Name of the group members and CNetIDs

Anthony Hakim (anth0nyhak1m)
Marc Loeb (mloeb)
Sasha Filippova (sashaf)
Yifu Hou (yifu)

A brief overview of the final project (200 words maximum)

Our project is dedicated to the housing geography of Chicago, and the socioeconomic factors that can account for its structures.

We take advantage of the city of Chicago's detailed database of building permits issued since 2006. Specific permits are required for a range of construction activities, including renovations, electrical installations, signage, and adding a porch to an existing structure. We narrow in on two permit categories ("WRECKING/DEMOLITION" and "NEW CONSTRUCTION"). When aggregated to the level of Chicago's 77 "Community Areas" (a standard data gathering spatial unit employed by the city) we believe that these provide a benchmark for estimating neighborhood decay and change. Certain (considered to be rapidly developing or gentrifying) such as Woodlawn and the Near North Side exhibit as many as fifteen times more new buildings than demolition. The opposite is true in community areas like Englewood.

To contextualize this metric, we retrieve a range of potential explanatory variables from the Chicago Open Data Portal and the Illinois Data Portal, including data on demographics (race, population), crime (homicides) and local commerce (grocery and liquor stores). Our central metric, and these socioeconomic variables, are all mapped via a Dash dashboard. This allows the user to explore the data for relationships.

The overall structure of the software (1-page maximum). It would be nice to include a helpful diagram of how the modules are connected with each other but this is not required.

build_dataframes.py (Marc, Sasha, Anthony)

This is the primary file of our package. We import and run functions from our util.py file, and other files and directories in build_dataframes.py, in order to assemble our DataFrame for display in Dash, and create a CSV file for fast deployment. Marc provided the GeoDataFrames, which include permits counts and demographics, and is imported and converted into regular dataframes. Other dataframes from util.py are also imported and converted, which were created and merged by Sasha and Anthony. Furthermore, the

5 initial dataframes are joined together to create our grand dataframe to be fed into the Dash app. This process involved further transforming the data for it to be joined on the same type of columns.

util.py (Sasha, Anthony)

Primarily, interacts with the Chicago Open Data Portal API - pulls 3 main datasets from the API: Crime, Socioeconomic Indicators, and Grocery stores. Uses pandas to clean the data grabbing the columns that are necessary for our display from our 3 large initial dataframes, and then transforms the data into panel form for the possibility of displaying different years for different variables in the Dash App.

building_permits.py (Marc)

This file contains a single function, which downloads and filters Chicago's database of building permits. It geocodes the records to fill in missing longitude and latitude data, and save the resulting GeoDataFrame as a geoJSON locally.

It has an optional testing parameter (set to false by default) which activates the testing mode.

geo_comm_areas.py (Marc)

This file contains a series of six functions (and one helper). Cumulatively, they download the geoJSON of Chicago's 77 Community Areas as a GeoDataFrame; retrieve the saved permits.geojson file (also as a GeoDataFrame) and conduct a spatial join to fill in the missing data for building permits without community area number; join the counts of new buildings and demolitions—and a sum of new building value—to the community areas GeoDataFrame; use an additional API to retrieve community area census data from Illinois.gov on race, population and housing vacancy; find the per capita number of buildings and demolitions and per capita new building value for each community area; and create a trio of time-series GeoDataFrames for building permits, demolitions, new building value per community area per year (for the years between 2006 and 2021).

build_map.py (Yifu)

build_map.py is the Python Dash file for data visualization and user interaction.
The file is built based on two major databases. comm_area: a GeoJson dataset generated from its own API and stored locally, containing the coordinates for 77 polygons of 77

community areas in Chicago city; and df: a joint data frame for all the information on Chicago Data Portal, read in from data processing module.
The module provides a chain dropdown altered for the user to build visualization on one particular Chicago data. The user can choose the type of data, a specific year and/or a specific group inside the data. Based on the data of choice, the module provides a colored choropleth Chicago map presenting data on each community area.

A description on the code responsibilities for each group member (i.e., who was responsible for what module, files, tasks, etc.).

Marc took the lead on wrangling and processing data pertaining to Chicago's database of building permits.

He signed the group up for a Socrata API key, and wrote the code for the geocoding and spatial joins needed to fill in the missing community area data for Chicago's building permits. He was also responsible for retrieving the census data supplement for the community areas, calculating the counts of new buildings and demolitions (and the cumulative new building value) per capita for each community area, and creating the time series GeoDataFrames for new buildings, demolitions, and new building value per community area per year.

He also wrote the argparse to manage the testing code, the project description for this document, and the project diagram.

Sasha and Anthony took the lead on wrangling and processing crime, grocery stores, and socioeconomic dataframes, initializing the virtual environment, managing virtual environment dependency requirements, and managing application architecture in the virtual environment to ensure the application runs smoothly.

Sasha and Anthony created the files util.py and build_dataframes.py. Anthony focused on the prototyping of grabbing, cleaning, and merging data for util.py and build_dataframes.py. Sasha focused on creating concise and efficient code in the two functions. These functions extract multiple pieces of data from the Chicago Data Portal API and the previously created GeoDataframes, and then cleans and transforms the data so that they are compatible for merging. Build_dataframes takes the already cleaned and transformed data and merges them on the columns to produce a grand dataframe to be fed into the Dash application. The build_dataframes function also uses columns to produce calculations of different ratios involving building to demolition, liquor stores to grocery stores. The demographics percentage was grabbed from the census data and not manually calculated like the others. To ensure a smooth application running process, Anthony took on the responsibility of creating the requirements.txt file, and placing the bash script file "install.sh" (gifted by Lamont) in the project so that a new virtual environment will be created with the necessary package dependencies already installed. Furthermore, Sasha took the lead on ensuring the path architecture runs smoothly on the virtual environment.

Yifu took the lead on visualizing our data via an interactive dashboard

He retrieved Chicago community area GeoJson data with API, designed the structure of the interface based on the two datasets. For user interaction, he applied a flexible Dash dropdown menu that can adjust options based on different user inputs. For choropleth map, he coded with package plotly.express to visualize community area map on different data options for 4 callbacks.

Short description on how to interact with the application and what it produces.

To interact with the map, a user will select from type of data, year, to specific target for visualization. The tree structure for selection is:

```
----- [Type of Data] Demography
          ----- [Race) Asian, Black, Hispanic, White
----- [Type of Data] Building Permits
          ----- [Year] 2001 – 2021
                    ----- [Permits] New, Demolished, Ratio of New / Demolished
----- [Type of Data] Crime: Homicide
          ----- [Year] 2001 – 2021
----- [Type of Data] Local Commerce
          ----- [Permits] Grocery, Liquor, Ratio of Liquor / Grocery
```

After selecting a specific target, the choropleth map will produce a Chicago City map of 77 community areas, on which the value of the target index on each community area is distinguished by the shade of color, and community area name, year, and value of the index are presented as hover data on each community area.

The user can check and compare different categories of data and have a glance at the general development in Chicago city over time, as well as cross comparing different community areas. (For example, the number of new building permits before and after 2008 are visually different on our map, the density of black and white community is clearly separated on the south and north sides of Chicago, homicide occurs in southern Chicago more often, but it is very dense in Austin).

What the project tried to accomplish and what it actually accomplished (200 words)

Our ultimate goal of Working with Chicago's database of building permits was met. Our Final product is a map dashboard intended for data exploration and pre-analysis, which can be built upon for researchers who are interested in gaining an intuitive idea of the relationship between urban development and the various socioeconomic indicators.

Our initial intention was to conduct regression analysis. We hoped to run basic predictive modeling to forecast what areas of Chicago were set to grow (more permits for new construction) or decay (more demolition permits), based on various socioeconomic indicators (race, crime, proximity, grocery stores, etc.) Working with incomplete spatial data for the building permits meant that a substantial chunk of the quarter was dedicated to setting up a pipeline for geocoding and spatially joining the records.

Our structure shifted from an object oriented design to a design based around geopandas, and dataframes, which we feed into a Dash dashboard.

Ultimately, this project accomplishes the goal of formatting complex spatial data with large dataframes in a suitable manner ready to be used for analysis by researchers. Last but not least, our data is in a suitable format to be displayed in an interactive map dashboard for exploratory data analysis.