

Sharing Code Among Academic Researchers: Lessons Learned

Carol Schmitz, Ameena Khan, and Libby Hemphill









Our Mission

Make our GitHub repositories as accessible as possible to other researchers with varying levels of technical skill

CASM Lab Best Practices for Sharing Code

- Code in a human-readable programming language – Python
- Use consistent structure to organize our public repositories
- Use separate development environments for each project and install libraries separately
- Use Jupyter Notebooks to explain how each script works, including inputs, outputs, and how to run the script

Our Repository Structure

 AmeenaKhan committed on GitHub Update everyblock_collect.py ...	
 data_samples	Raw data and data produced by scripts
 files	Any files needed or generated by the scripts
 scripts	Scripts to collect, cache, analyze, and parse data
 README.md	Description of repository, supported OS, contact information
 environment.yml	List of Python packages for Anaconda installation
 everyblock.ipynb	Jupyter Notebook with Markdown and code to explain and run scripts
 settings-example.cfg	Sample of all configurable options used by code

Challenges of Sharing Code

- Supporting multiple operating systems introduces complexity to the task, since certain libraries only exist for certain operating systems.
- Python libraries and other system packages can cause conflicts across multiple projects, but using separate environments for each project can help avoid conflicts

What's next?

- Collaborate with other researchers who share our priorities
- Design and develop a central repository for sharing data sets with accompanying analysis scripts

Our Workflow

Collect and Cache

The [EveryBlock collector](#) collects data from the Everyblock API and stores it in a text file. If the config file is filled in accurately, the command in the following cell should produce a text file that can be parsed and mapped with the other notebooks in this repo.

```
% run scripts/everyblock_collect.py
```

Parse and Add Census Tract

Step 1: Parse. The code will produce 2 tab-separated text files based on the input: one counts the number of posts from each schema type (as defined by the Everyblock API), and one provides metadata about each post.

```
% run scripts/everyblock_parse.py
```

Step 2: Add Census Tract. Takes output from Step 1 and adds Census Tract data from [the FCC](#) to each post, based on longitude and latitude. then returns the 15-digit census tract id ([Block][FIPS]) in the example below.

```
{ "Block": { "FIPS": "170311608005017" }, "County": { "FIPS": "17031", "name": "Cook" }, "State": { "FIPS": "17", "code": "IL", "name": "Illinois" }, "status": "OK", "executionTime": "190" }
```

The census ID is appended to each line and the file is saved. If no data is returned from the FCC website, "null" is appended rather than the FIPS number.

NOTE: This process is very long. It can take more than a day, depending on the response times from the FCC website.

```
% run scripts/everyblock_add_census_tract.py
```

Parse and Analyze

I prefer to start with a set of questions I want to use the data to answer and then parse it in a way that efficiently addresses those questions. But, a great first step is to get some bird's eye views of what's in the data in the first place.

Bird's Eye Views

EveryBlock allows users to post a variety of types of content (EveryBlock term: news items) that they call *schemas*. I wonder what the relative frequency of those schemas are and how those relative frequencies differ between neighborhoods and metros (EveryBlock's term for cities). To answer that question, I need something like

schema-freq.txt, a tab-delimited file:

```
Schema Freq
Announcements 12
Crime-posts 45
```

user-items-meta.txt, a tab-delimited file:

```
id pub_date poster_name schema latitude longitude reaction_count reaction_score comment_count url
7285706 2016-01-07T02:06:04.443Z ld2 talk 41.923467879795 -87.708659235119 1 3 19 http://chicago.everyblock.com/talk/jan07-everyone-alright-7285706/
```

*note: for items with multiple locations, there should be a line for each location

Files for Dedoose

We want one text file per neighborhood, sorted by schema, that contains the following data for each user-generated post:

- Location Name
- Title
- item date
- attribute>comment
- embed>description
- embed>url
- reaction count
- schema

```
% run scripts/everyblock_parse_to_dedoose.py
```