past 12 months. Furthermore, we include external factors in the forecast analysis considering the effects that food distribution related events and changes in economic conditions may have in the behavior of the food insecurity landscape in San Diego County.

7.1.2 Feature creation/extraction

To account for external factors affecting food distribution, we tested various variables, including unemployment rate, monthly inflation, rent prices, electricity prices, gasoline prices (all at the county-level), and special events like the Food Banks' Together Tour and the Farmers to Families Program during the Covid pandemic. Our final model includes unemployment rate, monthly inflation, and dummy variables for the Together Tour and the Farmers to Families Program.

Prophet requires users to provide forecasts of external variables used in the model to predict the dependent variable. We use a 12-month moving average to forecast unemployment rate and monthly inflation.

7.1.3 ML model

The machine learning model, built using Prophet, incorporates both time series data and additional regressors. We add unemployment rate, CPI percentage change, and dummy variables for special events as regressors. The model is configured to capture the core time series components such as trend and multiple seasonalities.

The Prophet model is configured with parameters that align with the data's nature, ensuring the model accurately captures underlying patterns and anomalies in the dataset.

7.1.4 Model training/implementation

The implementation of the forecasting model in the "Generate_Forecast.py" script follows a structured and methodical approach, encompassing several key steps.

Initialization of Class and Variables:

- The "Forecasting" class is initialized with dictionaries defining file levels ("zipcounty": ["DATA", "ZIP_COUNTY"], "city": ["DATA_BYCITY", "CITY"], "district": ["DATA_BYDISTRICT", "DISTRICT"]) and food banks ("fsd" :["FSD_FOOD", "FSD Total"],

"sdfb":["SDFB_FOOD","SDFB Total"], "both": ["BOTH_FOOD","Both Total"]), setting the foundation for data processing and model building.

Regressors Function:

- This function starts by reading unemployment and CPI data.
- It checks for consistency in data frame sizes and merges these datasets on the 'Date' column.
- A 12-month rolling average for unemployment rate and CPI is calculated for the specified number of forecast months.
- Monthly percentage changes for CPI are computed and adjusted.
- Pre-set dummy variables for specific events "Together Tour" and "Farmers to Families" are added to the regressors.
- The resulting regressors data frame is saved for future use.

Food Banks Forecast Workload Function:

- Data from food banks is imported and merged with the regressors data.
- A final table is initialized to store the forecast results.
- The Prophet model is set up for each location (zip, city, district) within the food banks' data.
- Regressors are added to the model.
- The model is trained, and forecasts are generated.
- The model's fit is assessed using the R-squared metric, and forecasts are only considered if they meet the threshold.
- Negative forecast values are adjusted to zero.
- The aggregated forecast for each food bank is added to the final table.
- The final forecast data is saved as a CSV file.

Creating Forecasts for Different Geographic Levels:

- Separate functions are called to create forecasts for different levels (zip/county, city, district) for each food bank.
- These functions loop through the food bank sheets corresponding to each geographic level and apply the forecasting workload function to generate and save the forecasts.
- Throughout this process, the script uses a combination of data manipulation, model building, and forecasting techniques to create a comprehensive view of future food bank distribution across various geographic levels.

7.1.5 Model evaluation/Accuracy

Model performance is evaluated using the R-squared metric to assess the fit of the model to the historical data. Forecasts are only considered reliable if they achieve an R-squared value above 0.5. We also perform a cross-validation analysis to assess the model's predictive accuracy, ensuring that the forecasts are not only fitting the historical data but also capable of predicting future food distribution needs accurately. The results are then scrutinized, and adjustments are made as necessary to improve the model's accuracy.

7.2 Food insecurity estimation model

Our estimation of the food insecure population is based on Feeding America's methodology, which considers local factors such as Poverty Rate, Unemployment Rate, and demographic data. We first analyze the relationship between these variables and food insecurity rates across US states over several years. Then we use this analysis to estimate food insecurity rates at the zip code level. By multiplying these rates with the population of each location, we calculate the number of food insecure people in each zip code.

7.2.1 Methodology

Following Feeding America's approach, we use a two-step process to estimate food insecurity in each zip code:

State-Level Data Analysis: Using state-level data from 2009-2019, we estimate a model where the food insecurity rate is determined by Poverty Rate, Unemployment Rate, Share of African Americans, and Share of People with Disability. Different from Feeding America's work, our analysis does not include Median Income, Share of Hispanics, and Share of Homeowners since the coefficients of these variables were found not statistically significant when we used all variables in the estimation. Our model accounts for year and state fixed effects to control for unobserved state-specific and year-specific influences on food insecurity. State-level food insecurity estimates are obtained from Feeding America and the other variables come from the US Census Bureau's American Community Survey. The regression equation used is:

$$FI_{st} = \alpha + \beta_{UnempRate} UnempRate_{st} + \beta_{PovRate} PovRate_{st} + \beta_{AfricanAm} AfricanAm_{st} + \beta_{Disability} Disability_{st} + \mu_t + \nu_{st} + \epsilon_{st},$$

where FI represents food insecurity rates, β_i are the coefficients for each independent variable, μ and ν are the fixed effects, and ϵ is the error term.

Zip Code-Level Estimation: Using the coefficient estimates from Step 1 and zip code-level data, we generate estimated food insecurity rates for individuals at the zip code level. This data includes 2017-2021 ACS 5-year estimates for zip codes and the latest available monthly zip code unemployment estimates from the San Diego Association of Governments (SANDAG). The zip code-level estimation is calculated as:

$$FI_{zipcode} = \sum(\beta_i \times X_{zipcode,i}),$$

where $X_{zipcode,i}$ represents the extracted features for each zip code.

From this estimation, we calculate both food insecurity rates and the number of food-insecure persons in each zip code. This methodology provides a comprehensive understanding of food insecurity rates across different zip codes, considering various socio-economic factors that influence food insecurity.

7.2.2 Feature creation/extraction

For the state-level analysis, we extract demographic and economic variables from the ACS, unemployment rates from state datasets, and food insecurity rates from Feeding America. For the zip code-level estimation, we utilize the ACS 5-year estimates and monthly unemployment data from SANDAG. These features are crucial for accurately reflecting the socio-economic conditions influencing food insecurity.

7.2.3 Regression model

The model is structured as a fixed-effects panel regression, accounting for both individual state characteristics (state fixed effects) and time-specific influences (year fixed effects). This structure allows us to isolate the impact of our key variables on food insecurity while controlling for other unobserved factors.

7.2.4 Model evaluation/Accuracy

The model's accuracy is evaluated based on the statistical significance of the coefficients and the overall fit of the model, as indicated by R-squared values.

7.3 “Hunger hot spots” definition

In our project, we define "hunger hot spots" using two specific metrics: "Meals (including CalFresh) per Population in Poverty (150% FPL)" and "Meals (including CalFresh) per Food

Insecure Population". These metrics help us understand the average number of meals received by individuals experiencing food insecurity in a particular zip code, whether it be over a month or a year.

Recognizing that our food insecurity estimates are on the conservative side, we also use the population living below 150% of the Federal Poverty Level (FPL) as an additional indicator of food insecurity. This approach is particularly relevant in San Diego County, where the cost of living exceeds the national average, making it a crucial factor in assessing food needs.

The main challenge of this analysis is to combine two different measures of food assistance—pounds of food from the food banks and monetary benefits from CalFresh—to understand how well a community is being served. The solution is to convert both measures to meal equivalents. For the pounds of food distributed, we follow Feeding America's guidelines and equate one meal to 1.2 pounds of food.

To convert CalFresh issuances we need to consider the average cost of a meal in San Diego County. We rely on the most recent data from Feeding America's Map the Meal Gap project and adjust it monthly to account for inflation. This ensures that our measurement of meals distributed remains accurate and reflective of current economic conditions, thereby enabling a more precise identification of hunger hot spots within the county.

The definition of "hunger hot spots" in our project is a meticulous process that merges quantitative data with qualitative understanding of local economic conditions. Our approach, which considers the higher cost of living in the county and combines different forms of food assistance into a unified measure of meal equivalents, reflects a commitment to accuracy and relevance in identifying areas most in need of intervention. This comprehensive analysis not only highlights the regions most affected by food insecurity but also guides targeted efforts to alleviate hunger in these critically identified hot spots.

8 Monthly updates

The Food Bank Dashboard is updated every month with the most recent data available. Below is a list with the steps taken to implement the update, from data request to dashboard upload. As a general rule, the names and addresses of the food banks' partner agencies are updated every first month of each quarter.

On the 15th day of the month:

- Ask collaborators for the most recent data:
 - Feeding San Diego (point of contact: Lisa Freedman - lfreedman@feedingsandiego.org)

- Jacobs & Cushman San Diego Food Bank (point of contact: Amy Eilts - aeilts@sandiegofoodbank.org)
- SDGE/LIHEAP (point of contact: Eddie Tabornal - ETabornal@sdge.com)
- County/CalFresh (point of contact: Gloria Garcia - Gloria.Garcia3@sdcounty.ca.gov)

Once all data is received, preprocess files in the Food Bank Project's folder in Google Drive:

- In the Jacobs & Cushman San Diego Food Bank's file (inside folder "Jacobs & Cushman SD FB - Data"):
 - Delete the columns 'Diapers' and 'Period Supply'
 - Then recalculate the sum of all columns
- In Feeding San Diego's file (inside folder "Feeding SD - Data"): delete blank columns and rows
- Download 'unemployment_wide.csv' from https://opendata.sandag.org/Economy/Unemployment_Wide/wr4p-hajy and save it to the folder "data/SANDAG_UNEMPLOYMENT"
- Update CalFresh file name to represent actual month of data (inside folder "CalFresh - Data > Raw Files")
- Create new CalFresh file using only the edited table of the first sheet. 'Zip Code', '[month] [year]', and 'Grand Total' columns. Name the sheet: "Total Issuances with Emergency". Add new file to "CalFresh - Data > Ready Files".
- Push to GitHub:
 - Feeding San Diego: new monthly file in "data/FEEDING_SD_FOOD/FEEDING_SD_FOOD_RAW/"
 - Jacobs & Cushman San Diego Food Bank: new monthly file in "data/SDFB_FOOD/SDFB_FOOD_RAW/"
 - SDGE/LIHEAP: new monthly file in "data/LIHEAP/LIHEAP_2023/"
 - County/CalFresh: new monthly file in "data/CALFRESH/CALFRESH_HISTORICAL/"
 - Unemployment_wide: updated file in "data/SANDAG_UNEMPLOYMENT/"

In the GitHub repository:

- Update the changeable variables at the top of "main.ipynb" (inside folder "pipeline/")
- Run "main.ipynb" cells [1] through [5]
- In the folder "data/FOOD_INSECURITY/DATA_FOR_ESTIMATION/", run "fi_estimation.R"
- Run the remaining cells in main.ipynb
- Update 'data_info2.csv' → Update 'Period' and 'Last Update'
- Update 'data_info3.csv' → Update 'Period' and 'Last Update'
 - For prices and unemployment, check <https://fred.stlouisfed.org>

- For CPI, check https://www.bls.gov/regions/west/news-release/consumerpriceindex_sandiego.htm
- Push changes to GitHub

In Tableau Desktop:

- Open latest workbook version from GitHub
- Save as 'maindashboard_v4_[current date]'
- Edit data connection to update data files -> Select live data
- Update 'LabelForecat_Reg_Month' -> Update month: add 1
- Update 'LabelForecast_Reg_Year'
- Update 'LabelForecast_Dis_Month'
- Update 'LabelForecast_Dis_Year'
- Update 'LabelForecast_Zip_Month'
- Update 'LabelForecast_Zip_Year'
- Update 'Last Update' date
- Update SD Tab tooltips
 - CalFresh
 - LIHEAP
 - Unemployment Rate
- Extract all data
- Upload final version to Tableau Desktop

In the GitHub repository:

- Push final workbook version

Finally, update DSA's website.

9 Responsible Data Science principles application

In our project, we have diligently applied the principles of responsible data science — Fairness, Veracity, Transparency, and Privacy — to ensure ethical and effective use of data in addressing food insecurity in San Diego County. The following outlines examples of how each principle has been integrated into various aspects of our project:

Fairness:

- **Understanding operations:** By volunteering at the food banks, we gained firsthand insight into their operations, enabling us to ensure that our analyses fairly represent the realities of food distribution and need.

- **Collaboration with experts and stakeholders:** Working alongside data science luminaries and keeping regular meetings with executives from both food banks has helped us refine our models to prevent biases and promote fairness in predicting food distribution and identifying hunger hot spots.

Veracity:

- **Collaboration with stakeholders:** Collaborating with both food banks and holding regular check-ins has also ensured the accuracy and relevance of our data. This continuous engagement allowed us to better understand the use of the dashboard and deliver improvements, ensuring the veracity of our work.
- **Collaboration with experts:** Experienced data science professionals have also helped us build and test our models, certifying that our work was accurate.
- **Datasheet:** We maintained a comprehensive datasheet to track the data used in the project, ensuring the veracity and credibility of our data sources.
- **Documentation:** We documented all the steps of the project development process, including the data collected, the methodologies of the models, and technology used to build the dashboard. This procedure guarantees the reproducibility of the project.

Transparency:

- **Datasheet and documentation:** Creating a datasheet and a thorough documentation is also important to provide transparency to the stakeholders.
- **Info tab development:** The development of the "info tab" on our dashboard is a direct result of our commitment to transparency. It provides users with clear information on data sources, methodologies, and the assumptions behind our models, making our processes open and understandable to all stakeholders.
- **Regular Updates:** Keeping stakeholders updated through regular reports and well-documented methodologies has been crucial in maintaining transparency throughout the project lifecycle.

Privacy:

- **Securing collected data:** We have taken rigorous measures to secure the data collected. Data is stored in secured environments, and access is strictly controlled to protect the privacy of individuals and organizations involved.
- **Ethical data handling:** All data handling procedures, from collection to processing, have been conducted with the utmost respect for privacy concerns.

By embedding these principles into the core of our project, we have strived to not only achieve our objective of combating food insecurity but also to set a standard for responsible and ethical data science practices in community-oriented projects.

10 Impact and future directions

Our project has made significant strides in addressing food insecurity in San Diego County. By aggregating the data from both food banks and applying data science techniques to predict food distribution and identify hunger hot spots, we have enabled local food banks to optimize their operations and reach those in greatest need more efficiently. The insights gained from our models have informed strategic decision-making, leading to more targeted and effective food distribution efforts. Furthermore, our commitment to responsible data science has fostered trust and collaboration among stakeholders, enhancing the overall impact of the project.

Key impacts include:

- Improved targeting of food assistance to areas identified as hunger hot spots.
- Enhanced understanding of the dynamics of food insecurity within the county.
- Strengthened partnership between both food banks in San Diego County.

Looking ahead, there are several avenues for expanding and enhancing the project:

- **Dashboard upgrade:** Create an executive summary tab displaying key metrics and relevant analyses.
- **Integration of additional data sources:** Incorporate new data sources such as CALPADS to improve our understanding of the need for food assistance at a more granular level.
- **Forecasting hunger hot spots:** Develop a model to forecast the demand for food assistance at the zip code level to help food banks proactively address needs.
- **Neighborhood clusters:** Incorporate new clusters of food bank activity combining neighborhoods with similar levels of poverty.
- **Enhancing model accuracy:** Continuous refinement of our models to increase accuracy and lead to more precise predictions and interventions.
- **Expansion to other regions:** Building on the success in San Diego County, there is potential to apply our models and methodologies to other regions, adapting to local contexts and data availability.

In summary, this project has not only provided immediate benefits in terms of improved food distribution strategies but also laid a foundation for future initiatives aimed at eradicating food insecurity. By continuing to leverage data science responsibly and innovatively, we can broaden



DSA Food Banks Project Documentation

Version 1
February 4, 2024

our impact and contribute to lasting change in communities both within and beyond San Diego County.