PopulationSim Documentation for 2040 Demo

# Introduction

PopulationSim fills a need for TDM19 and TDM23 that cannot be provided by UrbanSim, the synthetic population tool currently used by MAPC. We only have data from UrbanSim for Massachusetts, but the model area includes southern New Hampshire and all of Rhode Island. To cover NH and RI as well as provide synthetic population estimates for Massachusetts in the near term, e.g. before we get access to the newest UrbanSim data, we are testing PopulationSim.

PopulationSim takes aggregate control data (in our case, 2040) and disaggregate seed data (in our case, base year = 2019) to estimate disaggregate person and household level data for the year of the control data. This estimated disaggregated person and household level data is what we call “synthetic population” data. PopulationSim also creates synthetic outputs at the aggregate geographic levels of the control data and the seed data (PUMAs usually) and automatically compares them to the control data at the same levels (ex. TAZ and PUMA). This allows us to look at accuracy at multiple levels.

As you may see in the GitHub repo, PopulationSim is part of the ActivitySim project which is a platform for activity-based modeling. While TDM19 and TDM23 are not ABMs, we also require disaggregate person and household level data projected into the future to predict travel behavior.

A quick note - we do not use all of the functionality of PopulationSim, so please use the following documentation with that in mind.

# Sources

* GitHub Repo: <https://github.com/ActivitySim/populationsim>
* Documentation: <https://activitysim.github.io/populationsim/>

# Set Up PopulationSim

## Environment:

First: Install VS Build - <https://visualstudio.microsoft.com/visual-cpp-build-tools/>

Then:

**Two Choices** (both require Conda and Windows)

1. Use the popsim.yml file to create a new environment.
2. Follow these instructions in your Anaconda Prompt:
   1. conda create -n popsim python=3.8
   2. activate popsim
   3. conda install pytables
   4. pip install populationsim

## Folder Structure:

In the folder you plan to run PopulationSim in, create the following folders:

1. configs
2. data
3. output

They should all be empty for now. Next, download the PopulationSim repo from github - specifically you need the run\_populationsim.py file from the example\_calm folder. See: <https://github.com/ActivitySim/populationsim/blob/master/example_calm/run_populationsim.py>

Just as a note, there are multiple copies of this file in the example folders, not all of them are the same. The one we are using here is in ¾ of the examples. Please check to see which example folder has the newest version and evaluate its use if you are not sure which to use.

Put run\_populationsim.py in the same base folder where you put the three folders above. Your file structure should look like this:

base/

* configs
* data
* output
* run\_populationsim.py

With an environment, the file structure, and runner script set, you are almost ready to go. First, let's fill in the data and configs folders.

# Required Data

In the data folder, there should be five csv files. We can break this data up into three categories:

## **Geographic Data**

* 1. geo\_cross\_walk.csv - this is a relationship file between TAZs, PUMAs, and the model region. There can only be one region but all the TAZs and PUMAs that overlap the model region should be included. Do not include PUMAs that do **not** overlap with the model region. This was created by intersecting the TAZs and the PUMAs in a GIS.
  2. Each TAZ can only have ONE PUMA associated with it in the geocrosswalk file. Therefore, there is a small script that takes the intersection of TAZs and PUMAs csv and filters it so that the PUMA associated with the largest portion of a TAZ is associated with that TAZ and no other PUMAs are.
     1. Schema: REGION, TAZ, PUMA. Each field contains the IDs that represent the pairs of TAZ and PUMA intersections. (set the region variable to 1 for all rows).

| REGION | TAZ | PUMA |
| --- | --- | --- |
| 1 | 5426 | 2501600 |
| 1 | 2841 | 2501600 |

## 

## **Control Data**

* 1. This data is estimated for the year 2040 and is required at two different geography levels: TAZ and region.
     1. The TAZ data is the same as the model input: 2040\_LandUse\_update\_11\_05\_2018\_.csv .
     2. The region csv (region\_control\_2040.csv) is the aggregation of the TAZ level data up to the whole region (sum of all numeric columns that are not IDs). A region ID is also added, its value is not important but it needs to exist.

## **Seed Data**

* 1. This data comes from the 5YR ACS Estimates PUMS data, specifically the 2015-2019 data.
  2. At the PUMA level - you will have the following files:
     1. hh\_pums\_2019\_5YR.csv
     2. person\_pums\_2019\_5YR.csv
  3. Use the R script housed here: J:\Shared drives\TMD\_TSA\Model\software\PopSim\Inputs\downloadPUMS.R to get both datasets in the correct schema.
  4. Make sure all data are integers and that all 0 person households are filtered out (e.g. NP > 0)
  5. There is a line (25) where “odd” households are filtered out - use your own judgment. The decision for these households to be excluded was based on their large size (> 12), spread of household age, and the amount of non-related household members.

# Configuration Files

There are 3 configuration files used for PopulationSim:

1. settings.yaml
   1. If you need to create a new one from scratch, copy this file from example\_calm repo on github. It will need to be modified, but having the examples helps you understand what the program is looking for. Please read the documentation before attempting to create something besides the example already used.
   2. Modifications:
      1. Algorithm/Software Configuration: Don’t change
      2. Geographic Settings:
         1. geographies: [REGION, PUMA, TAZ]
         2. seed\_geography: PUMA
      3. Tracing: hash/comment out this whole section
      4. Data Directory: Don’t Change
      5. Input Data Tables: for each of the tables in the data folder, list the name that the program wants you to use (e.g. households, persons, TAZ\_control\_data etc.), the filename, the id column, and whatever columns you need to rename. Columns you need to rename:
         1. SERIALNO: hh\_id
         2. SPORDER: per\_num
         3. HH: num\_hh
         4. Make sure TAZ in geo\_cross\_walk is also the name of the ID (of TAZs) column of TAZ\_control\_data and if it is not, rename it in the TAZ\_control\_data file.
      6. Reserved Column Names: Don’t Change
      7. Control Specification File Name: Don’t Change
      8. Output Tables: don’t change but make sure includes the following:
         1. summary\_TAZ
         2. summary\_TAZ\_PUMA
         3. expanded\_household\_ids

They are important outputs for QA/QC.

* + 1. Synthetic Population Population Output Specification: only need to change columns:
       1. synthetic\_households.csv
          1. HHNP (number of people in household 1-4p)
          2. HHEMPCAT (number of workers in household 1-3p)
          3. HHINCPADJ (household income adjusted to 2019 in four categories)
       2. synthetic\_persons.csv
          1. EMP (is this person employed?)
          2. PINCPADJ (what is the 2019 adjusted income for this person in 2040?)
    2. Models: Make sure sub\_balancing.geography = TAZ is the only sub\_balancing step included. Hash/comment out any other sub\_balancing.geography step.
  1. Documentation: <https://activitysim.github.io/populationsim/application_configuration.html#configuring-settings-file>

1. logging.yaml
   1. This one is the easiest - just copy it from the example\_calm repo on github. This does not need to change.
2. controls.csv
   1. This is the most confusing in some regards so please read the instructions here if you wish to change anything: <https://activitysim.github.io/populationsim/application_configuration.html#specifying-controls>
   2. The gist is that for every value you want estimated and aggregated in the final TAZ and PUMA level results for QA/QC, you must define the following: variable name, geography level, seed table that the data comes from (households or persons), importance weight, field name of corresponding data in the control tables, and an expression that defines the constraints of the data.
   3. See the example stored here: J:\Shared drives\TMD\_TSA\Model\software\PopSim\popsim\configs\controls.csv

# Running PopulationSim

Good job! Now you will likely want to run PopulationSim in a debugger first to flag and traceback any data issues you have. I ran it in VSCode - make sure you are in the base folder where run\_populationsim.py is or the program will not be able to find the required folders you set up (data, configs, output). If you want to run it from the command line: activate your environment, navigate to your base folder, and type in: python run\_populationsim.py

Really weird note:

If you are having trouble with something not being able to convert from float to integer but you have traced back the value and it was input as an integer (or created by the program) and is maintained as an integer until suddenly it is a float, try adding a force data type change. Only do this in extreme cases. Specifically, this was done in lines 85 and 86 (first two lines of def chooser()) in expand\_households.py in the populationsim python library.

# Run Post Processor

See: "J:\Shared drives\TMD\_TSA\Model\software\PopSim\postprocess\_popsim.ipynb"

This jupyter notebook prepares the popsim outputs to be used in the model (tdm23).