Methods to Process the Household Income Census Data

**Overview**: ACS census provide socioeconomic data from a variety of topics and at different spatial scales. To extract the Census information for Census Block Group (CBG) on community/state level, I used method I; while this Census is not able to process CBG data on national scale, which I used method II in producing the national layer.

**Method I: Use** [**Summary File Retrieval Tool**](http://www2.census.gov/acs2012_5yr/summaryfile/UserTools/SummaryFileDataRetrievalTool.zip) **to process each community data**

**Steps:**

1. **Download** [**Summary File Retrieval Tool**](http://www2.census.gov/acs2012_5yr/summaryfile/UserTools/SummaryFileDataRetrievalTool.zip) **and unzip the excel tool.**
2. **Open SummaryFileDataRetrievalTool\_V1.0.0.13.2.xlsm, enable the macros, press  button and follow the instruction to retrieve the data.**

* First, you will be prompted to select a state and an ACS dataset (5-year 2008-2012).
* Second, you will be prompted to select a table. Here we used Table 19001 for Household Income.
* Excel will then retrieve the selection table for all available units of geography in the state. Please note: retrieval can take several minutes, particularly for multi-year data files and large geographic areas.

1. **Obtained an ACS Summary File from Step 2. Next, we joined the ACS Summary File to the TIGER/Line Shapefiles**

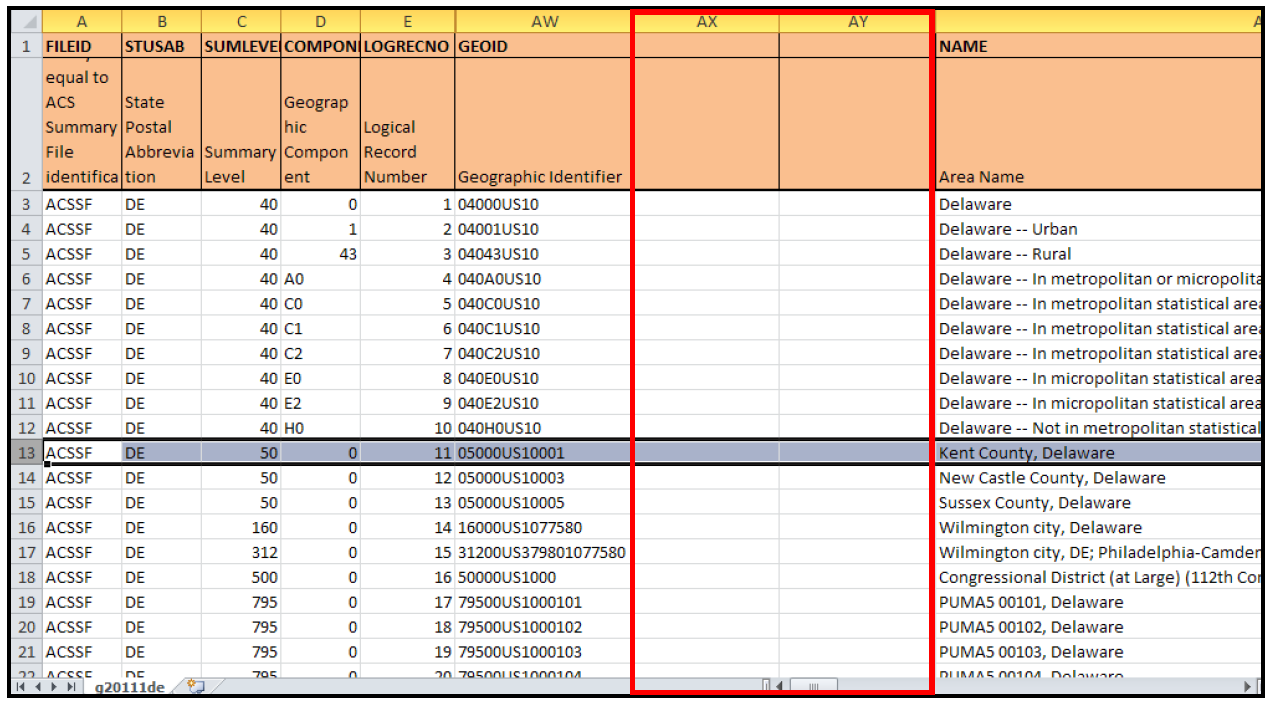
The variable GEOID joins the ACS Summary File to the TIGER/Line Shapefiles. For the ACS Summary File, GEOID is located in column AW of the geography file. It is not found in the estimates or margins of error files. GEOID’s corresponding variable in the 2012 TIGER/Line Shapefiles is also GEOID.

The ACS Summary File GEOID contains the necessary information to connect to the TIGER/Line Shapefiles, but it needs to be modified in order to exactly match up. Notice that the ACS GEOID, 05000US10001, contains the TIGER/Line GEOID string, 10001.

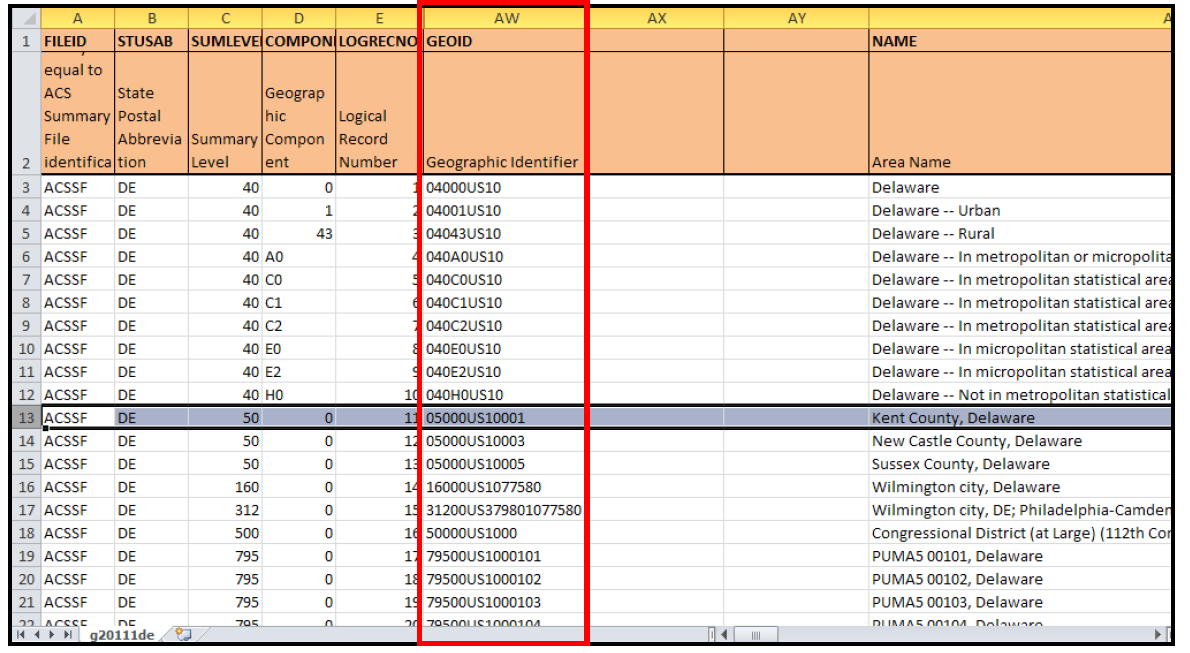
In order to create an exact match of both GEOIDs, it is necessary to remove all of the characters before and including the letter “S” in the ACS Summary File. By removing these characters, the new GEOID in the ACS Summary File exactly matches the field GEOID in the TIGER/Line Shapefiles.

The following is an example of how to modify the ACS Summary File’s GEOID in Excel so it can be joined with TIGER/Line Shapefiles:

1) Open the ACS Summary File comma delimited geography file in Excel. This example uses Delaware’s geography file (20121de.csv).

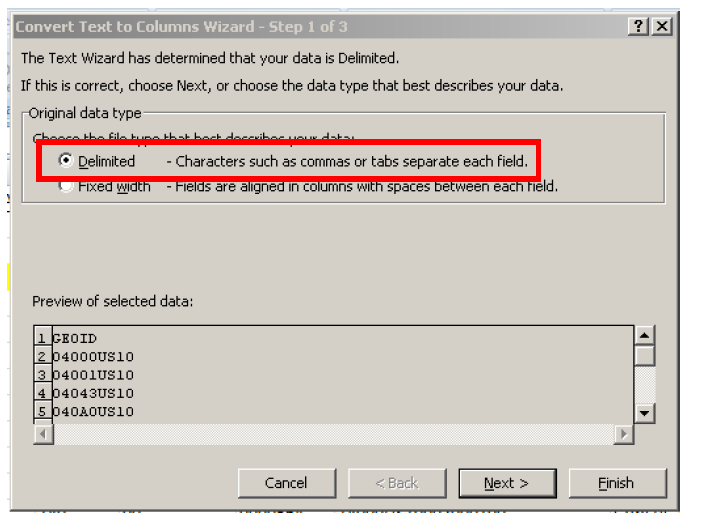
2) Insert 2 blank columns to the right of the column “GEOID.” Your modified GEOID will eventually go into the second column. 

3) Next, select the column “GEOID.”

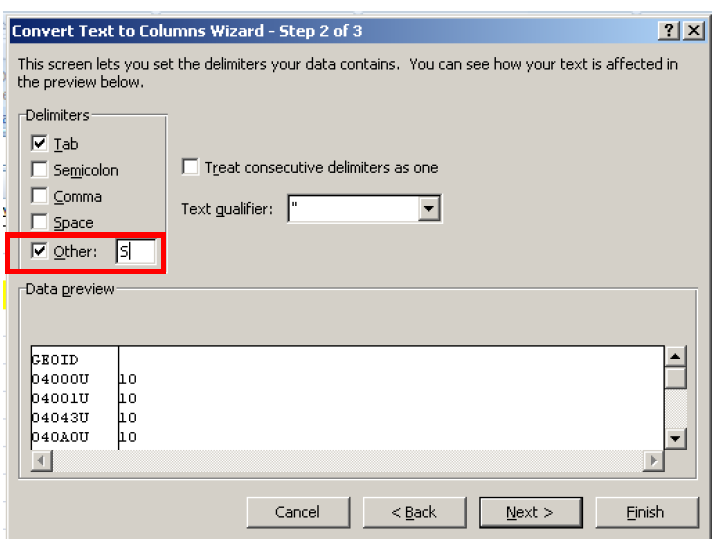


4) Select the “Data” tab from the top menu, then select “Text to Columns.” The “Convert Text to Columns Wizard” box should pop up.

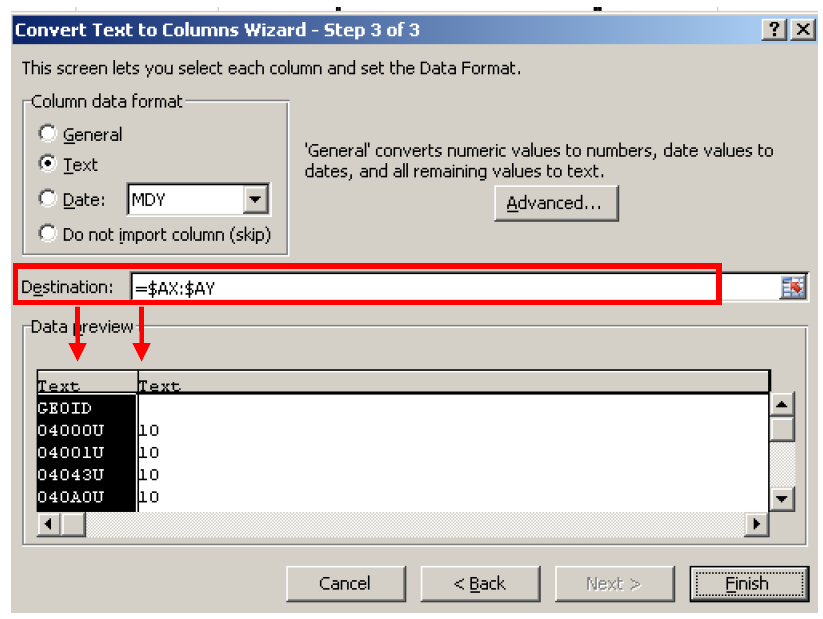
5) In the “Convert Text to Columns Wizard,” select “Delimited” under “Choose the file type that best describes your data:” then click “Next.”



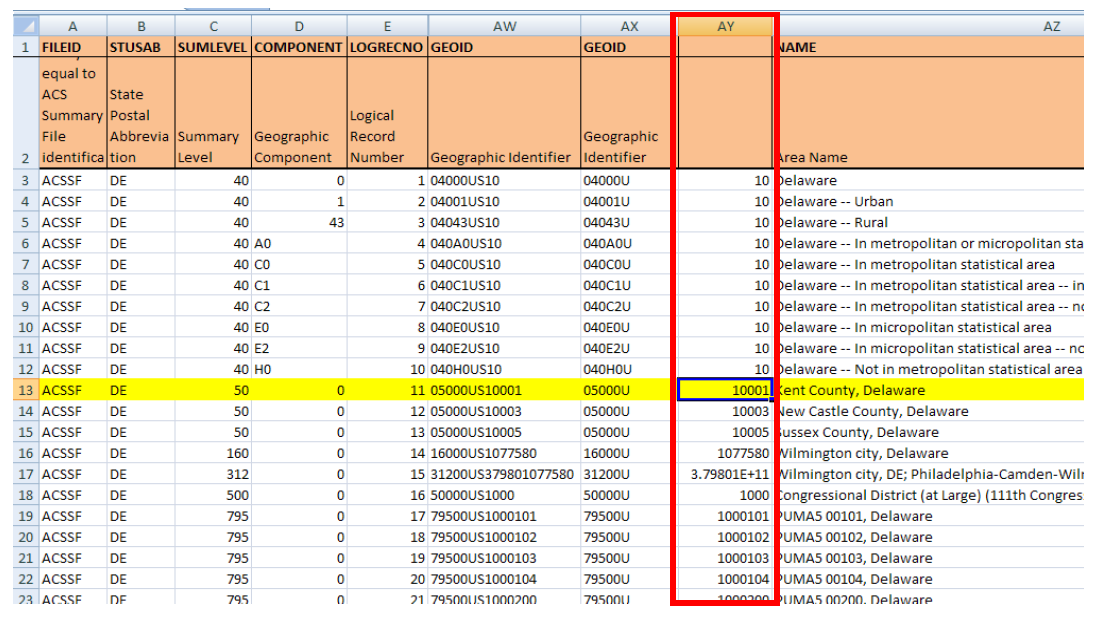
6) Check “Other” as the delimiter and type the letter “S” into the box. Click “Next.”



7) In the “Data preview” window, click on the top of both columns in “Data preview” and select “Text” under “Column data format.” In “Destination,” select the two blank columns that you created in Step 1. Click “Finish.”



8) Column AY should now contain the modified ACS GEOID that corresponds to GEOID in the TIGER/Line Shapefiles. The second screenshot shows the TIGER/Line Shapefile for Kent County, Delaware.



9) The ACS Summary File and the TIGER/Line Shapefile should now be ready to be joined using GIS software.

1. After extract the ACS data information. The following Census Income Categories information should be provided as shown in Table 1. We further classified and recorded into four categories:

**Table 1. Household Income Categories provided by Census ACS**

|  |  |  |
| --- | --- | --- |
| Census Income Categories | Further Classification | Field Title in GIS Attribute Table |
| Less than $10,000 | Percentage Population Below twice the poverty line | P\_less\_30k |
| $10,000 to $14,999 |
| $15,000 to $19,999 |
| $20,000 to $24,999 |
| $25,000 to $29,999 |
| $30,000 to $34,999 | Percentage Population Above twice the poverty line and below $75,000 | P\_30k\_75k |
| $35,000 to $39,999 |
| $40,000 to $44,999 |
| $45,000 to $49,999 |
| $50,000 to $59,999 |
| $60,000 to $74,999 |
| $75,000 to $99,999 | Percentage Population Above $75,000 and below $200,000 | P\_75k\_200k |
| $100,000 to $124,999 |
| $125,000 to $149,999 |
| $150,000 to $199,000 |
| $200,000 or more | Percentage Population Above $200,000 | P\_more\_200k |

**Method II: Use Python Script to organize the national data**

1. Download data from [2008-2012\_ACSSF\_All\_In\_2\_Giant\_Files(Experienced-Users-Only)](http://www2.census.gov/acs2012_5yr/summaryfile/). Unpack All\_Geographies\_Not\_Tracts\_Block\_Groups.tar.gz and 2012\_ACS\_Geography\_Files.zip. All\_Geographies\_Not\_Tracts\_Block\_Groups.tar.gz was called estimation file, and 2012\_ACS\_Geography\_Files.zip was called geographic information.
2. Yan developed python scripts to process the
3. Read estimation file by running read\_estfile.py, results were saved as est\_output\_US.txt.
4. Read geographic file by running read\_geofile.py, results were saved as geo\_output\_US.txt.
5. Combine est\_output\_US.txt and geo\_output\_US.txt by running join\_tables.py with Mapreduce.py in the same folder. This step generates a file final\_US.txt.
6. Split the big table into small ones with 60000 rows for each one. Run split\_final.py. This step generates output1.txt, output2.txt, output3.txt, output4.txt.
7. Import the four files, output1.txt, output2.txt, output3.txt, output4.txt, into Excel 2003 file. ArcGIS 10.2 doesn’t read Excel 2013.
8. Merge output1, output2, output3, output4 tables into one table, which is the national data dbf table. Do join with national CBG layer to produce a national data layer.

read\_estfile.py

# Path = C:/Python27/ArcGIS10.2

# Path = E:/Python27

# E:/ArcGIS/ACS/national/A\_python\_scripts

""" Value for household income from column 7 to 23 """

import os, sys, csv, glob

from collections import defaultdict

## Set work place

os.chdir("E:/ArcGIS/ACS/national/A\_python\_scripts")

## Helper functions

def get\_sum(list):

""" loop through a list (cannot be a string!!) and skip missing data '1','2','3','','4',..., get the sum of all values """

sum = 0

if list:

for i in list:

if i:

sum += float(i)

return sum # return a float number

def calculate(f):

result =[]

for line in f:

list = line.split(',')

CBG\_id =list[5]

state = list[2].upper()

total\_pop = list[6]

# # for household income

# # for total\_pop = 0 or missing, use -9999

if total\_pop: # get rid of missing values, but still have 0 value for total pop

total\_pop = float(total\_pop)

if total\_pop:

less\_30k = list[7:12]

less\_30k\_sum = get\_sum(less\_30k)

less\_30k = 100 \* less\_30k\_sum/total\_pop

range\_30k\_75k = list[12:18]

range\_30k\_75k\_sum = get\_sum(range\_30k\_75k)

range\_30k\_75k = 100 \* range\_30k\_75k\_sum/total\_pop

range\_75k\_200k = list[18:22]

range\_75k\_200k\_sum = get\_sum(range\_75k\_200k)

range\_75k\_200k = 100 \* range\_75k\_200k\_sum/total\_pop

more\_200k = list[22]

#more\_200k\_sum = get\_sum(more\_200k)

more\_200k = 100 \* float(more\_200k)/total\_pop

else:

less\_30k = -9999

range\_30k\_75k = -9999

range\_75k\_200k =-9999

more\_200k = -9999

else:

less\_30k = -9999

range\_30k\_75k = -9999

range\_75k\_200k =-9999

more\_200k = -9999

line\_result = [CBG\_id, state, str(less\_30k), str(range\_30k\_75k), str(range\_75k\_200k), str(more\_200k), str(sum([less\_30k, range\_30k\_75k, range\_75k\_200k, more\_200k]))]

#print line\_result

result.append(line\_result)

return result

## Read input files from a dictionary

datafile\_path = "E:/ArcGIS/ACS/national/Data/e2012\*0058000.txt"

datafile\_path2 = "E:/ArcGIS/ACS/national/Data/e20125nc0058000.txt"

## Run tools to process the data

all\_result =[]

for f\_name in glob.glob(datafile\_path2):

f = open(f\_name, "r")

f\_result = calculate(f)

#print f\_result

all\_result.extend(f\_result)

print len(all\_result)

## Write all results into a txt file

outfile = open("est\_output\_temp.txt", "w")

for item in all\_result:

outfile.write(" ".join(item))

outfile.write("\n")

outfile.close()

read\_geofile.py

# Path = C:/Python27/ArcGIS10.2

# Path = E:/Python27

# E:/ArcGIS/ACS\national/A\_python\_scripts

""" This script read geofile and get a list of GEO\_id and LOGRECNO """

import os, sys, csv, glob

from collections import defaultdict

## Set work place

os.chdir("E:/ArcGIS/ACS/national/A\_python\_scripts")

## Calculate function

def Calculate(f):

f\_result = []

for line in f:

""" extract """

list = line.split(',')

LogRecNo = list[4]

state = list[1]

GEO\_id = list[48][7:]

if len(GEO\_id) == 12:

line\_list = [LogRecNo, state, GEO\_id]

f\_result.append(line\_list)

return f\_result

## Read file

datafile\_path = "E:/ArcGIS/ACS/national/Geo\_Files/\*.csv"

datafile\_path2 = "E:/ArcGIS/ACS/national/Geo\_Files/g20125nc.csv"

## Loop through files

all\_result =[]

for f\_name in glob.glob(datafile\_path):

f = open(f\_name, "r")

f\_result = Calculate(f)

#print f\_result

all\_result.extend(f\_result)

print len(all\_result)

## Write all results into a txt file

outfile = open("geo\_output\_nc.txt", "w")

for item in all\_result:

outfile.write(" ".join(item))

outfile.write("\n")

outfile.close()

combine\_txt\_file.py

import sys, os

filenames = ['geo\_output\_US.txt', 'est\_output\_US.txt']

with open('E:/ArcGIS/ACS/national/A\_python\_scripts/combine\_tables\_US.txt', 'w') as outfile:

for fname in filenames:

with open(fname) as infile:

for line in infile:

outfile.write(line)

join\_tables.py

import MapReduce

import sys

"""

Word Count Example in the Simple Python MapReduce Framework

"""

mr = MapReduce.MapReduce()

# =============================

# Do not modify above this line

def mapper(record):

# key: document identifier

# value: document contents

#print record

line = record.split()

#print line, len(line)

key = tuple([line[0],line[1]])

value = line[2:]

mr.emit\_intermediate(key, value)

def reducer(key, list\_of\_values):

# key: word

# value: list of occurrence counts

join\_list\_temp = [key]

join\_list = [join\_list\_temp[0][0]]

num\_join\_lists = len(list\_of\_values)

#print join\_list, list\_of\_values, num\_join\_lists

if num\_join\_lists == 2:

for i in range(num\_join\_lists):

join\_list.extend(list\_of\_values[i])

mr.emit((join\_list))

# Do not modify below this line

# =============================

if \_\_name\_\_ == '\_\_main\_\_':

inputdata = open(sys.argv[1])

mr.execute(inputdata, mapper, reducer)