Geocoding Analysis

!!!DO NOT UPLOAD!!!

Description

This project is designed to take the open-source dataset for the Chicago Medical Case Examiner's Notes (here) and combine it with the openly available Chicago Land Use datasets (here) to analyze opioid death trends by Land Use area and geographic regions.

API Reference

An API Reference for each module is available here.

Requirements

This project assumes you are running on a unix environment and have:

Software	Version
Python	>=3.8
Poetry	>= 1.0
R	>= 4.1

> *Some package dependencies may break on ARM based processors (like 1 Macs) as support has not yet arrived. It is best to use a Linux-environment in these cases. Our pipeline is run using Arch. It is also required to have `make` and `unzip` installed, but these come pre-installed with most UNIX-based OSes.

Installation

Getting this project up and running locally is a multi-step process. First, you can clone the git repository onto your local machine and change into that directory.

```
git clone https://github.com/UK-IPOP/geocoding.git
cd geocoding
```

Then you can install the projects main dependencies: `poetry install --no-dev` or install all dependencies (including development) using `poetry install`. Then activate the poetry created virtual environment by running `poetry shell`.

Now you're code environment is ready.

Methodology

In order to perform this analysis, we ran the pipeline on 08-24-21 and used the resulting data-file for analysis in **[PAPER]**. To see an example of potential analysis that could result from this, see our Tutorial Notebook.

Pipeline Explanation

The pipeline has multiple stages that need to run in succession.

1. Download the raw Land Use files and apply the Data Dictionary (extracted from this PDF) to each Land Use polygon to create new shapefiles.
2. Geocode the pharmacies provided by Cook County.
3. Download and geocode Case Archive records.
4. Calculate the distance to the closest pharmacy from each Case Archive record.
5. Spatially join the Land Use shapes to the Case Archive records giving each record a corresponding Land Use category.
6. Spatially join the Census Tract shapes to the Case Archive records.
7. Spatially join the Parks shapes to the Case Archive records identifying Park locations.
8. Extract drug names and classifications (i.e. fentanyl or non-fentanyl) from `primary_cause` and `secondarycause` fields in data.
9. Merge extracted drugs dataset into spatially-joined dataset for final output.
10. Cleanup dataset, add hotel/motel, hot/cold, death_date-related columns.
11. Cleanup files.

> *These records will be geocoded which can take hours so it is recommended to pre-filter your data using the Cook County Open Data explorer and then modify the pipeline as needed if you do not require **all** of the records.

The following image may more clearly explain the data flow:

Pipeline Flowchart

Pipeline Flowchart Image

Pipeline Usage

Running the pipeline is simple once you have everything setup.

Inside the home directory of the geocoding project simply run: `make pipeline`

This can take up to 10 hours depending on the system you are running and the number of ME records you are geocoding.

If you wish to remove any steps from the pipeline those lines can simply be removed from the `Makefile`

Additionally, sometimes the pipeline may stall (ArcGIS sometimes times-out) at which point you can simply resume running the commands sequentially found in the `Makefile`. Each command takes roughly 5-10 minutes *at most*, with the exception of the Case Archives geocoding which takes a few hours.

Pipeline Improvements & Contributing

Currently, there are two main problems with the pipeline:

1. It requires two languages to run.
2. Some processes take a *very* long time.

The first of these can be solved simply my migrating the singular R script into a Python script.

The second is an issue of performance and comes down to two main culprits. The first is the geocoding of the case archives and the pharmacies. This is limited by web requests to the ArcGIS geocoding service. Asynchronous support proved useful to speed up this process and provided a 40% reduction in speed, but the 50,000+ web requests still takes a few hours to complete on the ArcGIS server. The second culprit is the distance calculation which runs on all of the case_archives and all of the pharmacies. This ends up being almost 100 million iterations and is CPU limited. Multiprocessing could speed up this process.

Feel like contributing? See Contributing. 😃

Data Dictionary

This table shows the columns that we add for analytical purposes and their definitions. You can see the Land Use data labels here and the Case Archives here under 'Columns in this Dataset'.

New Columns

This has been moved to an online spreadsheet (for now). A markdown table will be generated and inserted below before pushing this file to GitHub.

Column Name	Data Type	Description
geocoded_score	float	The returned ArcGIS confidence score if re-geocoding was performed. Scales from 0-100; 100 being most confident.
geocoded_address	string	The returned ArcGIS street address if re-geocoding was performed.

Column Name	Data Type	Description
full_address	string	The concactenated address from source data incident address fields. This also includes some preprocessing (removing special characters) and normalization (lowercasing).
recovered	integer	Whether there was recovered geocoding. 1 if re-geocoding was performed successfully, 0 if re-geocoding was not performed.
final_latitude	float	The decided latitude. If recovered = 1, then this is the ArcGIS returned (re-geocoded) latitude; otherwise it is the source latitude.
final_longitude	float	The decided longitude. If recovered = 1, then this is the ArcGIS returned (re-geocoded) longitude; otherwise it is the source longitude.
closest_pharmacy	float	Distance in kilometeres from the incident location to the nearest pharmacy from the pharmacy datafile.
LANDUSE	integer	Land use id from the land use shapefile.
STATEFP	integer	State code from census shapefile.
COUNTYFP	integer	County code from census shapefile.
GEOID	float	Unique ID for census tract.
INTPLTLAT	float	Latitude from census tract. Suspect it is centerpoint.
INTPTLON	float	Longitude from census tract. Suspect it is centerpoint.
CFNAME	string	Park name from park shapefile
CFTYPE	string	Park type from park shapefile
CFSUBTYPE	string	Park subtype from park shapefile.
ADDRESS	string	Park address from park shapefile.
FNISCODE	integer	Park code from park shapefile.
SOURCE	string	Source from park shapefile.
Jurisdicti	string	Jurisdiction from park shapefile.
Community	string	Community from park shapefile.
landuse_name	string	Label for LANDUSE category.
landuse_sub_name	string	Minor category label for LANDUSE.
landuse_major_name	string	Major category label for LANDUSE.
death_datetime	datetime	Datetime stamp of recorded death.
death_time	time	Time of death extracted from death_datetime. (24 hour range)

Column Name	Data Type	Description
death_year	integer	Year of death extracted from death_datetime.
death_month	integer	Month of death extracted from death_datetime. Range 1-12.
death_day	integer	Day of death extracted from death_datetime. Range 1-31.
death_week	integer	Week of death extracted from death_datetime. Range 1-52.
motel	integer (bool)	Whether any of the keywords: "hotel", "motel", "holiday inn", "travel lodge" are found in the full_address field. 0 or 1 (True).
hot_combined	integer (bool)	Whether "hot" was found in the primary or secondary cause fields in addition to source heat_related field. 0 or 1 (True).
cold_combined	integer (bool)	Whether "cold" was found in the primary or secondary cause fields in addition to source cold_related field. 0 or 1 (True).
primary_combined	string	Concactenation of primarycause fields (main, line_a, line_b, line_c).
repeated_address	integer (bool)	Whether or not the full_address is repeated in the dataset.
address_repititions	integer	....PENDING...
repeated_lat_long	integer (bool)	Whether or not the pair of final_latitude, final_longitude repeat in the dataset.
lat_long_repititions	integer	....PENDING...
death_street	string	Street address the death occured at.
death_city	string	City the death occured in.
death_county	string	County the death occured in.
death_state	string	State the death occured in.
death_zip	string	Zip code the death occured in.
death_location	string	Generalized location the death occured in.
death_location_1	string	Roll-up location (more grouped/general than death_location) the death occured in.

In addition to this the new file has various columns regarding the Land Use categories which are self-explanatory and come from the above-mentioned dataset and this PDF.

Following those columns, there are **MANY** columns for drug extractions. These columns all have one pattern, each drug has two columns. One with the suffix "_primary" which means the value (True/False -- 1/0 -- 9 for not searchable) was extracted from the `primarycause` column, while the one with the "_secondary" suffix was extracted from the `primarycause_linea, primarycause_lineb, primarycause_linec, secondarycause` columns. Additionally, each selected drug belongs to specific categories and those categories are added and labeled (primary vs. secondary) for each record as well. For a table of drugs, search terms, and categorization, checkout our drug dictionary file.

Support

For questions on implementation or issues you can either make a GitHub Issue or contact @nanthony007

License

This project is GNU v3 Licensed which means that work you perform utilizing this project must attribute to this project (the source), you must disclose any source-changes you make, and your resulting work must also be GPLv3 Licensed.

Citation

If you use this work, please cite this repository, **insert people here**:

@@@ bibtex citation