7-21-2014

Processing Government Data: ZIP Codes, Python, and OpenRefine

Frank Donnelly
CUNY Bernard M Baruch College

code{4}lib JOURNAL

Issue 25, 2014-07-21 ISSN 1940-5758

Processing Government Data: ZIP Codes, Python, and OpenRefine

While there is a vast amount of useful US government data on the web, some of it is in a raw state that is not readily accessible to the average user. Data librarians can improve accessibility and usability for their patrons by processing data to create subsets of local interest and by appending geographic identifiers to help users select and aggregate data. This case study illustrates how census geography crosswalks, Python, and OpenRefine were used to create spreadsheets of non-profit organizations in New York City from the IRS Tax-Exempt Organization Masterfile. This paper illustrates the utility of Python for data librarians and should be particularly insightful for those who work with address-based data.

By Frank Donnelly, Geospatial Data Librarian, Baruch College CUNY

1 Introduction

One aspect of a data librarian's mission is to help make datasets more accessible to users, which is not always a simple matter of teaching users how or where to search for them; it can also involve processing and cleaning datasets to make them readily usable. In the so-called age of "Big Data", the traditional roles that relegated programming to programmers, analysis to statisticians, and finding and archiving to librarians is breaking down, as the need for interdisciplinary professionals who can understand and manage all of these related processes grows [Stanton 2012]. With a few programming skills and some subject knowledge, be it bibliographic, geographic, scientific, or something else, data librarians can play a pivotal role in connecting users to useful data (for a discussion of evolving data services in academic libraries see [Wang 2013], [Mooney and Silver 2010], and [Read 2007]).

This case study illustrates a process used to create a useful subset of the IRS Tax-Exempt Masterfile for New York City and a summary of the records by borough in one spreadsheet, as an example of how data librarians can add value to unrefined datasets. This process involved three stages: using tools created by the Missouri Census Data Center to crosswalk ZIP Codes with other types of geography, using Python to select records within the geographic area of interest and to append geographic identifiers that would be helpful to users, and using OpenRefine to clean up data in the city address field of the dataset. This study demonstrates the utility of specific tools like Python and OpenRefine that data librarians and researchers can use to process data, and provides a process and script that can be used to produce this IRS dataset for other states. Common issues faced when working with address data are also illustrated; address-based datasets are widely available, but US ZIP Codes pose unique challenges for researchers [Grubesic 2008].

Baruch College, located in midtown Manhattan, is one of the senior colleges in the City University of New York (CUNY) system. Students and faculty from disciplines as diverse as marketing, real estate, entrepreneurship, public affairs, and sociology are often interested in finding neighborhood-level data within the city. Many of these questions involve generating lists of different institutions for specific places; in particular there is a keen interest in studying non-profits. The Tax Statistics Division of the IRS publishes a dataset called the Tax-Exempt Organization Masterfile [IRS 2014], which was created by scraping data from the 990 forms that all organizations must file if they are seeking tax-exempt status. The Masterfile contains the name and address of every organization, along with a number of fields that classify organizations by tax-exempt status. While this resource does not include every non-profit (religious organizations, state and federal government institutions, and small charities with annual revenue less than $50,000 are not required to file – but some do anyway), it is an extensive and solid source for this type of information. GuideStar, an information services and analysis organization for non-profits, uses this dataset as a foundation for building some of their products [GuideStar 2013].

The IRS Masterfile data is available for download in state-based csv files[i]. In many instances a user will want to capture all organizations in a particular city. However, a major challenge is that a city, as represented in a US postal address, is often not the same as an actual municipality [Stevens 2006]. Postal city names will often refer to places that are unincorporated or that appear inside incorporated places, so that a user would have to know something about how the ZIP Code[ii] coincides with municipal boundaries. The City of New York is a particularly complex example. New York, NY, only refers to addresses in Manhattan, while addresses in three other boroughs are referred to by their borough name: Bronx, Brooklyn, or Staten Island. In Queens, the city name can be Queens, or it can be one of 42 officially recognized names assigned to different neighborhood post offices. So at minimum a user wishing to capture every record in New York City would have to look for at least 47 different possibilities for city name. Unfortunately the situation is even more complicated, as residents (including the filers of the 990 forms) may use place name variants as a city address (such as Harlem in Manhattan or Riverdale in the Bronx), or may abbreviate (BKLYN, BLYN) or misspell (Broooklyn, Brooklin) the city name. Rather than using the city name, users can extract data using the ZIP Code, but in this example one would need to know what the 300 plus ZIP Codes in New York City are.

Rather than forcing many users to repeat this arduous task, a process was created to identify ZIP Codes by county, process the state-based IRS file to select records with just those ZIP Codes of interest, clean the city field to standardize it, and provide an easily accessible spreadsheet file. While this script was created to generate specific output for New York City, it was generalized so that other data librarians and researchers can use it to generate data for their own geographic areas without having to modify it. You would just need to generate a table that relates counties to ZIP Codes for your area of interest, which the script requires for input; the process for doing that is addressed in the next section.

2 ZIP Codes

A basic web search for a list of all ZIP Codes in New York City reveals countless sources, of various and sometimes dubious quality. The Missouri Census Data Center (MCDC) has been providing invaluable tools, data, and advice to US Census data users for many decades, and is a solid, definitive source [MCDC 2013a]. One of the many resources they provide is a master list or crosswalk of all 5-digit ZIP Codes that contains every ZIP Code, the type of ZIP Code it is, the postal city name, a state abbreviation, and a census ZCTA number[iii]. The challenge here is the same one that

faces us in parsing the IRS file, specifically having to select all ZIP Codes using 47 different city names. It would be preferable if we could select all ZIP Codes in a particular county[iv]. To determine what county each ZIP Code is in, we can use another MCDC tool called MABLE / Geocorr to relate different census geographies. In this case we can look up the county code for every ZCTA, and then relate the county code to the ZIP Code in the crosswalk via the ZCTA number; ZIP Codes can't be directly related to counties as they are not a census geography.

What is a ZCTA? ZIP Code Tabulation Areas were created by the US Census Bureau to approximate US Postal Service (USPS) ZIP Codes. Contrary to popular belief, ZIP Codes have no geographic area; the USPS assigns ZIPs to ranges of addresses associated with street segments. Private firms will use different methods to convert this data into actual geographic areas, and the US Census Bureau achieves this by aggregating census blocks where addresses in the blocks share common ZIP Codes [Census 2013]. Some ZIP Codes have no tangible area, either because they represent clusters of post office boxes or they are assigned to large, individual organizations that process a lot of mail. As a result there are fewer ZCTAs than ZIP Codes, as ZCTAs only represent ZIP Codes that occupy geographic space. The MCDC crosswalk can be used to assign all ZIP Codes to ZCTAs based on their location, so that PO Box and institutional ZIPs are assigned to the ZCTA where they are physically located [MCDC 2013b]. This enables users working with address-based data to aggregate ZIP Codes to ZCTAs, in order to associate their data with census data published at the ZCTA level and to map their data in geographic information systems (GIS) using ZCTA boundaries from the Census TIGER files.

For this project the ZCTA number acted as a bridge to connect the USPS ZIP Codes in the crosswalk with the table output from the MABLE / Geocorr tool that lists which county each ZCTA is in, so that we can assign a county to every ZIP Code. This makes it possible for us to select all of the New York City ZIP Codes from the crosswalk, and will allow us to append the county codes to every record in the IRS file, so that we can summarize data by counties and so that users will be able to select records by county.

Using the MABLE / Geocorr interface (at http://mcdc.missouri.edu/websas/geocorr12.html), you select a source (ZCTA) and a target (county) geography, and specify how to assign one to the other (by land area or by population). Since some ZCTAs have no residential population, the option to include records with zero population is checked. You select a state and use a filter box to input ANSI / FIPS codes [v] to limit the output to specific counties. The output table will list every ZCTA / county combination. ZCTAs don't cross state boundaries but may cross county boundaries; in those instances there will be more than one record for each ZCTA, and the apportionment column will display the percentage of the population (or land area) that is in each county. Once the file was downloaded and opened in a spreadsheet, duplicate records were deleted so that a ZCTA was assigned to only one county. In New York City there are only a handful of cases where ZCTAs cross county boundaries and in this case the effect was marginal[vi].

The ZIP crosswalk and the Geocorr table were imported into a SQLite database using the SQLite Manager, an extension available for Firefox. A basic SQL query was executed that related the two tables based on ZCTA number (see Figure 1). Since the Geocorr table only contains records for NYC, all non-matching records outside the city fall away, and the final list consists of one record for each NYC ZIP Code, with its associated ZCTA and county. The query was exported out as a text file (zip_county_nyc.csv) that contained each ZIP – County combination.

```
SELECT zip, county FROM geocorr12_nyc, zipcodes WHERE zcta5=zcta ORDER BY zip
```

ZIP	ZIPtype	CityName	Stab	ZCTA
00501	U	Holtsville	NY	11742
00544	U	Holtsville	NY	11742
06390	P	Fishers Island	NY	06390
10001	S	New York	NY	10001
10002	S	New York	NY	10002
10003	S	New York	NY	10003
10004	S	New York	NY	10004
10005	S	New York	NY	10005
10006	S	New York	NY	10006
10007	S	New York	NY	10007
10008	P	New York	NY	10007
10009	S	New York	NY	10009
10010	S	New York	NY	10010

zcta5	county	cntyname	zipname	pop10	afact
10001	36061	New York NY	NEW YORK	21102	1
10002	36061	New York NY	NEW YORK	81410	1
10003	36061	New York NY	NEW YORK	56024	1
10004	36061	New York NY	GOVERNORS ISLAND	3089	1
10005	36061	New York NY	NEW YORK	7135	1
10006	36061	New York NY	NEW YORK	3011	1
10007	36061	New York NY	NEW YORK	6988	1
10009	36061	New York NY	NEW YORK	61347	1
10010	36061	New York NY	NEW YORK	31834	1
10011	36061	New York NY	NEW YORK	50984	1
10012	36061	New York NY	NEW YORK	24090	1
10013	36061	New York NY	NEW YORK	27700	1
10014	36061	New York NY	NEW YORK	31959	1

Figure 1: MCDC crosswalk on left gets joined to Gecorr table on right using the ZCTA number

3 Processing with Python

A Python 3 script was written to serve two purposes: create a subset of the IRS file that contains just records for NYC, and create a summary table that counts the number of records for each county by type of organization. The entire program (irs_exempt_csv.py) was written as a function (irs_process) to process one state-based IRS csv file, stored in a folder that's named for the month and year (the IRS seems to update the data on a monthly basis). The function takes four arguments from the user. The first is the name of the text file that contains the ZIP Code to county relationship. The second is a month and year combination (mar2014) to identify which folder to draw the IRS data from and to append the date to the name of each output file. The third is the two-letter state code that identifies which state-based IRS file to process. The last element is a place name created by the user that generally describes the location of the records.

```
process_irs_csv('zipcounty_nyc.csv','mar2014','NY','nyc')
```

The path method from the Python os module is imported first, which allows for the construction of cross-platform paths for reading and writing files using the parameters supplied by the user and static strings. The first part of the program defines some general functions that will be called subsequently.

```
1 #!/usr/bin/python
```

```
import os.path

def process_irs(zipfile,monthyear,state,place):

#FUNCTIONS
    def dictreader(file,delim):
        """Read a file that has two values per line as a dictionary"""
        dictionary={}
        readfile=open(file)
        for line in readfile:
            key,value=line.strip().split(delim)
            dictionary[key]=value
        readfile.close()
        return dictionary

    def output_wlist(file,header,somelist):
        """Write a list out to a file, with a header row"""
        writefile=open(file,"w")
        writefile.writelines("\t".join(header)+"\n")
        for records in somelist:
            writefile.writelines("\t".join(records)+"\n")
        writefile.close()

    def addto_dict(somedict,somelist):
        """Add elements to a list value in a dictionary based on the number
        of items in another list"""
        for values in somedict:
            somedict[values]=somedict[values]*len(somelist)
```

The second part of the program creates the subset file that has the individual organization records for our area of interest. The ZIP to county relationship text file is read in as a Python dictionary called zipcounty using the dictreader function. In other programming languages the dictionary data structure is known as a hash or an associative array, but the concept is the same. A dictionary is an unsorted series of key / value pairs that are indexed by key, so that the values can be quickly retrieved by invoking their key. In this example the ZIP Code serves as the key and the county ANSI / FIPS code is the value, and the dictionary looks like this:

{'10001':'36061','10301':'36085', '10034':'36061'…}

The IRS file is read in next. The header row is captured first and saved in a list called header, and then as the program loops over each line in the file it checks the ZIP Code to see if it is included in the zipcounty dictionary; if it's in the dictionary then it's an NYC ZIP, and the record is appended to a new list called keeplist that will contain just the NYC records. Thus, each record is parsed using commas and stored as an individual list within keeplist, which is a master, nested list. An individual record list looks like this:

['133833921', 'W B YEATS SOCIETY OF N Y', '% NATIONAL ARTS CLUB', '15 GRAMERCY PARK S', 'NEW YORK', 'NY', '10003-1705', '0000', '03', '3', '2000', '199810', '1', '16', '092000000', '1', '13', '201306', '0', '0', '02', '0', '06', '0', '0', '0', 'A20Z', 'NATIONAL ARTS CLUB']

This format makes it easy for the program to loop through the list to select the list element that contains the ZIP Code, by using the index number for that element (in this case the ZIP Code is index number 6; in Python the first index is always 0). In the IRS file the ZIP Code is stored as a ZIP+4 number, so in the script we had to specifically check the first five digits for the values in index 6 to compare them against the dictionary to decide whether to keep the record or not. Subsequently we modify keeplist to create a five digit ZIP Code for each record as a new list element, and we get the associated county code for that ZIP from the zipcounty dictionary and append it as a new element. We also have to make sure we add headers for our two new elements in the header list. The list format also makes it simple to write the output; keeplist is passed through a simple function that writes the column header and each record out on its own line, with each element separated by a tab. The output file's name is appended with the date and place name initially supplied by the user (i.e. irs_xorgs_mar2014_nyc.txt).

```
zippath=os.path.join('z_geo',zipfile)
zipcounty=dictreader(zippath,',')

taxpath=os.path.join('%s','eo_%s.csv') %(monthyear,state.lower())
readfile=open(taxpath)
keeplist=[]
counter=0
header=readfile.readline().strip().split(",")
for line in readfile:
    record=line.strip().split(",")
    if any(v==record[6][0:5] for v in zipcounty.keys()):
        keeplist.append(record)
        counter=counter+1
readfile.close()

for record in keeplist:
    record[2]=record[2].strip('% ')
    zip5=record[6][0:5]
    record.append(zip5)
    record.append(zipcounty.get(zip5))

header.append('ZIP5')
header.append('County')

outpath=os.path.join('%s','irs_xorgs_%s_%s.txt') %(monthyear,monthyear,place)
output_wlist(outpath,header,keeplist)

print(counter,'records have been written to',outpath)
```

The third and final part of the program creates a summary of the records. The IRS uses Exempt-Organization codes to classify each organization into categories based on IRS regulations. Most of the organizations in the Masterfile are classified as 501(3)c (abbreviated in the data as 03), which includes most public charities and private foundations. This part of the program counts records by EO Code for each county.

First, the county codes in the zipcounty dictionary are converted to a data structure called a set, which gives us a unique list of every county code in our area of interest. That set is then converted into a sorted list; this conversion is necessary because sets cannot be sorted. Next, a dictionary is created for each of the EO Codes, where the code is the key and the value is a list with a zero as the only element. A second dictionary is created to hold the sum total of records by county. Both dictionaries are modified by passing each one into the addto_dict function; this function counts the number of counties in the county list and adds the appropriate number of values to each code in the dictionary. So in this instance, since there are five counties in NYC, each EO Code in the dictionary eocodes will have a list value with five elements. Each element will hold a count of the number of records for each county.

Once the dictionaries are ready, the program can loop over the keeplist of organization records. It looks at each EO subcode and county ANSI / FIPS code in keeplist and checks for the same EO subcode in the dictionary; once it finds the matching code it adds a 1 to the appropriate element in the list, based on the index number of the matching fips in the county list (the county codes are sorted in numeric order, so the value that's updated in the dictionary is in the same position as the value in the county list). If there isn't a matching code in the dictionary the value is dumped into an unknown category. Once the value is accounted for the sum total dictionary is updated as well.

```
fips=set(zipcounty.values())
counties=sorted(list(fips))

eocodes=dict.fromkeys(['01','02','03','04','05','06','07','08','09','10','11','12',
                       '13','14','15','16','17','18','19','20','21','22','23','24',
                       '25','26','27','29','40','50','60','70','71','81','92',
                       'unknown'],[0])
summary=dict.fromkeys(['Total'],[0])

addto_dict(eocodes,counties)
addto_dict(summary,counties)

for record in keeplist:
    subcode=record[8]
    fips=record[29]
    if subcode in eocodes:
        eocodes[subcode][counties.index(fips)]=eocodes[subcode][counties.index(fips)]+1
    else:
        eocodes['unknown'][counties.index(fips)]=eocodes['unknown'][counties.index(fips)]+1        summary['Total']
```

The remainder of the program formats the data for output. A list from the Census Bureau of all the county codes and their names [Census 2010] is read in as a dictionary using dictreader, and the names are used as column headers. Additional headers are inserted for the EO Code number and for a description of the categories; the category names are also read into a dictionary using dictreader and are inserted into the eocode dictionary in front of the county totals for each code. Since we are writing output from a dictionary and not a list we can't use the output_wlist function defined earlier. In writing the output from the dictionary, every key / value pair is written to one line, but since the value is a list the output would appear in list notation, with elements separated by commas and enclosed in brackets. As a hack we can strip out the brackets, and by default the remaining commas will serve as delimiters.

```
countyname=dictreader('z_geo/county_name.txt','\t')

sumheader=[]
for fips in counties:
    sumheader.append(countyname.get(fips))

sorted(sumheader)
sumheader.insert(0,'EO SubCode')
sumheader.insert(1,'Description')

descriptions=dictreader('z_geo/irs_501.txt','\t')

for key in descriptions:
    eocodes[key].insert(0,descriptions.get(key))
summary['Total'].insert(0,'')

outpath2=os.path.join('%s','irs_xcounts_%s_%s.txt') %(monthyear,monthyear,place)

writefile=open(outpath2,'w')
writefile.writelines(','.join(sumheader)+'\n')
for key, values in sorted(eocodes.items()):
    writefile.writelines(', '.join([key,(str(values)).strip('[]')])+'\n')
for key, values in (summary.items()):
    writefile.writelines(', '.join([key,(str(values)).strip('[]')])+'\n')
writefile.close()

print('Summary has been written to',outpath2)
```

The output file name is appended with the date and place name supplied by the user (i.e. irs_xcounts_mar2014_nyc.txt), and the content looks like this:

EO SubCode, Description, Bronx County, Kings County, New York County, Queens County, Richmond County
01, '501(c)(1) – Government Instrumentality', 1, 0, 1, 1, 0
02, '501(c)(2) – Title Holding Corporations for Single Parent Organizations', 5, 16, 117, 30, 4
03, '501(c)(3) – Charitable and Religious Organizations', 2328, 9230, 16725, 4533, 988…

4 Cleaning with OpenRefine

The last step of the process was to clean the city field in the organization records from our Python output using OpenRefine (http://openrefine.org/), which would make it easier for end users to select and manipulate records using this field. Originally a Google product called GoogleRefine, the project was reorganized and released as open source in 2012. It is a desktop application that works within a web browser. You connect to a data file (which can be stored in any number of formats) and operate in a spreadsheet-like view. For this project the text-facet tool was used to group records into categories based on city name, which gave us the ability to see every single variant (and error) associated with this field so that they could be corrected.

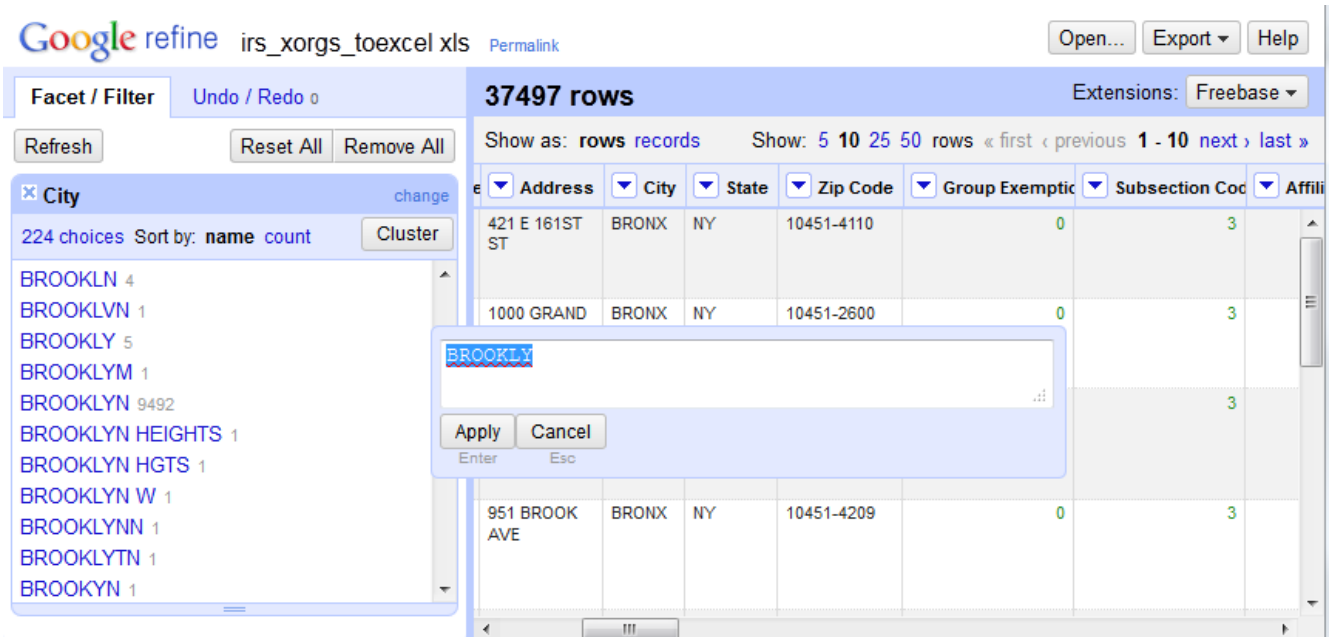

Figure 2: OpenRefine interface

OpenRefine has a cluster tool that uses heuristic algorithms to select several clusters of text that it considers similar enough to possibly be identical, and the end user has the ability to correct them all en-masse. For items the cluster tool does not catch, the user can select individual facets and correct all the records that use that facet. For example, the different abbreviations and misspellings of Brooklyn could be quickly identified, and each cluster of records could be corrected en-masse to match the correct term (see Figure 2). Using this tool, all misspellings were corrected, abbreviations standardized, and data that fell into the wrong field (i.e. a portion of the street address) adjusted. Unofficial variants of city names were left as is.

5 Conclusion

The cleaned text file of organizational records was imported into Excel using the Text Import wizard, and care was taken to assign the ZIP and county code fields as text so that leading zeros would not be dropped. The organizational summary file was imported into a second sheet in the workbook and cosmetic adjustments were made to each sheet. Lastly, a third sheet was inserted in front of the data sheets that contained a brief description of the file, some metadata, and links to the original source. The file was stored in the older .xls format to insure greater cross and backward compatibility. The data file was uploaded to the library's LibGuide platform and embedded in a guide for finding data for New York City at http://guides.newman.baruch.cuny.edu/nyc_data/hhs, a likely location where users doing city research would find it. A dedicated LibGuide box was created to hold the file, so that other librarians could link to it if their guide carried related content.

An annotated copy of the Python 3 script is included with this article, along with the necessary input files (that are read in as lists), a copy of NY State's data for Mar 2014, the ZIP to county relationship file for NYC, and the SQLite database that contains the list of all ZIP Codes and ZCTAs in the US. To generate data for other parts of the country, download the IRS csv file for your state and follow the steps in the ZIP Code section of this paper to generate a ZIP Code to County relation file for your area of interest. Create the same folder structure so that you don't have to modify path names in the script: place the script in a folder, and in subfolders store the relation file and other input files in a folder called z_geo and the IRS data in folders named for the date, i.e. mar2014. You can import the function process_irs or run the script irs_exempt_csv.py to load it in memory, then call the function with the necessary arguments: process_irs('zipfile','monthyear','state','place').

References

[Census 2013] U.S. Census Bureau [Internet]. (updated 2013). ZIP Code Tabulation Areas (ZCTAs) in Geography – Reference; [cited 2014 May 20]. Available from http://www.census.gov/geo/reference/zctas.html.

[Census 2010] U.S. Census Bureau [Internet] (updated 2010). 2010 FIPS Codes for Counties and County Equivalent Entries; [cited 2014 May 20]. Available from: http://www.census.gov/geo/reference/codes/cou.html.

[Grubesic 2008] Grubesic, TH. Zip codes and spatial analysis: problems and prospects. Socio-Economic Planning Sciences 42(2008): 129-149.

[GuideStar 2013] GuideStar [Internet] (updated 2013). What is the IRS Business Master File, and why is it important? In FAQs: Due Diligence and GuideStar Charity Check Overview; [cited 2013 Oct 23]. Available from: http://www.guidestar.org/rxg/help/faqs/guidestar-charity-check/overview.aspx.

[Hayter 2005] Hayter CT. 2005. Adding a little ZIP to the mail: The development and growth of the ZIP Code. DttP: Documents to the People 33(3): 43-51.

[IRS 2014] Internal Revenue Service [Internet]. (updated 2014). SOI Tax Stats – Exempt Organizations Business Master File Extract; [cited 2014 Apr 23]. Available from: http://www.irs.gov/Charities-&-Non-Profits/Exempt-Organizations-Business-Master-File-Extract-EO-BMF..

[MCDC 2013a] Missouri Census Data Center [Internet]. (updated 2013). [cited 2013 Oct 17]. Available from: http://mcdc.missouri.edu/.

[MCDC 2013b] Missouri Census Data Center [Internet]. (updated 2013). All About ZIP Codes: 2010 Supplement; [cited 2013 Oct 17]. Available from: http://mcdc.missouri.edu/pub/allabout/zipcodes_2010supplement.shtml.

[Mooney and Silver 2010] Mooney H, Silver B. 2010. Spread the news: promoting data services. College & Research Libraries News 71(9): 480-483.

[Read 2007] Read EJ. 2007. Data services in academic libraries: Assessing needs and promoting services. Reference & User Services Quarterly 46(3): 61-75.

[Stanton 2012] Stanton JM. 2012. Data science: what's in it for the new librarian? Information Space, July 16. [Internet]. [cited 2014 May 20]. Available from: http://infospace.ischool.syr.edu/2012/07/16/data-science-whats-in-it-for-the-new-librarian/.

[Stevens 2006] Stevens N. 2006, June 23. Changing Postal ZIP Code Boundaries. Washington DC: Congressional Research Service. Order Code RL33488. [Internet]. [cited 2013 Oct 17]. Available from: https://www.fas.org/sgp/crs/misc/RL33488.pdf.

[Wang 2013] Wang M. 2013. Supporting the research process through expanded library data services. Program: electronic library and information systems 47(3): 282-303.

About the Author

Frank Donnelly (francis.donnelly@baruch.cuny.edu) is the Geospatial Data Librarian and an Assistant Professor at the William and Anita Newman Library at Baruch College, City University of New York (CUNY). He is a subject specialist in geography and geographic information systems (GIS) and has extensive experience working with US Census data. He holds an MLIS from the University of Washington and an MA in Geography from the University of Toronto.

Notes

[i] The IRS began publishing the data in this format in January 2014 to make it more accessible. Prior to that time the data was available in state-based fixed-width text files that required a codebook to parse, or as a series of Excel spreadsheets, with data for large states divided into several files.

[ii] A ZIP Code is a US postal code. An abbreviation for Zone Improvement Plan Code, ZIPs were introduced in 1963 to speed mail processing and delivery. For a brief history see [Hayter 2005].

[iii] The original source of the crosswalk is the UDS Mapper at http://udsmapper.org/zcta-crosswalk.cfm.

[iv] The five boroughs of NYC are also individual counties within the State of New York; in some instances the borough names differ from the county ones. In this paper I use the term county, since that is the common term others would use in applying this project to their own geographic area.

[v] Every piece of census geography is assigned an ANSI / FIPS number to uniquely identify it. Every state is assigned a two digit code and every county a three digit code; for example 36 is the code for New York State, and within New York State 005 is the code for the Bronx. For a list of state and county codes see [Census 2010].

[vi] The effect may be greater in other places, for instance if a ZIP Code is split in half by a county. The MCDC estimates that 10% of all ZIP Codes cross county boundaries. If a ZIP Code split introduces too much error, records for those organizations would have to be spot-checked or geocoded to unequivocally identify which county they are located in. In NYC there are three ZIP Codes (11001, 11003, and 11040) that are partially located in Queens but are primarily in Nassau County on Long Island. These ZIPs were manually deleted from the zipcounty_nyc.csv file so that the IRS records in these areas would not be included in the final output.

Sample File

A copy of the sample data files and scripts can be found here: http://journal.code4lib.org/media/Issue25/donnelly/article_asset_files.zip