PopGen 2.0

User Guide

The PopGen Team

Acknowledgements

PopGen is a state-of-the-art open source synthetic population generator for advanced travel demand modeling. Under the direction of Professor Ram M. Pendyala, Arizona State University, the development of PopGen has been made possible through the efforts and support of many individuals and agencies/sponsors. Professor Karthik Konduri of the University of Connecticut is the chief software architect and developer of PopGen 1.1 and 2.0. Over the years, a number of individuals have assisted in the development, testing, and enhancement of PopGen including: Mr. Bhargava Sana, San Francisco County Transportation Authority; Dr. Venu Garikapati, Arizona State University; and Dr. Daehyun You, Maricopa Association of Governments. The original development of PopGen was largely made possible with support from the Ira A. Fulton Schools of Engineering at Arizona State University and the Exploratory Advanced Research Program (EARP) of the Federal Highway Administration, US Department of Transportation. The methodology underlying PopGen was originally developed by Professor Xin Ye, Tongji University with input from Professor Hillel Bar-Gera of Ben Gurion University in Israel. Mr. Kunal Mehan, Georgia Institute of Technology, prepared the documentation for PopGen 2.0, while Mr. Keith Christian, Kimley-Horn and Associates, prepared documentation and training materials for PopGen 1.1 and provided graphics support for this enterprise. Ms. Julie Dunbar of Dunbar Transportation Consulting also assisted in the preparation of documentation and training materials.

The PopGen software system has benefited greatly from the support of several sponsors. Mr. Brian Gardner of the Federal Highway Administration served as the project manager for the EARP project that provided partial funding for the development of PopGen. Additional support for the development of PopGen has been provided by the Strategic Highway Research Program (SHRP2), Baltimore Metropolitan Council, Southern California Association of Governments, Maricopa Association of Governments, and Colorado Department of Transportation. The support and cooperation of these agencies, as well as that of Georgia Institute of Technology and the University of Connecticut, is gratefully acknowledged.

Disclaimer

PopGen is made available to the public under the GNU General Public License Version 3.0 (http://www.gnu.org/licenses/gpl.html). This software is provided by the developers and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the developers, contributors, or sponsors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Table of Contents

Introduction to PopGen

What is PopGen?

The advent of activity-based microsimulation models of travel demand has provided the ability to simulate activity-travel patterns of individual travelers in time and space. The application of activity-based travel demand model systems requires the generation of synthetic populations whose attribute distributions match those of the general population in small geographies (say, a census tract, traffic analysis zone or TAZ, block group, or block). The generation of synthetic populations has therefore received much attention in the profession and several synthetic population generators or population synthesizers have been developed and applied in various urban areas that are transitioning to activity-based model systems. PopGen is a synthetic population generator that incorporates a new heuristic algorithm for iteratively generating a synthetic population. In traditional synthetic population generators, the distributions of household-level attributes match well with census (actual) distributions, but the distributions of person-level attributes are not adequately controlled to ensure the best possible match with census (actual) distributions. The algorithm implemented in PopGen is capable of generating synthetic populations while controlling and matching both household-level and person-level attribute distributions. PopGen is written entirely in Python and employs data structures and numerical libraries that provide efficient computational performance. PopGen provides the flexibility to synthesize a population at various levels of geography for small, medium, or large regions. PopGen has been tested and applied in a number of regions. Results obtained from these varied applications of PopGen are extremely promising and suggest that PopGen is capable of synthesizing populations with attribute distributions that

closely match actual distributions in the general population. The developers of PopGen are now working on the next generation population synthesizer, which will incorporate full population evolution and socio-economic dynamics to evolve a base-year population over time. PopGen is being made available to the professional community through open-source licensing arrangements with a view to foster further research and development in population synthesis.

PopGen 2.0 vs PopGen 1.1

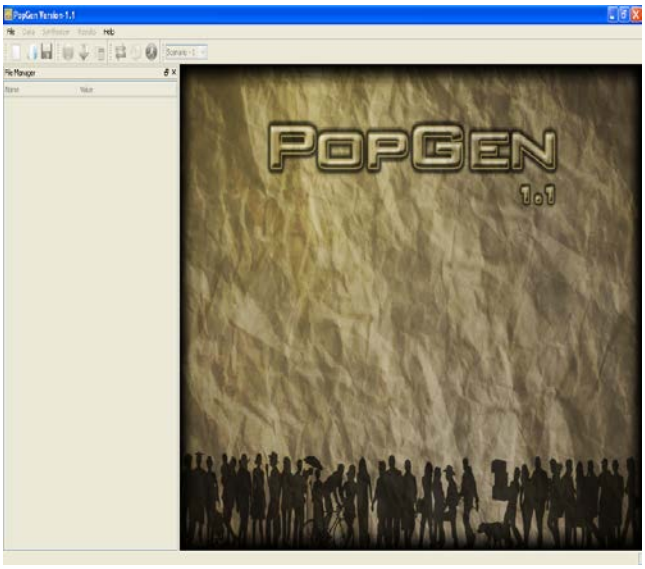


Figure 1.1. PopGen 1.1.

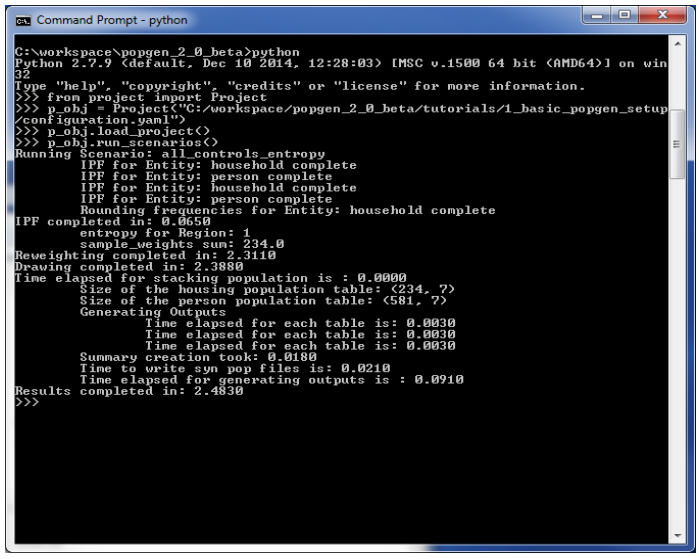


Figure 1.2. PopGen 2.0.

One major enhancement made in PopGen 2.0 over PopGen 1.1 is that in PopGen 2.0, users can control for variables of interest at multiple geographic resolutions simultaneously. For example, if control totals for some variables of interest are available at the TAZ level, and for some other variables at the County level, it is possible to control for variables from both of the geographic resolutions simultaneously using PopGen 2.0. This helps in generating a robust synthetic population and fully utilizing all of the information available to the

analyst for the region in question. The interface in PopGen 2.0 is also vastly simplified; instead of using a Graphical User Interface, PopGen 2.0 is operated through a simple set of operations using the Command Line Interface. This allows the user to more efficiently and quickly carry out the synthetic population generation. Owing to this simplified interface and an enhanced algorithm, PopGen 2.0 is less computationally intensive than PopGen 1.1.

As the PopGen development team recognizes that some users may prefer the use of a PopGen software system with a graphical user interface, the team has continued to refine and support PopGen 1.1. PopGen 1.1 offers a full-fledged graphical user interface and provides flexibility and functionality in the generation of a synthetic population. However, PopGen 1.1 does not incorporate the ability to accommodate control distributions for variables at multiple geographical resolutions. PopGen 1.1 can only accommodate controls at a single geographical resolution.

Introduction to Command Line Interface

These are some basic instructions to use a Command Line Interface. These have been tested to work with Windows 7, 8, 8.1 and 10. Open a Windows command line window by following the steps below:

1. Click ***Start.***
2. In the *Search* or *Run* line type **'cmd'** and press ***Enter***.

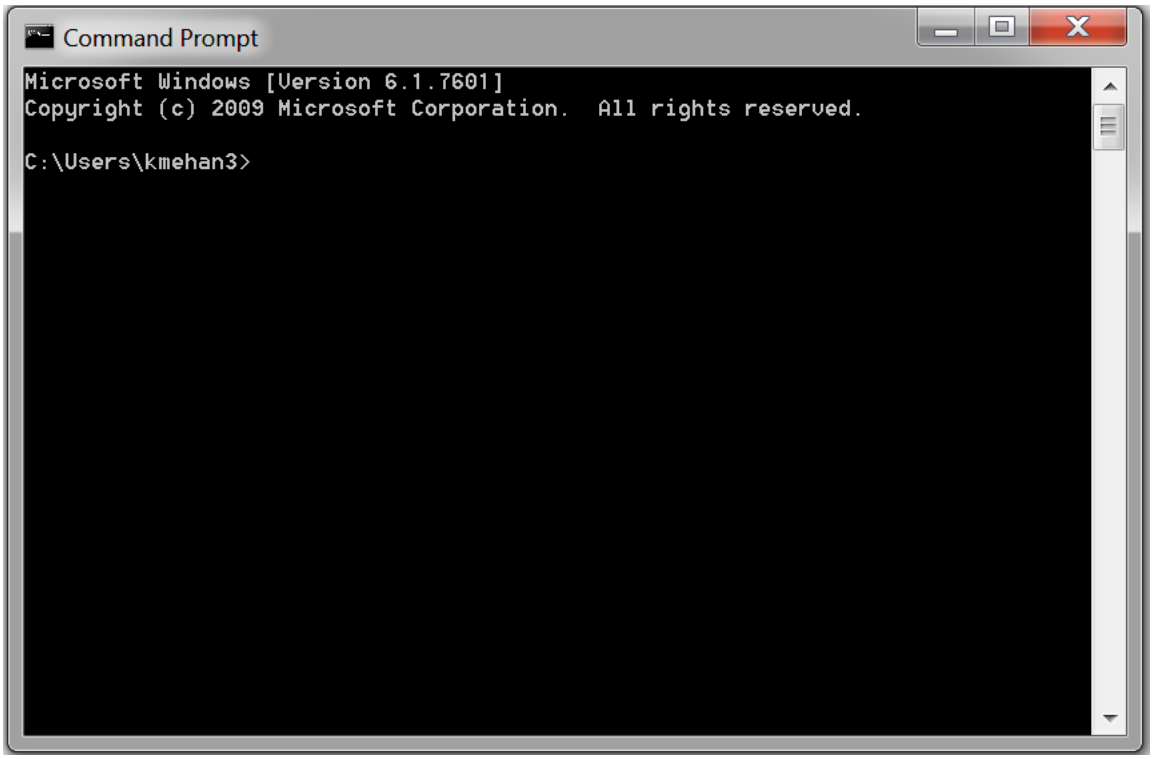


Figure 1.3. Command Line Window.

Understanding the prompt

After following the above steps, the Windows command line should be shown (similar to the example above). Typically, Windows starts you at your user directory. In the example above, the user is *kmehan3*, so our prompt is *C:\Users\Windows>.* This prompt tells us we are in the *C:* drive (the default drive letter of the hard drive) and currently in the *kmehan3* directory, which is a subdirectory of the *Users* directory.

Listing the files

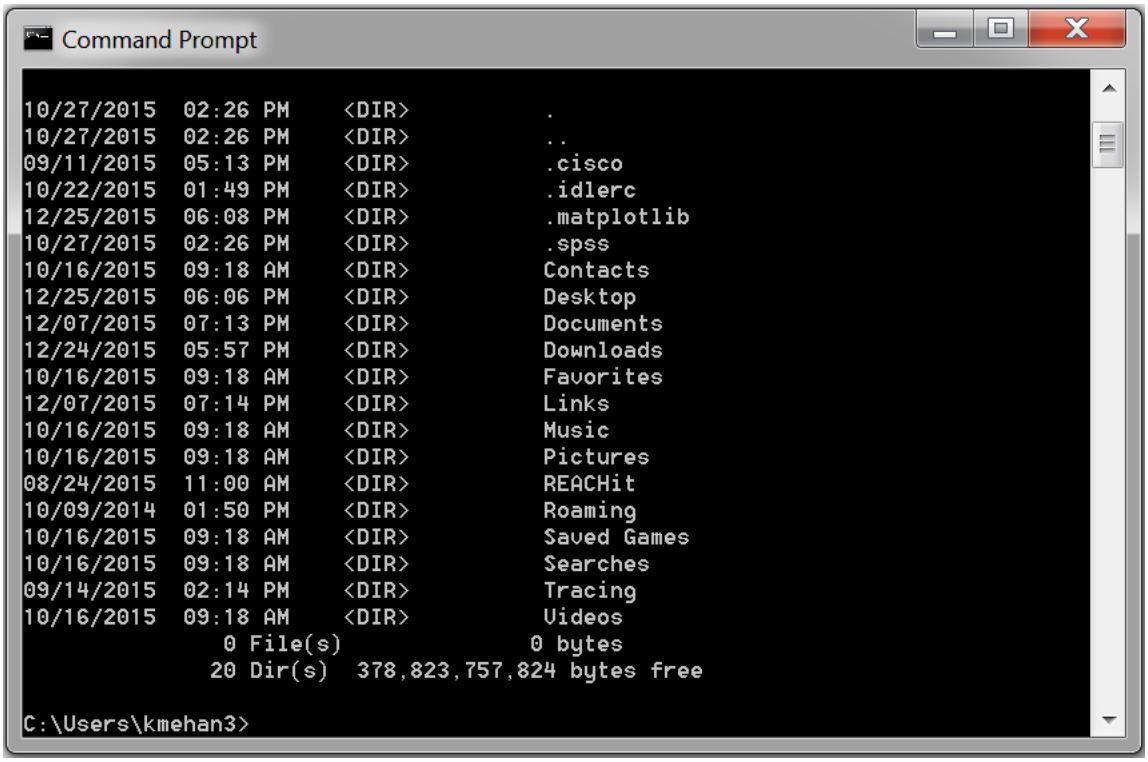


Figure 1.4. *dir* Command.

Type **'dir'** at the prompt to list files in the current directory. You should get an output similar to the example image above. Without using any specific *dir* options, this is how *dir* output appears. As can be seen, you are given lots of useful information including the creation date and time, directories and the name of the directory or file. In the example above, there are 0 files listed and 20 directories as indicated by the status at the bottom of the screen.

Every command in the command line has options, which are additional switches and commands that can be added after the command. For example, with the *dir* command you can type **'dir /p'** to list the files and directories in the current directory one page at a time. This switch is useful to see all the files and directories in a directory that has dozens or hundreds of files. The *dir* command can

also be used to search for specific files and directories by using wildcards. For example, if you only wanted to list files or directories that begin with the letter **'m'** you could type **'dir m*'** to list only the *Music* directory, for example.

Moving into a Directory

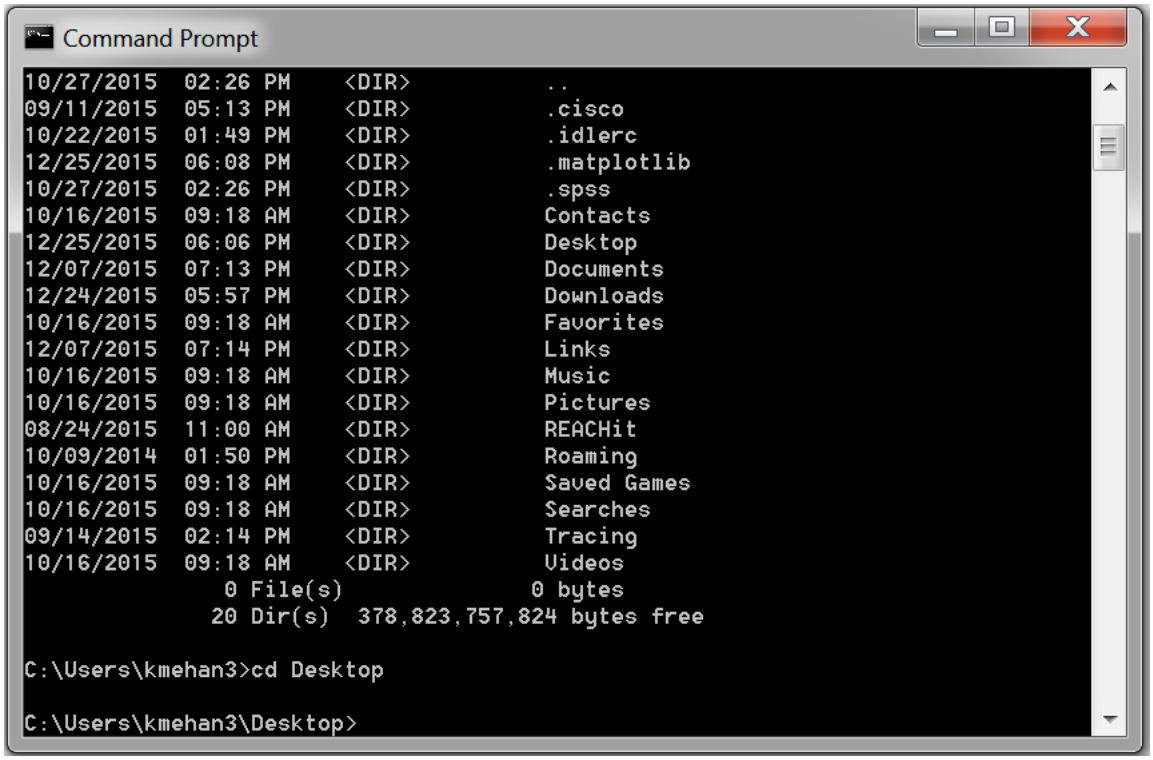


Figure 1.5. *cd* Command.

Now that we've seen a list of directories (shown above) in the current directory, move into one of those directories. To move into a directory, we use the *cd* command; so to move into the *Desktop*, type **'cd desktop'** and press ***enter***. Once you've moved into a new directory, the prompt should change; so in our example, the prompt is now *C:\Users\kmehan3\Desktop>.* Now in this desktop directory, see what files are found in this directory by typing the **'dir'** command again.

Moving back a Directory

You learned earlier the *cd* command can move into a directory. This command also allows you to go back a directory by typing **'cd..'** at the prompt. When this command is typed you'll be moved out of the *Desktop* directory and back into the *user* directory. If you wanted to move back to the root directory, typing **'cd\'** takes you to the *C:\>* prompt. If you know the name of the directory you want to move into, you can also type **'cd\'** and the directory name. For example, to move into *C:\Windows>* type **'cd\windows'** at the prompt.

List of Basic Commands

1. *cd* <directory name>

   cd is the basic command; it allows you to change directory

2. *dir* [name of directory]

   dir allows you to list all contents of the specified directory

3. *copy* <source> <destination>

   Allows you to copy a file from a <source> folder to a <destination> folder

4. *del* <file>

   Deletes a specific file

5. *move* <source> <destination>

   Allows you to move a file from a <source> folder to a <destination> folder

6. *ren* <source> <destination>

Renames the specified file

7. *edit* <filename>

   Opens the default editor to allow modification of a specified file

8. cls

   Clears the command line screen

9. exit

   Quits the command line terminal

Setup and Installation

Downloading Python 2.7

PopGen 2.0 requires Python 2.7 or higher to run correctly.

- If you have an earlier version of Python, please uninstall it, restart your computer and proceed with the Python 2.7 Installation Procedure.
- If you already have Python 2.7 or higher installed on your computer, please skip directly to **"Downloading PopGen 2.0"**.

We will be using Python 2.7.9 64-bit to demonstrate the installation procedure; it can be found at **https://www.python.org/ftp/python/2.7.9/python-2.7.9.amd64.msi**

Installing Python 2.7.9

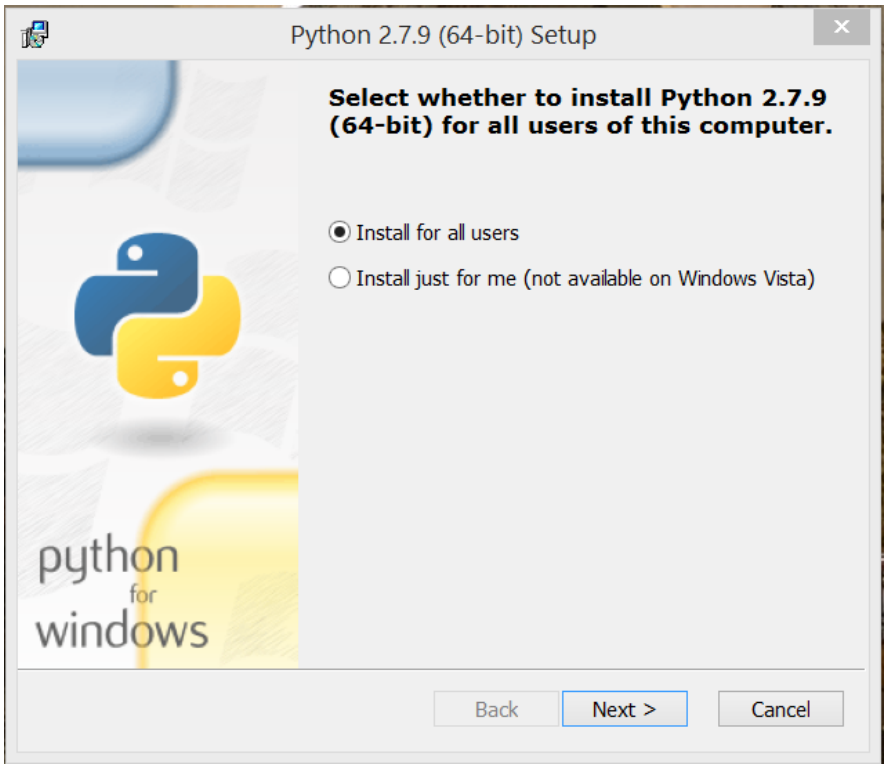


Figure 2.1. Installation Window.

To start the installation procedure, double click on the **python-2.7.9.amd64.msi** file. The Python Installation window should open (resembling the above), but may vary based on your operating system.

1. Click ***next*** to continue installation.
2. The default installation path on the next screen should *C:\Python27*, please change the path name to '**C:\Python27.9'** in the box.
3. Clicking ***next*** will take you to the *Customize Python 2.7.9* screen, click ***next*** again to continue.
4. Wait for Python to complete the installation; do not interrupt the installation in any way. After it is complete, press the Finish button to exit the installation.

Congratulations, you have successfully installed Python 2.7.9!

Testing Python Installation

Before we install any dependencies, we need to make sure that our Operating System is pointing to the path for the Python Installation.

1. Navigate to the *Start Menu.*
2. Click on ***Control Panel.***
3. Navigate to *System.*
4. Click ***Advanced system settings.***
5. Click ***Environment Variables.***

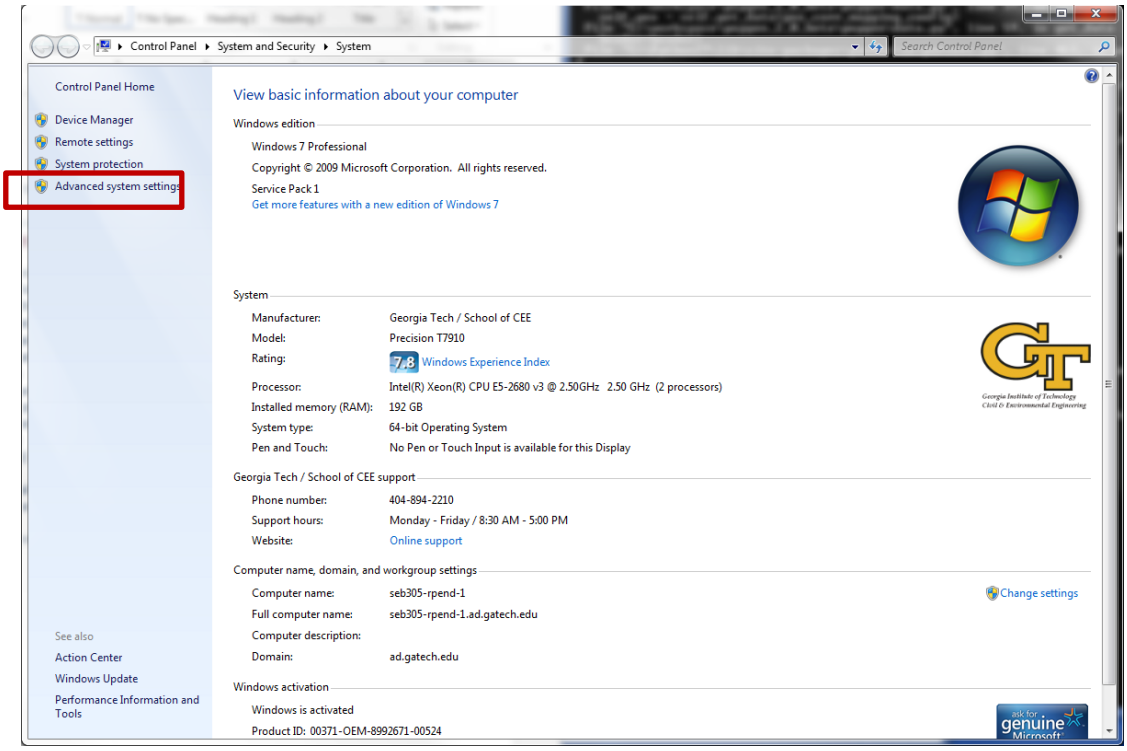


Figure 2.2. System Overview.

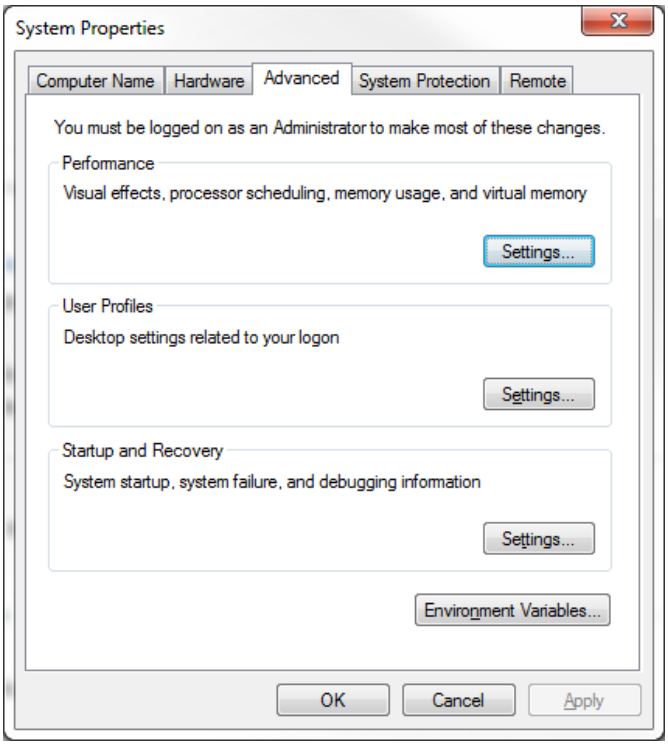


Figure 2.3. System Properties.

6. Select ***Path*** variable in the *System variables* section and click ***Edit...***
7. Delete any (and all) old paths. For example, *C:\Python27* and *C:\Python27\Scripts*
8. If the paths *C:\Python27.9* and *C:\Python27.9\Scripts* exist, then let it be, otherwise add them.

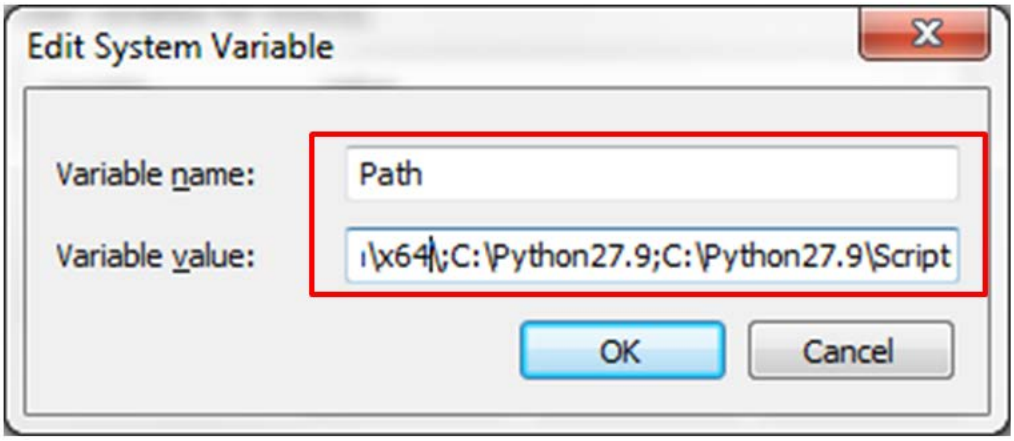


Figure 2.4. Edit System Variable.

9. Click ***OK.***

Downloading & Installing PopGen 2.0

The latest releases of PopGen 2.0 can be found under the releases page at

https://github.com/foss-transportationmodeling/popgen/releases

As of **11/5/2016**, the latest release of PopGen is v2.0.0b2 and can be directly downloaded from **https://github.com/foss-transportationmodeling/popgen/archive/v2.0.b2.zip**

1. Extract the zip file to the folder *popgen-2.0.b2*
2. Open *command prompt* and navigate to the directory *popgen-2.0.b2*
3. Once in the directory, Type **'python setup.py install'** and hit ***enter.***

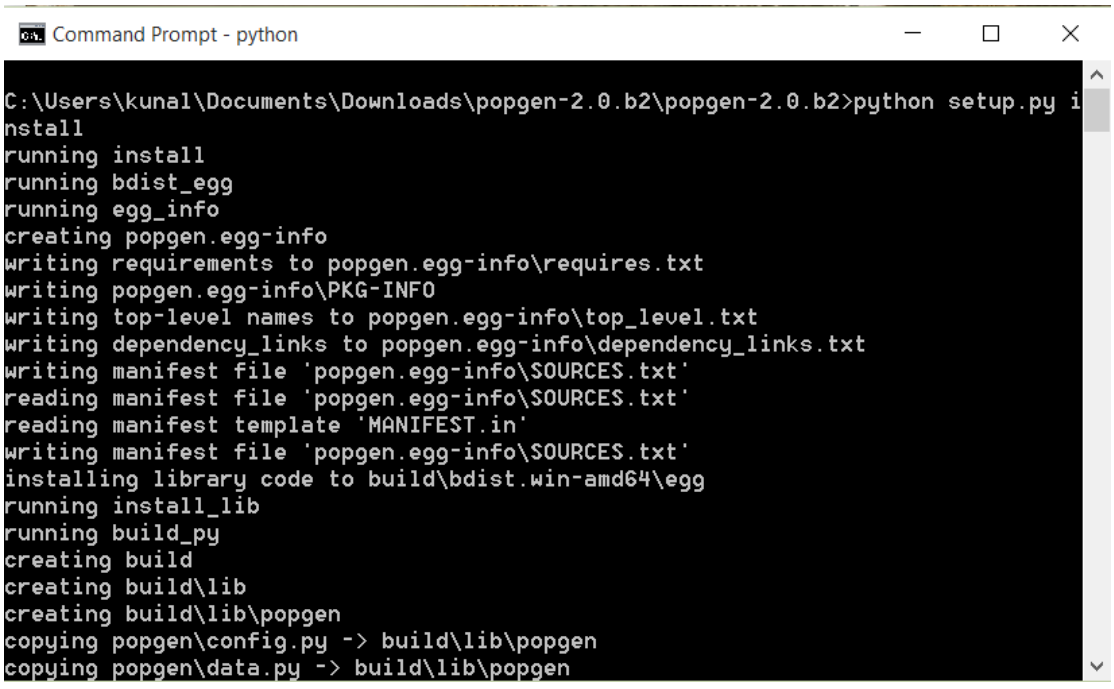


Figure 2.5. Python Command Prompt.

Data Preparation: Input Files

Geographical Correspondence

PopGen requires the mapping between different spatial resolutions in order to generate a synthetic population. For example, the American Community Survey (ACS) Public Use Microdata Sample (PUMS) data is available at the resolution of a Public Use Microdata Area (PUMA), whereas the ACS summary files are available at the level of census tracts or blockgroups. To generate a population at the level of census blockgroups, a mapping between PUMA ⇔ census blockgroups is required. Similarly, to invoke multilevel controls for different variables of interest, mapping between all spatial resolutions involved is required. For example, if the user has a few variables available at the blockgroup level and a few others at the level of census tract, PopGen 2.0 needs the mapping between PUMA ⇔ Census Tract ⇔ Census Blockgroup. This mapping is to be provided in separate files as discussed below:

Note:

1. Upper Spatial Resolution: More Aggregate Detail, Region Level.
2. Lower Spatial Resolution: More Disaggregate (finer) Detail, Geo Level.

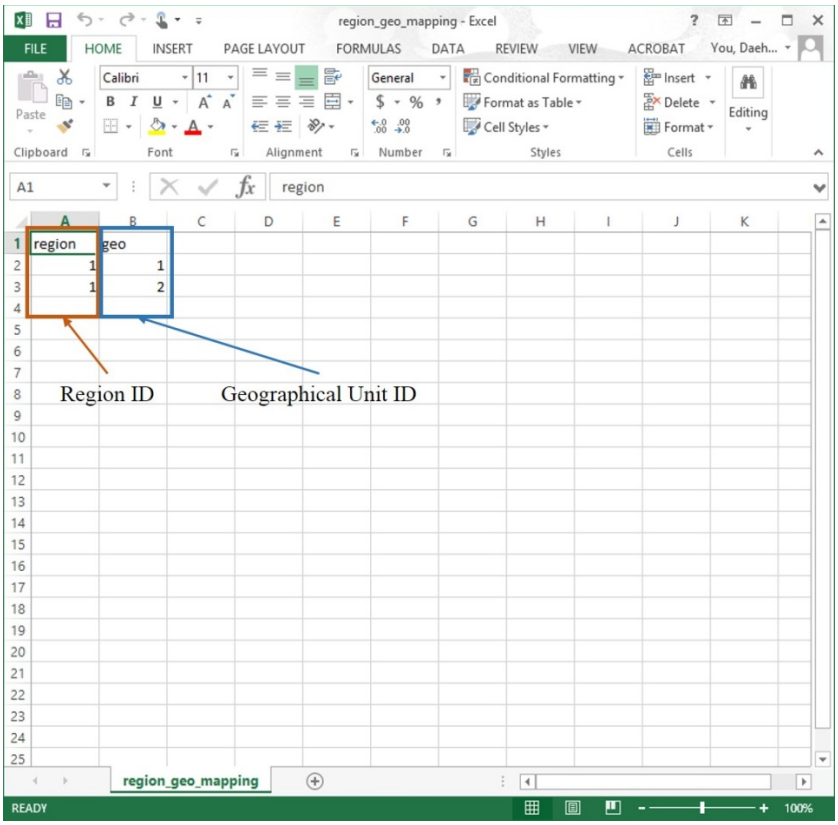


Figure 3.1. Mapping between regions and individual geographical units.

1. **Mapping Information between region (upper spatial resolution) and geo (lower spatial resolution):** When controlling for variables at multiple geographic resolutions, mapping information is needed to match regions with geographical units (geos). Since region is the upper spatial resolution compared to a geographical unit, the mapping is essentially one-to-many. Figure 3.1 shows an example of this input data file. Column names used here should also be furnished in the pertinent section in the configuration file (*inputs → column_names → region/geo*).

2. **Mapping between spatial resolutions in geo (lower spatial resolution) and sample data:** Mapping between spatial resolutions between the *geo* and *sample data* is required in order to: i) satisfy marginal controls and ii) draw households

from sample data (geographical resolution) and assign them to geographical resolution observed in *geos* as synthetic population. Figure 3.2 shows an example of this input data file. The first and second columns in the figure are for the spatial representation of geo and sample data respectively. Column names used here should also be furnished in the pertinent section in the configuration file (*inputs* → *column_names* → *sample_geo/geo*).

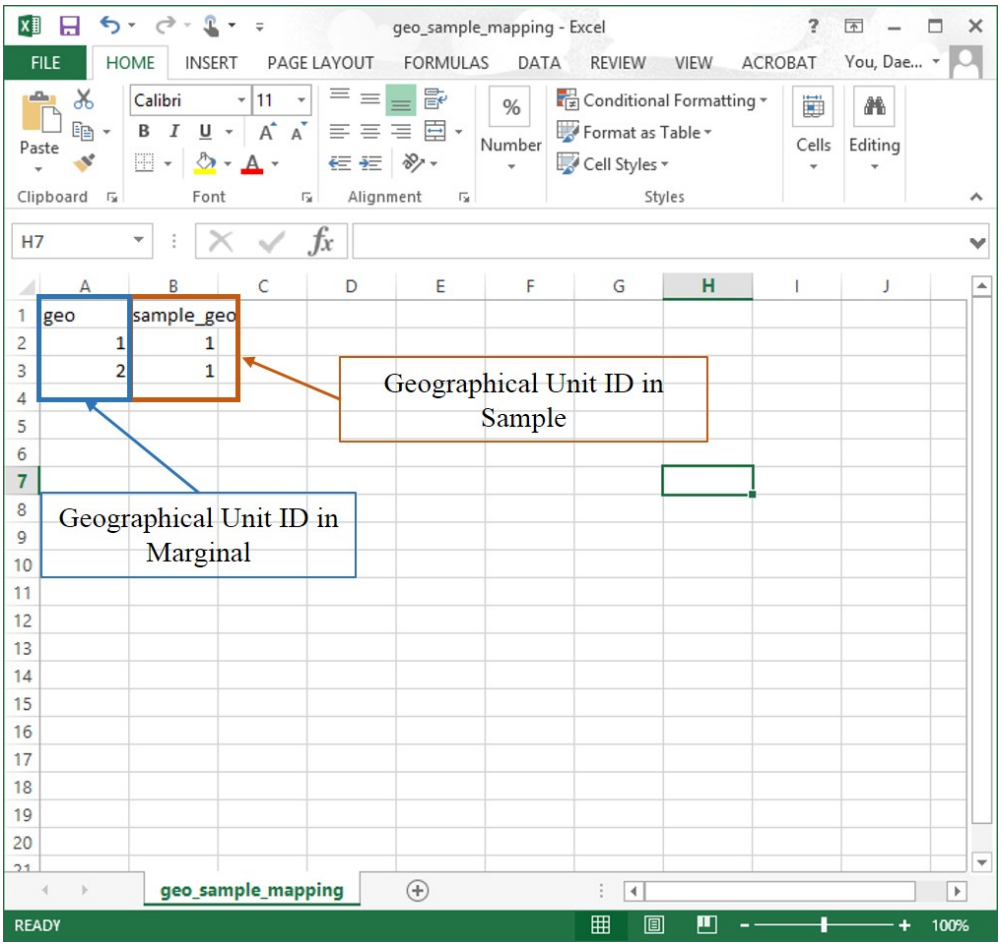


Figure 3.2. Mapping between marginal and sample geographical unit levels.

3. **Mapping between spatial resolutions in region (upper spatial resolution) and sample data:** Similar to the mapping described above, a mapping is required between spatial resolutions between the *region* and sample data, to satisfy marginal controls at the upper spatial resolution. Figure 3.3 shows an example of this input data file. If no correspondence is needed between spatial representation of the region and the sample data, a value of **'-1'** should be provided here. In the example discussed here, there is only one region with two geos in it, which are already mapped to the spatial resolution in the sample data. Therefore, no additional mapping is necessary. Column names used here should also be furnished in the pertinent section in the configuration file (*inputs* → *column_names* → *region/sample_geo*).

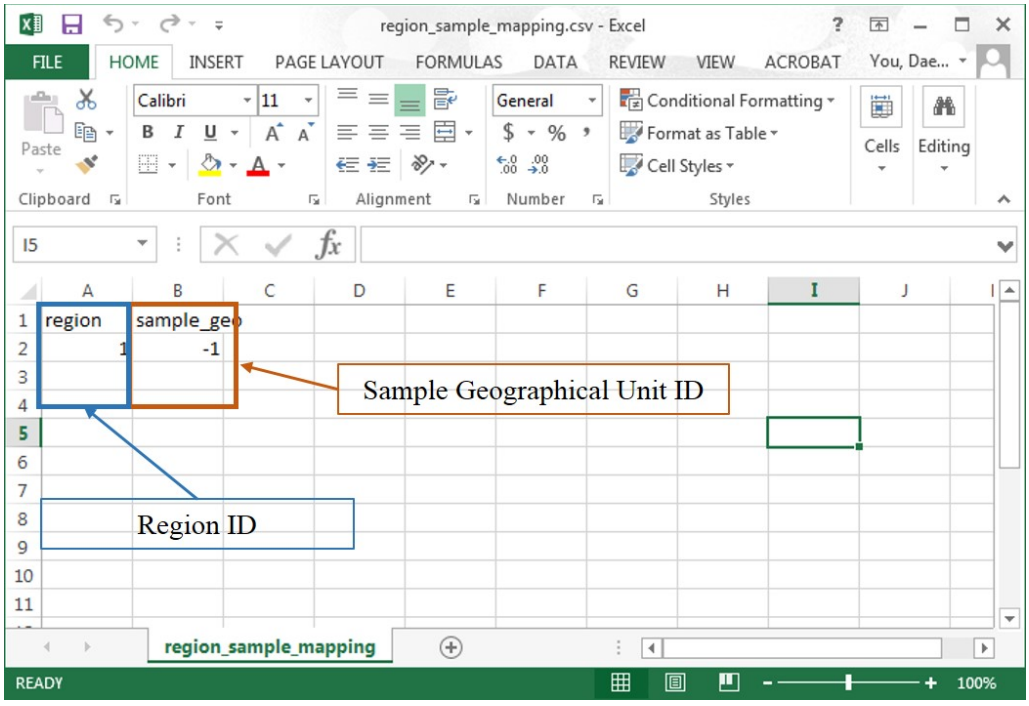


Figure 3.3. Mapping between region and sample geographical unit levels.

Region (Upper Spatial Resolution) Level Marginal Data

This section details how to prepare marginal data at the 'region' level, to be used as controls in the synthetic population generation. The input data should include the following information:

a. Names of the control variables.
b. Categories of the control variables.
c. Marginal distributions of the control variables.

1. **Household marginal distributions at the region level:** Household-level marginal data at the *region* level should be provided in this file. Figure 3.4 shows an example for a control variable corresponding to type of the household (labeled *rhhldtype*) with three categories at the region level (only one region for synthesis). In this example, name of the variable is defined in *row 1*, categories are defined in *row 2*, spatial resolution of the control variable is given in *row 3* and the distribution (number of households in each category) of the variable is provided in row 4. More control variables can be included by adding columns to the right. Controls in multiple regions can be incorporated by adding rows with marginal distributions for all the variables of interest. File name and path corresponding to this data should be provided under *inputs* → *location* → *marginal* → *region* → *household* in the configuration (*.yaml*) file.

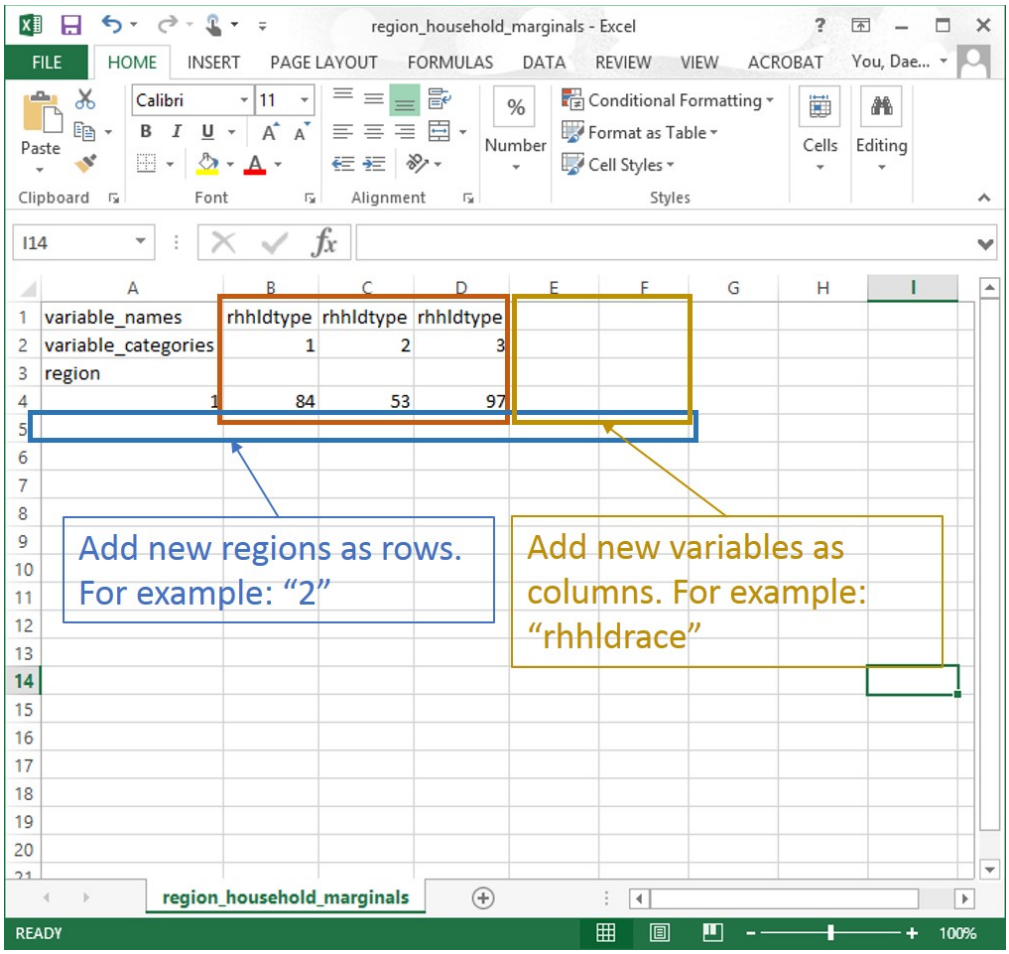


Figure 3.4. Household marginal input data at the region level.

2. **Person-level marginal distributions at the region level:** If person-level variables of interest are available at the region level, these variables should be provided in this file in order to be used as controls. The structure of the data is the same as that of household-level marginal distributions (shown in Figure 3.4). However, a separate file should be prepared and provided for person-level control variables. Note that the number of regions (rows) for which marginal distributions are provided should be consistent, between household-level and person-level data at the region level. File name and path corresponding to this data should be provided under *inputs* →

location → *marginal* → *region* → *person* in the configuration (*.yaml*) file.

3. **Groupquarter marginal distribution at the region level:** If groupquarter-level variables of interest are available at the region level, these variables should be provided in this file in order to be used as controls. The structure of the data is the same as that of household-level marginal distributions (shown in Figure 3.4). File name and path corresponding to this data should be provided under *inputs* → *location* → *marginal* → *region* → *groupquarter* in the configuration (*.yaml*) file.

Note: All the variables names at the region level should be prefixed with the letter *r*.

Geo (Lower Spatial Resolution) Level Marginal Data

This section details how to prepare marginal data at the 'geo' level to be used as controls in the synthetic population generation. The input data should include the following information:

a. Names of the control variables.
b. Categories of the control variables.
c. Marginal distributions of the control variables.

Structure of this data is exactly same as the one depicted for *region* level. The only difference is that the geographical resolution should be identified as *geo* (Figure 3.5, row 3).

1. **Household-level marginal data:** Household-level marginal data at the 'geo' level should be provided in this file. Figure 3.5 shows an example for a control variable corresponding to type of the household (labeled *hhldtype*) with two categories at the geographic (*geo*) unit level. Information regarding the variable is furnished along the columns in the spreadsheet, while information pertaining to geos is provided along the rows. Unlike at the *region* level, there are two distinct *geos* here. Marginal distribution for the variable *hhldtype* is provided in different rows for each of these geographical units. Please note that the same type of variable can constitute of different categories at the *region* and *geo* levels. File name and path corresponding to this data should be provided under *inputs* → *location* → *marginal* → *geo* → *household* in the configuration (*.yaml*) file.

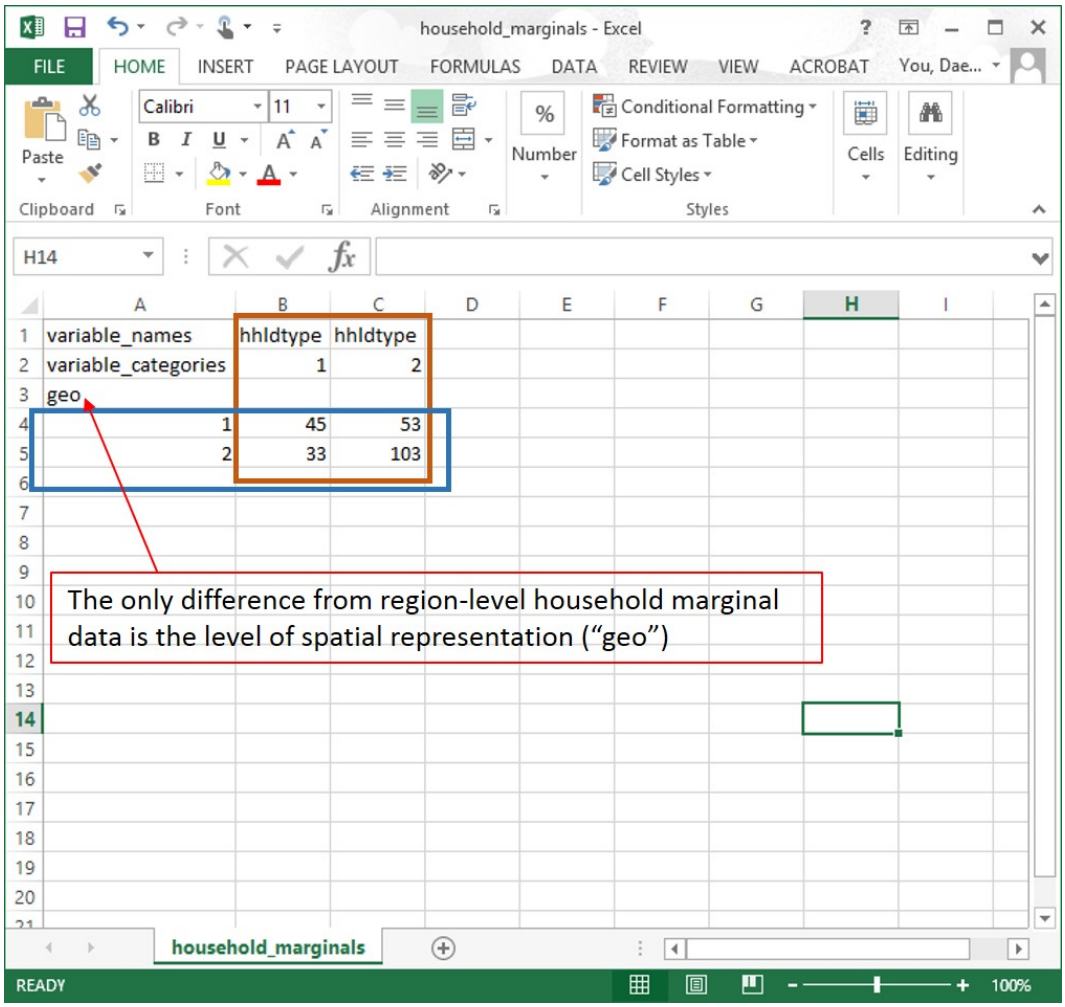


Figure 3.5. Household-level marginal distributions at the geographical unit level.

2. **Person-level marginal data:** Person-level marginal data at the 'geo' level should be provided in this file. Figure 3.6 shows an example for a control variable corresponding to type of the person (labeled *ptype*) with three categories at the geographic (*geo*) unit level. Note that the number of geos (*rows*) for which marginal distributions are provided should be consistent between household-level and person-level data at the geo level. File name and path corresponding to this data should be

provided under *inputs* → *location* → *marginal* → *geo* → *person* in the configuration (*.yaml*) file.

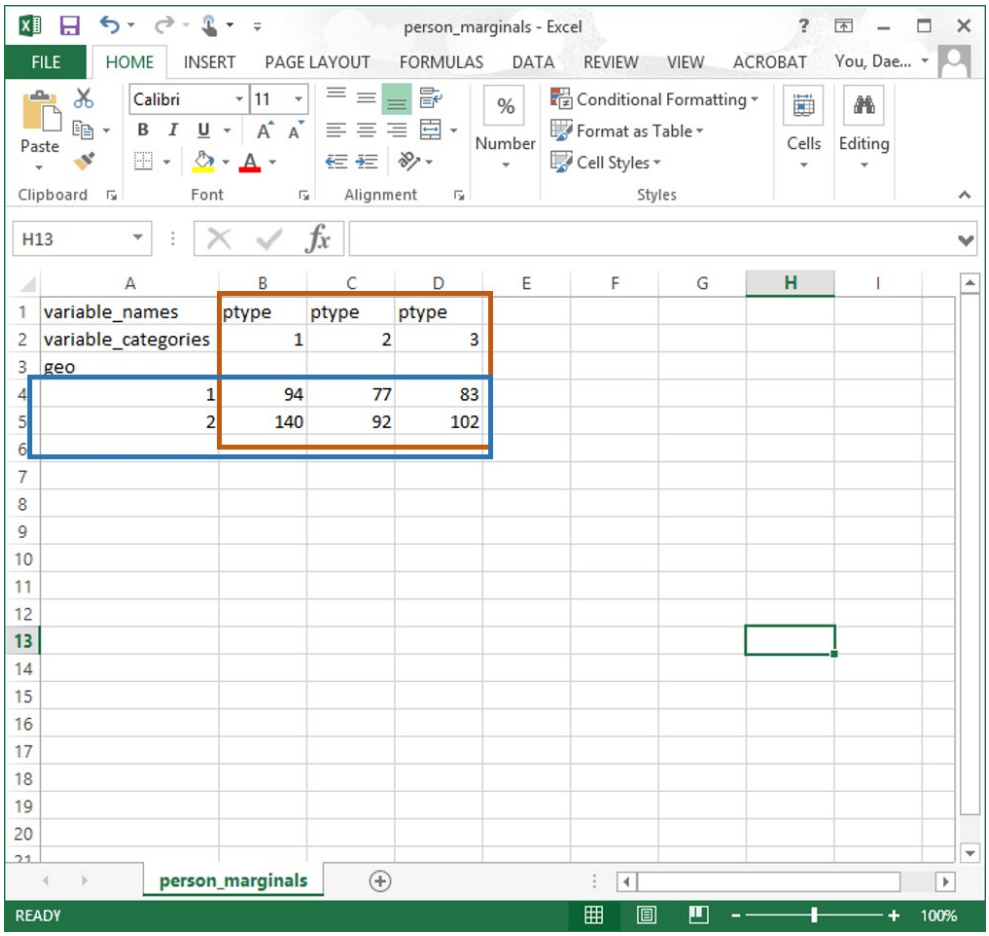


Figure 3.6. Person-level marginal distributions at the geographical unit level.

3. **Groupquarter-level marginal data:** If groupquarter-level variables of interest are available at the geo level, these variables should be provided in this file in order to be used as controls. The structure of the data is the same as that of household-level/person-level marginal distributions (shown in Figure 3.5 and Figure 3.6). File name and path corresponding to this data should be provided under *inputs* → *location* →

marginals → *geo* → *groupquarters* in the configuration (.yaml) file.

Sample Input Data

This section details how to prepare household-, person-, and groupquarter-level sample data for use in the population synthesizer. Households will be randomly drawn from the sample data based on household weights computed from the IPF and reweighting procedures. The input data should include the following information:

a. Unique id for each record.
b. ID for the sample geography to which the household belongs.
c. Different variables of interest for the record (household/person/groupquarter).

1. **Household sample data:** Sample data at the household level should be provided in this file. Each records in the input data **must** include unique household ID, sample geographical unit ID and all household-level control variables at the region (upper spatial resolution) as well as geo (lower spatial resolution) levels. Figure 3.7 shows an example of a household sample data file. In this example, the first column corresponds to the unique household ID, second column corresponds to the sample geographical unit ID. Third and fourth columns correspond to the control variables at geo and region levels respectively. Each column corresponding to a control variable essentially comprises of category values listed in the household marginal input data. For example, the column *hhldtype* contains values 1 or 2 for each household as the marginal distribution for this variable (at the geo level) comprises of these two categories only (see Figure 3.5). Similarly, the column *rhhldtype* contains values 1, 2 or 3 for

each household as the marginal distribution for this variable (at the region level) comprises of these three categories only (see Figure 3.4). Perfect consistency should be maintained in variable names provided in the sample data, marginal data and the configuration file. File name and path corresponding to this data should be provided under *inputs* → *location* → *sample* → *household* in the configuration (*.yaml*) file.

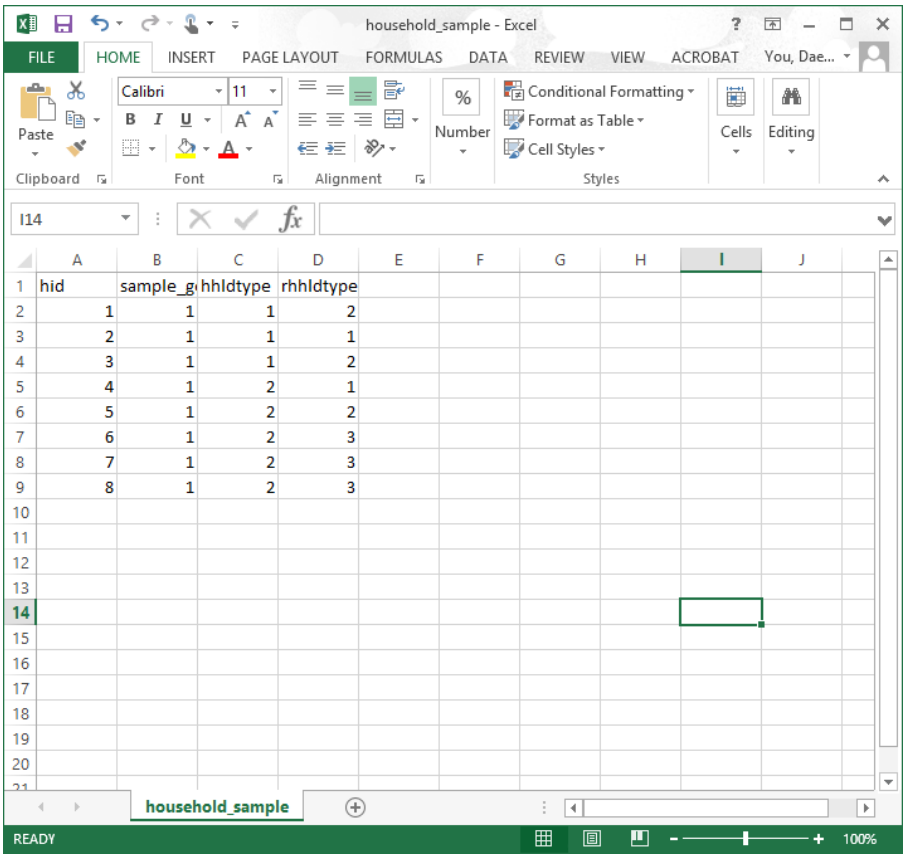
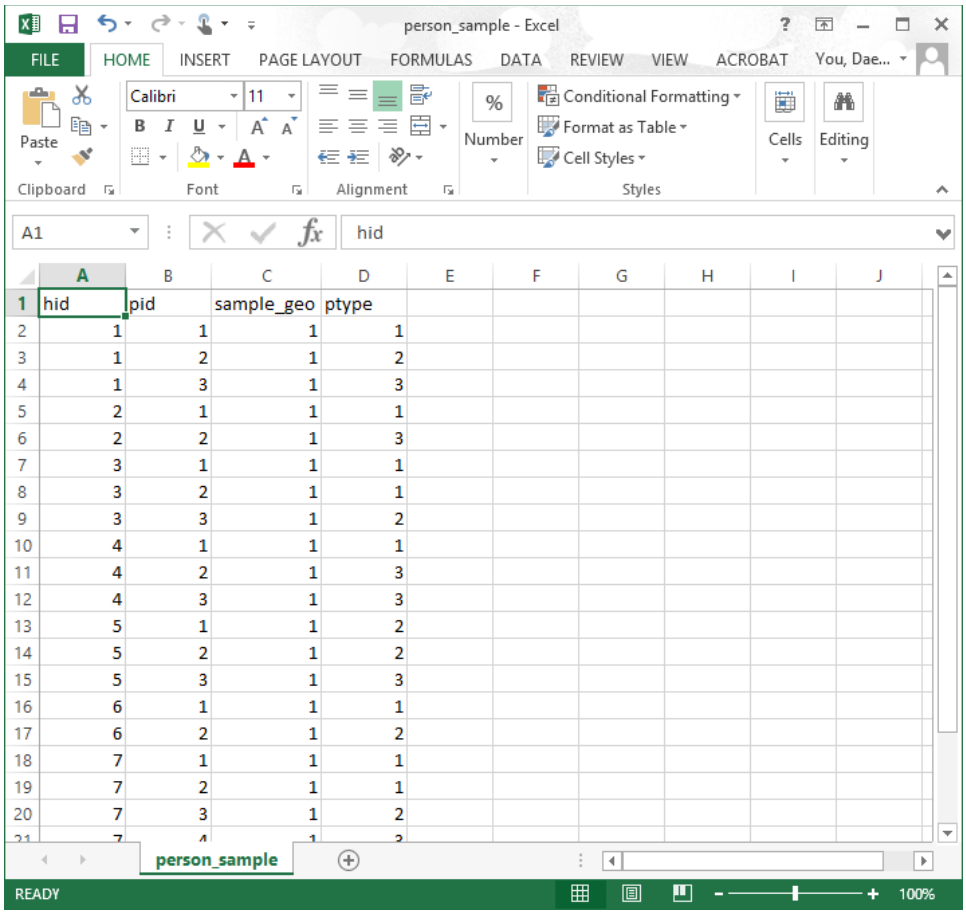


Figure 3.7. Household sample data at the sample geographical unit level.

2. **Person-level sample data:** Sample data at the person level should be provided in this file. The input data **must** include unique household ID, person ID, sample geographical unit ID and all person-level control variables at the region (upper

spatial resolution) as well as geo (lower spatial resolution) levels. Household ID and person ID combined together act as the unique ID for each person (row). Figure 3.8 shows an example of a person sample data file. Each column corresponding to a control variable essentially comprises of category values listed in the person marginal input data. For example, the column *ptype* contains values 1, 2 or 3 for each person, as the marginal distribution for this variable (at the geo level) is comprised only of these three categories (see Figure 3.6). Perfect consistency should be maintained in variable names provided in the sample data, marginal data and the configuration file. File name and path corresponding to this data should be provided under *inputs* → *location* → *sample* → *person* in the configuration (*.yaml*) file.



	A	B	C	D
1	hid	pid	sample_geo	ptype
2	1	1	1	1
3	1	2	1	2
4	1	3	1	3
5	2	1	1	1
6	2	2	1	3
7	3	1	1	1
8	3	2	1	1
9	3	3	1	2
10	4	1	1	1
11	4	2	1	3
12	4	3	1	3
13	5	1	1	2
14	5	2	1	2
15	5	3	1	3
16	6	1	1	1
17	6	2	1	2
18	7	1	1	1
19	7	2	1	1
20	7	3	1	2

Figure 3.8. Person-level sample input data at the sample geographical unit level.

3. **Groupquarter sample data:** Sample data at the groupquarter level should be provided in this file. The input data **must** include unique household ID, sample geographical unit ID and all groupquarter-level control variables at the region (upper spatial resolution) as well as geo (lower spatial resolution) levels. The structure of this file is exactly same as that of household-level sample data (see Figure 3.7). File name and path corresponding to this data should be provided under *inputs* → *location* → *sample* → *groupquarter* in the configuration (*.yaml*) file.

Data Preparation: Configuration File

To run a population synthesis project in PopGen 2.0, the user first needs to set up a configuration file (*.yaml*) that contains detailed information regarding the scenario settings, parameter settings, input and output data. Figure shows a part of a sample configuration file.

A configuration (*.yaml*) can be segmented into 5 parts. The user is expected to provide all of the required information corresponding to a project. Each of the segments in the configuration file are explained below:

1. *Synthesize* – Set to ***True*** if the purpose of the project is to generate a synthetic population, ***False*** otherwise.
2. *Name* – Name of project.
3. *Location* – Path of the project folder (For example *C:/SynTest/Houston/*. Please note that the folder separators are forward slashes).
4. *Inputs* – Information regarding input data
    a. *entities* – All *entities* in the synthetic population should be listed here (household, person, and groupquarter).
    b. *housing_entities* – Household-level entities such as household and/or groupquarter should be listed here.
    c. *person_entities* – Person-level entities should be listed here.
    d. *column_names* – Column names used for identifying information should be listed under this section
        i. *hid* – Column name that contains the household ID in household-level input data.
        ii. *pid* – Column name that contains the person ID in person-level input data.
        iii. *geo* – Column name that contains the geographical unit ID.

iv. *region* – Column name that contains the region-level zone ID.
v. *sample_geo* – Column name that contains the geographical unit ID in the sample data.

e. location – Path and file names of all input data should be listed under this section
   i. *geo_corr_mapping* – Geographical correspondence information for matching one spatial resolution unit to a different spatial resolution unit (for example County ⇔ Census Tract). There are three types of spatial mappings pertinent to a population synthesis with controls at multiple geographical resolutions: i) geographical unit to sample geographical unit (*geo_to_sample*), ii) region to sample geographical unit (*region_to_sample*), and iii) region to geographical unit (*region_to_geo*).
   ii. *sample* – household-, groupquarter- and/or person-level sample data.
   iii. *marginals* – household- and person-level marginal distributions at both region-, and geo-levels.

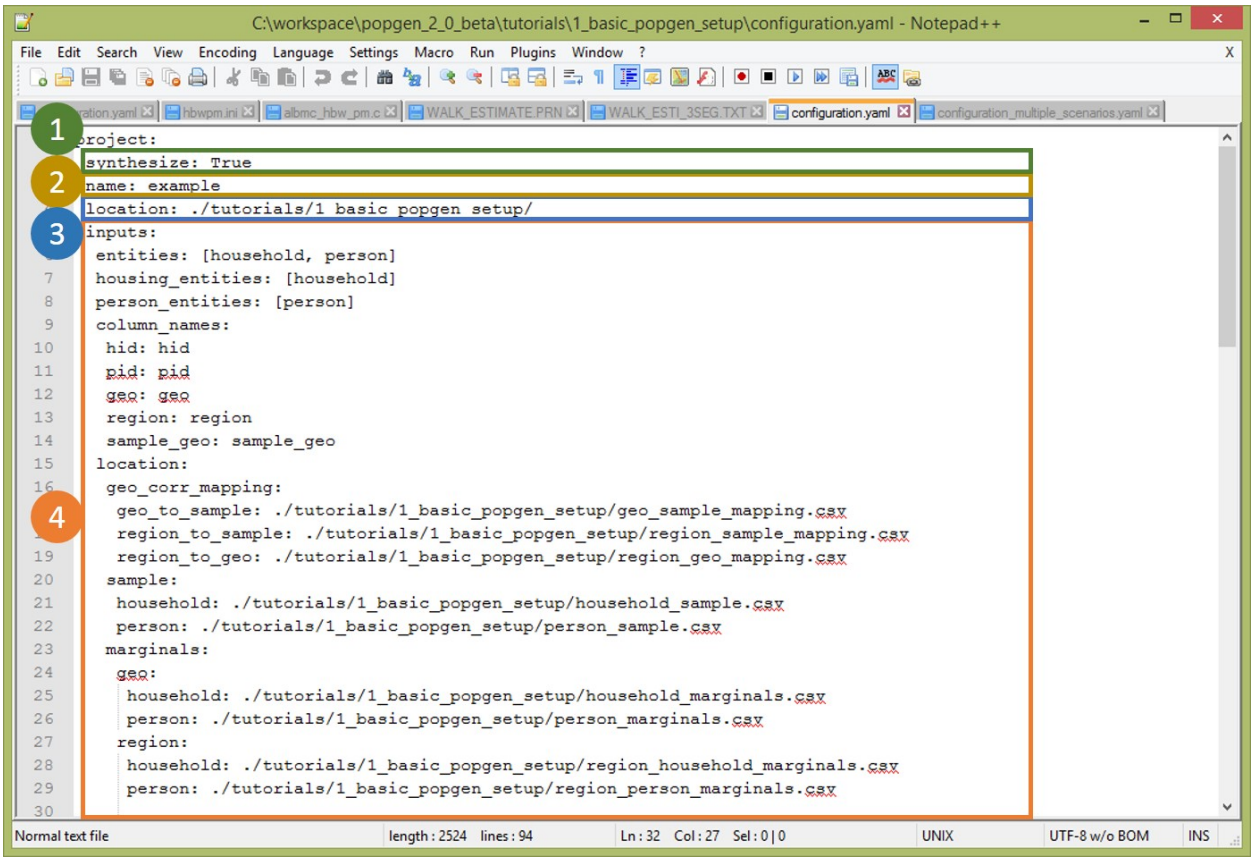


Figure 3.9. The first half of a configuration in PopGen 2.0.

5. Scenario – Scenario settings for the project should be provided here (see Figure 3.10)

    a. *description* – Brief description of the scenario.
    b. *apply_region_controls* – ***True*** if controls are applied at multiple geographical resolutions (region and geo). ***False*** otherwise.
    c. *control_variables* – List of all control variables for households, persons, and/or groupquarters. If *apply_region_*controls are set to ***True***, control variables at the region level should also be listed here.
    d. *parameters* – Settings for IPF procedure, reweighting, and drawing household samples should be provided here
        i. *ipf* – Settings for the IPF procedure such as tolerance, number of iterations, zero marginal correction factor,

rounding procedure, and archive performance frequency.

ii. *reweighting* – Settings for the reweighting procedure such as, tolerance, number of inner/outer iterations and archive performance frequency. Either *Entropy* or *IPU* procedures can be selected for reweighting.

iii. *draws* – Settings for the sample drawing procedure such as p-value tolerance, number of iterations, and seed number.

e. geos_to_synthesize – This setting provides the option to select specific regions and/or geographical units to synthesize population. To select all zones, set the option all_ids to True. If all_ids is set to False, PopGen 2.0 will synthesize population only for the zones listed under ids (see Figure 3.10) .

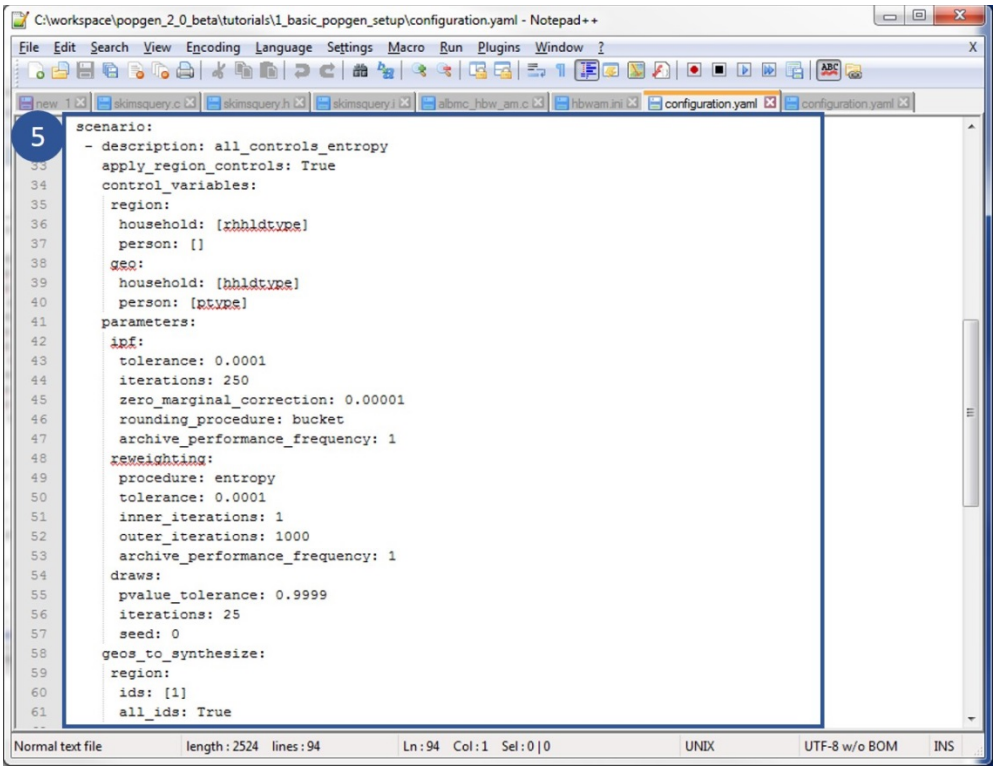


Figure 3.10. The second half of a configuration in PopGen 2.0.

f. *outputs* – Settings to define/control the outputs generated by the synthesizer (see Figure 3.11)
   i. *performance* – This setting provides the user with control over the performance metrics generated by the program for different procedures (IPF, reweighting, and drawing). Performance metrics will only be generated for procedures listed here.
   ii. *weights* – This setting is to export the weights generated during the synthesis. If *export* is set to ***True***, the weights will be exported. If *collate_across_geos* is set to ***True***, the weight for a household across all geos will be summed up. If it is set to ***False***, the weights for a household in each geography will be output separately.
   iii. *multiway* – Using this setting, the user can generate cross-tabulations of control variables x geographical units. If multiple variables are provided with the *[]*

brackets, a multiway cross tabulation will be generated (for example person age by gender).

iv. *summary* – The setting is to define output file names to populate summaries of synthetic population at region and geographical unit levels (separate file names should be provided for each level of the spatial resolution).

v. *synthetic_population* – The setting is to define output file names to for the synthetic population (separate file names should be provided for household and person levels).

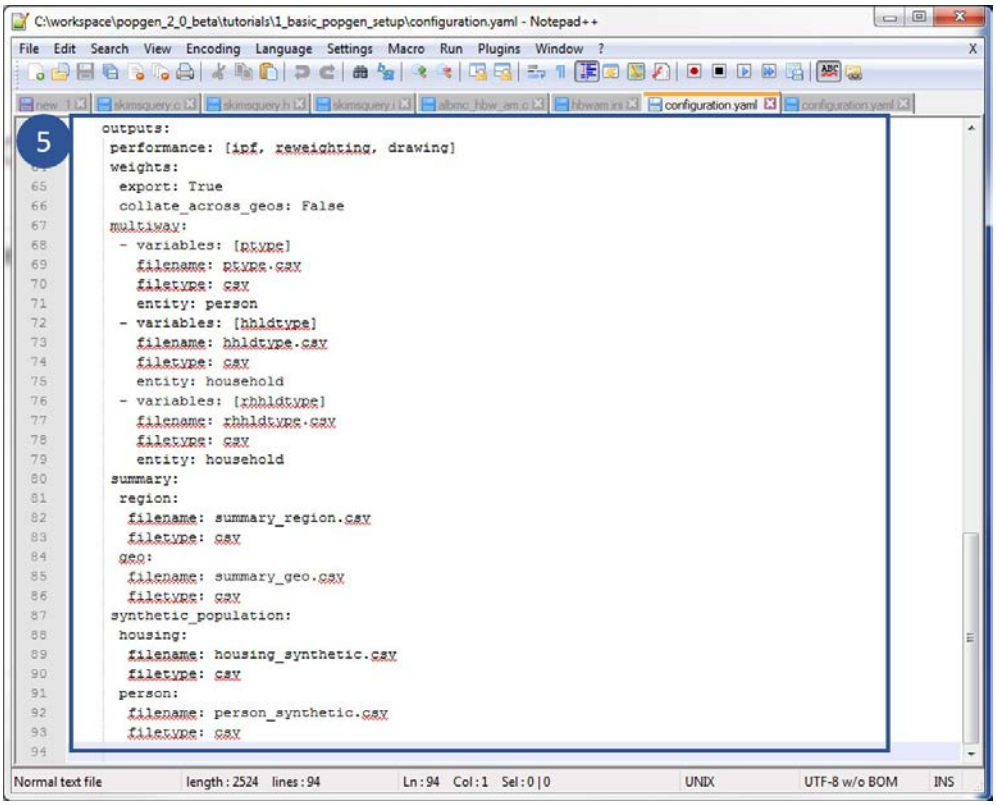


Figure 3.11. Part of output information in a configuration in PopGen 2.0.

Configuration File Example

A copy of sample input data and a configuration file are provided below, consistent with the formats specified in previous sections. We will now examine the configuration file labelled *configuration.yaml* to demonstrate the parameters and inputs that were explained in the previous section.

1. *synthesize: True* - This parameter invokes PopGen to start the population synthesis. This parameter should always have its value set to ***True***. Refer Figure 3.12.

2. *name: Colorado_Synthetic_Population* - The name of the project in this example is *Colorado_Synthetic_Population*. Users may choose an appropriate name per their discretion as long it is a single string with no spaces and no special characters. Refer Figure 3.12.

3. *location: C:/Users/kmehan3/Desktop/….* - Specify the project location, i.e. where all the files are stored and where the output needs to generated. Refer Figure 3.12.

4. *inputs:…* **-** These refer to the respective variable names, matched to the variable names within the respective input files. Refer Figure 3.12. Reference Section 3.b.i-4a:4d

```
project:
 synthesize: True
 name: Colorado_Synthetic_Population
 location: C:/Users/kmehan3/Desktop/PopGen_2_0_input/
 inputs:
  entities: [household, person]
  housing_entities: [household]
  person_entities: [person]
  column_names:
   hid: hid
   pid: pid
   geo: geo
   region: region
   sample_geo: sample_geo
```

Figure 3.12. Project Wide Setting.

5. *location:...* - These are the locations for the input data files. Refer Figure 3.13. Reference Section 3.b.i- 4e.

```
   location:
    geo_corr_mapping:
     geo_to_sample: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/geo_sample_mapping.csv
     region_to_sample: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/region_sample_mapping.csv
     region_to_geo: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/region_geo_mapping.csv
    sample:
     household: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/household_sample.csv
     person: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/person_sample.csv
    marginals:
     geo:
      household: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/household_marginals.csv
      person: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/person_marginals.csv
     region:
      household: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/region_household_marginals.csv
      person: C:/Users/kmehan3/Desktop/PopGen_2_0
_input/region_person_marginals.csv
```

Figure 3.13. Input File Locations.

6. *appy_region_controls: True* **-** This parameter prompts PopGen to take into account specific control variables while synthesizing the population. Refer Figure 3.14.
7. *control_variables: region:* **-** This is where the control variables for the region-level data (the aggregate geographic level) should be specified. In this example, we are controlling for income, household size, presence of children, householder age and number of workers at the household level. At the person level, we are controlling for employment, age, gender and race. Refer Figure 3.14.
8. *geo:* - This is where we specify control variables for the geo-level (the more disaggregate geographical resolution) data. In this example, at the household level, we are controlling for traffic analysis zone aggregate income. At the person level, we are controlling for total population. Refer Figure 3.14.

```
    apply_region_controls: True
    control_variables:
     region:
      household:
[hhldinc,hhldsize,hhldtype,childpresence,hhldrage,workers]
      person: [employment,agep,gender,race]
     geo:
      household: [tazinc]
      person: [totpop]
```

Figure 3.14. Control Variables.

9. *parameters:..* - This is where we specify the performance parameters for the ipf and ipu procedures. These settings remain constant for most applications, but can be changed as necessary depending on the computational power of the machine on which PopGen 2.0 is being run. Refer Figure 3.15. Reference 3.i.b-5d

```
parameters:
 ipf:
  tolerance: 0.0001
  iterations: 250
  zero_marginal_correction: 0.00001
  rounding_procedure: bucket
  archive_performance_frequency: 1
 reweighting:
  procedure: ipu
  tolerance: 0.0001
  inner_iterations: 1
  outer_iterations: 50
  archive_performance_frequency: 1
 draws:
  pvalue_tolerance: 0.9999
  iterations: 25
  seed: 0
```

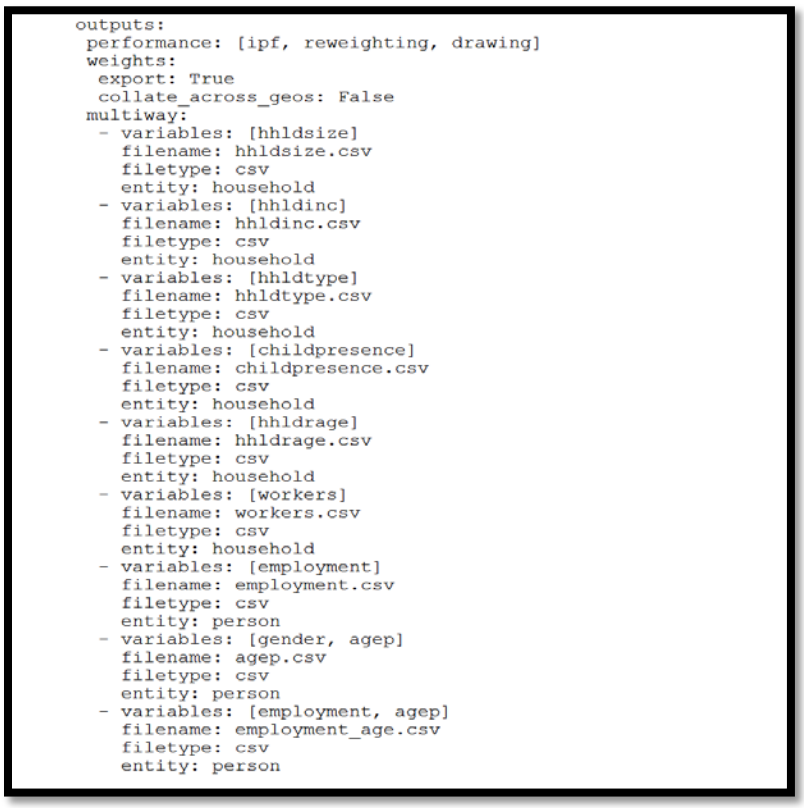
Figure 3.15. Performance Parameters.

10. *geos_to_synthesize:* - This is where we specify the region-level *ids* and geo-level *ids* for which we would like to generate the synthetic population. It should be ensured that each of the specific geo *ids* specified map to a valid region *id* as specified. If a user specifies just the *region ids* and leaves the *geo ids* field empty, a synthetic population will be generated for all geo's within a region (recommended). Refer Figure 3.16.

```
    geos_to_synthesize:
     region:
      ids: [3,4,5]
      all_ids: True
     geo:
      ids:
[1216,1217,1218,1219,1220,1221,1222,1223,1226,1227,1228]
      all_ids: True
```

Figure 3.16. Geo & Region ID's.

11. *outputs:..* - The settings under this category which include 'performance' and 'weights' remain static across projects. Refer Figure 3.17.
12. *multiway:..* - This is where users may specify individual output files they would like to see based on a particular variable or a set of variables at the level of *geo*. A user may specify the variable name under *variables* as *[varname]*, the file name under *filename*, the file type under *filetype* (in this example we use *csv* and this works well for most projects) and under *entity*, we specify what entity our variable refers to, in our example a *household* or a *person*. Refer Figure 3.17.



```
outputs:
 performance: [ipf, reweighting, drawing]
 weights:
  export: True
  collate_across_geos: False
 multiway:
  - variables: [hhldsize]
    filename: hhldsize.csv
    filetype: csv
    entity: household
  - variables: [hhldinc]
    filename: hhldinc.csv
    filetype: csv
    entity: household
  - variables: [hhldtype]
    filename: hhldtype.csv
    filetype: csv
    entity: household
  - variables: [childpresence]
    filename: childpresence.csv
    filetype: csv
    entity: household
  - variables: [hhldrage]
    filename: hhldrage.csv
    filetype: csv
    entity: household
  - variables: [workers]
    filename: workers.csv
    filetype: csv
    entity: household
  - variables: [employment]
    filename: employment.csv
    filetype: csv
    entity: person
  - variables: [gender, agep]
    filename: agep.csv
    filetype: csv
    entity: person
  - variables: [employment, agep]
    filename: employment_age.csv
    filetype: csv
    entity: person
```

Figure 3.17. Output Multiway Files.

13. *summary:* - This is where we specify the names of the output files. Refer Figure 3.18.

a. The *region* summary consists of a summary of all control variables from the synthetic population aggregated to the level of region. Summaries are generated at the region-level for variables that are controlled at the geo-level as well.
b. The *geo* summary consists of a summary of all control variables from the synthetic population aggregated to the level of geo. Summaries are generated at the geo-level for variables that are controlled at the region-level as well.
c. The *housing* summary consists of the list of generated synthetic households along with region id and geo id, and characteristics of the household, such as *hhldinc*, *hhldtype* etc.
d. The *person* summary consists of the list of generated synthetic people (i.e. population) along with their region id and geo id, the household id they belong to, and person characteristics such as age, gender and race.

```
summary:
 region:
  filename: summary_region.csv
  filetype: csv
 geo:
  filename: summary_geo.csv
  filetype: csv
synthetic_population:
 housing:
  filename: housing_synthetic.csv
  filetype: csv
 person:
  filename: person_synthetic.csv
  filetype: csv
```

Figure 3.18. Summary Files.

Running PopGen 2.0

Checklist before Execution of PopGen 2.0

The following prerequisites are required before running PopGen 2.0, please go through the list carefully before attempting to run a new project in PopGen.

1. Knowledge of Command Line Interfaces
2. Python Installation Complete
3. PopGen 2.0 Installation Complete
4. Data Preparation of Input Files
5. Data Preparation of Configuration Files

If you have gone through all the preceding steps, then you are now ready to run PopGen 2.0. If at any point during the process, you encounter errors while running PopGen, please refer to this list to ensure all steps have been followed correctly.

Running PopGen 2.0

The following steps should be adhered to run a PopGen project correctly, the steps should be followed closely taking due notice of the (upper/lower) case of the letters.

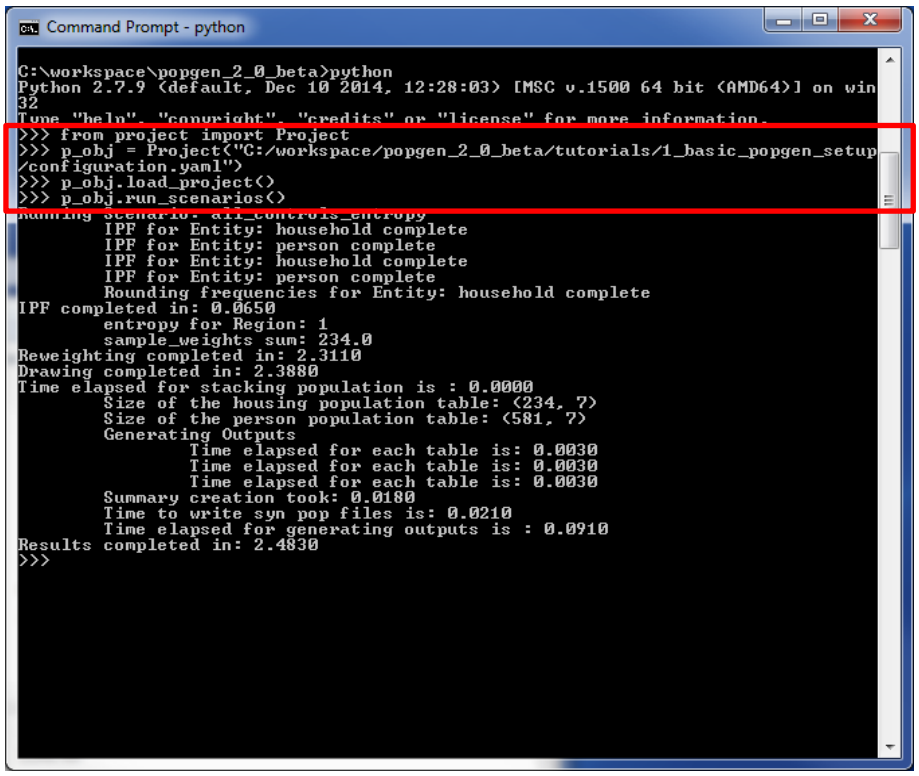


Figure 4.1. PopGen Project Window.

1. Open the Command Line Interface, if you do not know how to do so, refer to Section 1c.
2. Type **'python'**, a cursor with three arrows >>> should appear.
3. Type **'from popgen import Project'**; adhere to the correct case for the letters. In this command, *popgen* refers to the location of the *popgen* directory on your machine (as it appears on the *PYTHONPATH* system variable

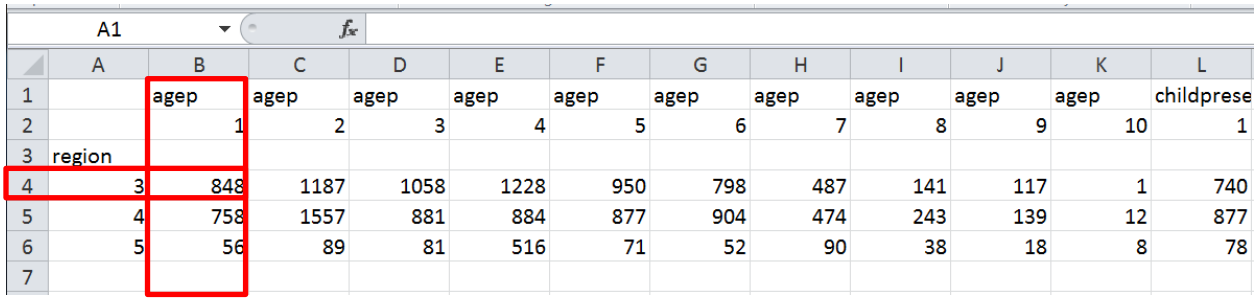
4. Type **‘p_obj = Project(‘<directory>’)’** where <directory> represents the full file path and location (Eg: *C:/workspace/PopGen/Project1/configuration.yaml*) of the YAML configuration file for your project.
5. Type **‘p_obj.load_project()’**
6. Type **‘p_obj.run_scenarios()’**, this should begin execution of PopGen
7. After PopGen has completed the generation of synthetic data, the >>> cursor should reappear, signaling that the computer is no longer busy and is able to take further commands.
8. Type **‘exit()’** to exit project and then type **‘exit’** to exit the Command Line Interface

Running PopGen 2.0: Interpreting Output Files

For the purpose of demonstration, the screenshots in the following sections are a result of running a synthetic population generation based off the configuration file example in Section 3.b.iv.

Interpreting Regional Summary

The regional summary file provides a cross tabulation of each variable (and the categories within it) with the respective count in that region. The **rows** represent the **region id** and the **columns** indicate the **variable** and its respective category.



	A	B	C	D	E	F	G	H	I	J	K	L
1		agep	agep	agep	agep	agep	agep	agep	agep	agep	agep	childprese
2		1	2	3	4	5	6	7	8	9	10	1
3	region											
4	3	848	1187	1058	1228	950	798	487	141	117	1	740
5	4	758	1557	881	884	877	904	474	243	139	12	877
6	5	56	89	81	516	71	52	90	38	18	8	78
7												

Figure 4.2. Age vs Region ID.

1. In this example, the variables presented in the screenshot are *agep* (age of person), *childpresence* (number of children).
2. To figure out the total number of people in **region 3** that fall in **age category 1 (agep 1)**, observe **B4**, which indicates that there are 848 people within the generated synthetic population that are in region 3 and of age category 1.
3. The same process can be repeated to find the total count for any region correlated with a variable.

Interpreting Geographic (Geo) Summary

The regional summary file provides a cross tabulation of each variable (and the categories within it) with the respective count in that region. The **rows** represent the **geo id** and the **columns** indicate the **variable** and its respective category.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	name	agep	agep	agep	agep	agep	agep	agep	agep	agep	agep	childprese	childprese
2	categories	1	2	3	4	5	6	7	8	9	10	1	2
3	geo												
4	1216	353	530	431	545	425	378	218	69	60		337	556
5	1217	383	809	440	446	483	440	254	127	75	8	449	584
6	1220			3	9	1		2					7
7	1221	8	11	2	41	4	7	11	2	2		10	24
8	1222	495	657	627	683	525	420	269	72	57	1	403	728
9	1223	375	748	441	438	394	464	220	116	64	4	428	557
10	1227	36	68	67	363	50	42	62	31	15	6	56	279
11	1228	12	10	9	103	16	3	15	5	1	2	12	64

Figure 4.3. Child Presence vs Geo ID.

1. In this example, the variables presented in the screenshot are *agep* (age of person), *childpresence* (number of children).
2. To figure out the total number of people in **geo 1223** that have **1 child (childpresence 1)**, observe cell **L9**, which indicates that there are 428 people within the generated synthetic population that are in geo 1223 and have one child.
3. The same process can be repeated to find the total count for any geo correlated with a variable.

Interpreting Synthetic Population

The synthetic population is generated in two separate files; one at the household level and one at the person level. Each file provides a list of the entities along with the variable (categories within the variable) that each entity belongs to.

Multiway Variable Data

In addition to the general summary and synthetic population files, PopGen can also generates summary of a particular variable or group of variables.

Group of Variables

	A	B	C	D	E	F	G	H	I	J	K
1	employme	1	1	1	2	2	2	2	2	2	2
2	agep	1	2	3	3	4	5	6	7	8	9
3	geo										
4	1216	353	530	59	217	436	360	302	156	17	2
5	1217	383	809	72	219	361	389	344	151	25	10
6	1220				1	9	1		2		
7	1221	8	11			34		5	9		
8	1222	495	657	77	340	564	432	338	162	17	3
9	1223	375	748	64	222	331	323	362	155	32	2
10	1227	36	68	1	42	288	45	34	41	13	
11	1228	12	10	1	3	80	12	3	10	3	

Figure 4.4. Employment vs Age vs Geo ID.

Presented above is a summary of the group of variables that consists of *employment* and *age* categories. In the above example, for instance to locate the number of people in **employment category 1** and **age category 3** in **geo 1223**, navigate to the column where the *employment* **row value is 1**, *agep* **value is 3** and the **row** in which **column 1 value is 1223**, which in this case yields us a result of 64. This process can be followed for any groups of variables.

Single Variable

	A	B	C	D	E	F	G	H	I
1	geo	1	2	3	4	5	6	7	8
2	1216	64	38	46	38	282	381	37	7
3	1217	36	34	37	39	334	504	27	22
4	1220	1	1			1	4		
5	1221	3		1	1	12	14	3	
6	1222	94	49	72	44	389	466	12	5
7	1223	53	30	40	43	288	495	16	20
8	1227	35	7	18	11	76	187	1	
9	1228	6	1	6	5	18	37	3	

Figure 4.5. Household Income Category 1 vs Geo ID.

Presented above is a summary of the single variable *hhldinc*. The name of the variable does not show up in the spreadsheet but is instead indicated in the name of the file, which in this case is *hhldinc.csv*. In the above example, for instance to locate the number of in people **household income category 1** in **geo 1223,** navigate to the column where the **column value in row 1** is '1' and the **row** in which **column 1 value is 1223**; which in this case yields us a result of 55. This process can be followed for any variable summary file.

Person Synthetic Data

The person synthetic data file has a unique record for each person in the population along with their region id and geo id, the household id they belong to, and their characteristics such as age, gender, race, and employment.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	unique_pe	geo	unique_id	pid	sample_ge	agep	gender	race	employme	relate	totpop	hid	entity
2	0	1216	1	1	400	8	1	1	2	0	1	38	person
3	1	1216	1	2	400	8	2	1	3	1	1	38	person
4	2	1216	1	3	400	2	1	1	1	7	1	38	person
5	3	1216	2	1	300	4	1	1	2	0	1	544	person
6	4	1216	2	2	300	4	2	1	4	1	1	544	person
7	5	1216	2	3	300	4	1	1	2	10	1	544	person
8	6	1216	3	1	902	6	1	2	4	0	1	641	person
9	7	1216	3	2	902	5	2	2	2	1	1	641	person
10	8	1216	3	3	902	3	1	2	4	2	1	641	person

Figure 4.6. Unique Person ID.

1. *unique_person_id* represents the unique id of each person that has been generated by PopGen
2. *geo* represents the geo which the person belongs to
3. *unique_id_in_geo* represents the unique household ID assigned by PopGen. Starts at 1 and increases sequentially to h (for each lower level geographical unit), where h is the total number of synthetic households generated within the geo
4. *pid* represents the id of the person from the sample data
5. *sample_geo* represents the geographical unit (PUMA) ID for the household from the sample data
6. *agep* the numerical value under this column represents the age category which they belong to, which in this case can be from 1-8

7. *gender* the numerical value under this column represents the gender category which they belong to, which in this case can be 1-2, which for example may be used to represent M-F.
8. *race* the numerical value in this column represents the racial category they belong to belong to, in which case each numerical value represents a particular race.
9. *employment* represents the employment status of the person, each numerical value can represent a different status
10. *relate* represents the relationship of the person to the household head. A value of '0' points to self.
11. *totpop* the value of this field will always be 1, since each person represents a total population of 1
12. *hid* represents the household id from the original sample data to which the person belongs to
13. *entity* the value of this field will always be person since this synthetic data file only consists of people

Housing Synthetic Data

The housing synthetic data provides a cross tabulation of each unique household to the variable (and the categories within it) which it belongs to.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	unique_h	geo	unique_id	sample_g	hhldtype	hhldinc	hhldsize	childprese	hhldfam	hhldrage	workers	tazinc	hid	entity
2	0	1216	1	400	1	5	3	2	1	2	2	2	38	household
3	1	1216	2	300	1	7	3	2	1	1	3	3	544	household
4	2	1216	3	902	1	3	7	1	1	1	2	1	641	household
5	3	1216	4	902	1	3	7	1	1	1	2	1	641	household
6	4	1216	5	502	5	1	2	2	2	2	1	1	859	household
7	5	1216	6	815	1	5	7	1	1	1	3	2	1015	household
8	6	1216	7	815	1	5	7	1	1	1	3	2	1015	household
9	7	1216	8	815	1	5	7	1	1	1	3	2	1015	household

Figure 4.7. Housing Data.

1. *unique_housing_id* represents the unique household id of each household that has been generated in the synthetic population
2. *geo* represents the geo which the person belongs to
3. *unique_id_in_geo* represents unique household ID assigned by PopGen. Starts at 1 and increases sequentially to h (for each lower level geographical unit), where h is the total number of synthetic households generated within the geo
4. *sample_geo* represents the geographical unit (PUMA) ID for the household from the sample data
5. *hhldtype* the numerical value under this column represents the household type to which the household belongs to
6. *hhldinc* the numerical value under this column represents the household income category to which the household belongs to
7. *hhldsize* the numerical value under this column represents the size of the household by number of people in the household
8. *childpresence* the numerical value in this column represents the whether or not children are present in the household, a simple Y/N can be represented by 1 or 2
9. *hhldfam* the type of the family
10. *hhldrage* the age of the household head categorized into 'below 65' (category 1) or '65 or above' (category 2)
11. *workers* the numerical value in this column represents the number of workers in the household
12. *tazinc* the numerical value of this represents the TAZ income zone to which the household belongs to

13. *hid* represents the household id from the original sample data to which the household belongs to
14. *entity* the value of this field will always be household since this synthetic data file only consists of households

Some Notes:

- The combination of *geo* and *unique_id_in_geo* is the unique identifier at the household level for the synthetic population. The combination of *geo*, *unique_id_in_geo*, and *pid* is the unique identifier at the person level for the synthetic population. The unique identifier combination of *geo* and *unique_id_in_geo* can be used to merge household information to person records
- *hid* and *pid* fields help in the addition of uncontrolled variables from the ACS sample files to the synthetic population
- *unique_housing_id* and *unique_person_id* are not critical pieces of information for the synthetic population. These fields are generated by PopGen for bookkeeping purposes. These ID fields are not related with one another and cannot be used to merge household information to person records
- *sample_geo* represents PUMA ID. This field might not be critical for the synthetic population

Appendix

Download and Installing Dependencies

If the dependencies did not auto-install, you may have to manually install them. The following dependencies are required for PopGen:

1. NumPy 1.9.2
2. SciPy 0.16.1
3. Pandas 0.16.1
4. PyYAML 3.11
5. Python DateUtil 2.4.2
6. Pytz 2015.2

The dependencies can be downloaded from **https://github.com/foss-transportationmodeling/popgen/archive/ Dependencies.zip**

Download and extract the dependencies to a folder named "Dependencies".

Installing Dependencies

1. Navigate to the Dependencies folder that you had downloaded earlier. Open the following folders located in *Dependencies/64bit* folder:
    a. *numpy-1.9.2+mkl-cp27-none-win_amd64*
    b. *scipy-0.16.1-cp27-none-win_amd64*
    c. *pandas-0.16.1-cp27-none-win_amd64*
2. Copy and paste the following sub-folders to *C:/Python27.9/Lib/site-packages*
    a. *numpy*

b. *numpy-1.9.2.data*

c. *numpy-1.9.2.dist-info*

d. *scipy*

e. *scipy-0.16.1.dist-info*

f. *pandas*

g. *pandas-0.16.1.dist-info*

3. Find *PyYAML-3.11.win-amd64-py2.7.exe* in the 64bit
4. Double click the ***executable file*** and follow the setup wizard to complete the *PyYAML* installation

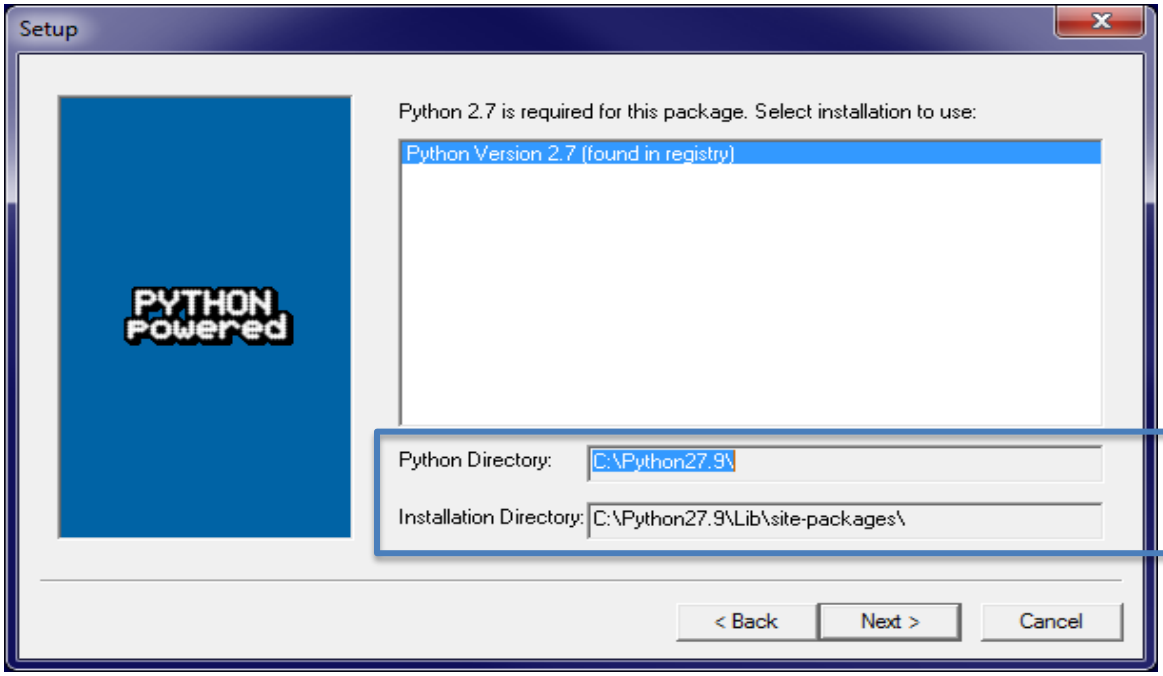


Figure 5.1. Package Extraction.

5. Install *python-dateutil-2.4.2* (same as step 5)
    a. Open the command prompt window
    b. From the command prompt window, change the directory to *python-dateutil-2.4.2*
    c. Type **'python setup.py install'** and hit enter

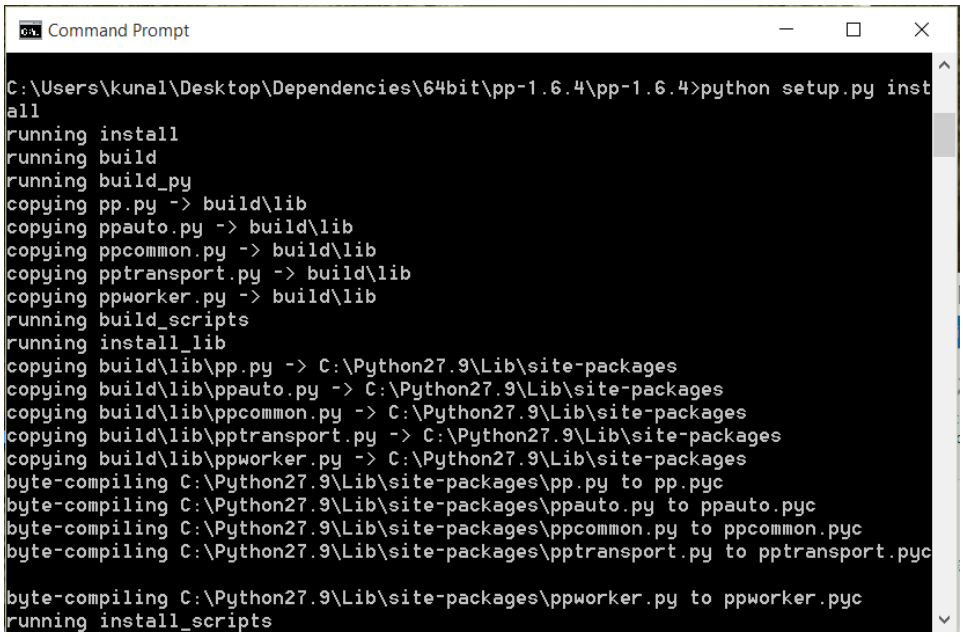


Figure 5.2. Package Installation.

6. Install *pytz-2015.2* (same as step 5)
    a. Open the ***command prompt*** window
    b. From the command prompt window, change the directory to *pytz-2015.2*
    c. Type **'python setup.py install'** and hit enter

Glossary

1. *unique_housing_id*

   Unique household ID assigned by PopGen. Starts at 0 and increases sequentially to H, where H is the total number of synthetic households generated

2. *geo*

   Geographical unit ID (lower level geographical resolution) for the household

3. *unique_id_in_geo*

   Unique household ID assigned by PopGen. Starts at 1 and increases sequentially to h (for each lower level geographical unit), where h is the total number of synthetic households generated within the geo

4. *sample_geo*

   Geographical unit ID for the household from the sample data (PUMA)

5. *hid*

   Unique Original household ID from the sample data

6. *unique_person_id*

   Unique person ID given by PopGen. Starts at 0 and increases sequentially P, where P is the total number of synthetic persons generated

7. *pid*

   Original person ID from the sample data

Resources

For any feedback related to PopGen, you may post to https://github.com/foss-transportationmodeling/popgen/issues. We are always looking for feedback to improve PopGen.

Repository Link: https://github.com/foss-transportationmodeling/popgen/releases

Command Line Reference: https://technet.microsoft.com/en-us/library/bb490890.aspx

Python FAQ: https://docs.python.org/2/faq/