

**METROPOLITAN WASHINGTON COUNCIL  
OF GOVERNMENTS (MWCOC)**

# **MWCOC POPULATION SYNTHESIZER**

**Final Report | August 4, 2021**



55 Railroad Row  
White River Junction, VT 05001  
802.295.4999  
[www.rsginc.com](http://www.rsginc.com)

**PREPARED FOR:**  
METROPOLITAN WASHINGTON COUNCIL OF GOVERNMENTS  
(MWCOC)

**SUBMITTED BY:**  
RSG



# CONTENTS

<b>1.0 INTRODUCTION .....</b>	<b>3</b>
<b>2.0 MWCOC POPULATION SYNTHESIZER .....</b>	<b>5</b>
2.1 SOFTWARE SETUP .....	5
DIRECTORY SETUP .....	8
2.2 CONFIGURATION .....	11
CONFIGURING POPULATIONSIM .....	11
2.3 RUNNING THE SOFTWARE .....	14
2.4 POPULATION SYNTHESIS OUTPUTS.....	15
<b>3.0 DATA PREPARATION AND APPLICATION .....</b>	<b>17</b>
3.1 INPUT DATA PREPARATION .....	17
SEED SAMPLE.....	17
MARGINAL CONTROLS .....	18
SCENARIO APPLICATIONS .....	19
3.2 CONTROLS AND SETTINGS.....	20
<b>4.0 VALIDATION .....</b>	<b>22</b>
4.1 VALIDATION PROCEDURES.....	22
VALIDATION SUMMARY STATISTICS.....	22
VALIDATION CHART .....	23
FREQUENCY DISTRIBUTION PLOTS .....	23
EXPANSION FACTOR DISTRIBUTION .....	23
4.2 BASE YEAR (2018) VALIDATION .....	24
4.3 FUTURE YEAR VALIDATION.....	28

## LIST OF FIGURES

FIGURE 1 MWCOG POPULATION SYNTHESIZER DIRECTORY STRUCTURE .....	9
FIGURE 2 MWCOG RESIDENTIAL POPULATIONSIM VALIDATION – 2018 .....	25
FIGURE 3 OUTLIER TAZ - 3468.....	26
FIGURE 4 OUTLIER TAZ – 2504 .....	27
FIGURE 5 MWCOG GQ POPULATIONSIM VALIDATION - 2018 .....	27
FIGURE 6 MWCOG RESIDENTIAL POPULATIONSIM VALIDATION - 2045 .....	29

## LIST OF TABLES

TABLE 1. DATA SOURCES FOR SEED SAMPLE AND MARGINAL CONTROLS .....	18
TABLE 2: EXAMPLE HOUSEHOLD SIZE CONTROL ADJUSTMENTS.....	19
TABLE 3. MWCOG POPULATIONSIM MARGINAL CONTROLS, RESIDENTIAL .....	20
TABLE 4. MWCOG POPULATIONSIM MARGINAL CONTROLS, GROUP QUARTERS .....	21

# 1.0 INTRODUCTION

---

Historically, most travel demand forecasting models used in the U.S. have been aggregate, trip-based models that make use of zonal averages and estimate zone-to-zone flows of person and vehicle trips. Over the past couple of decades, many large cities in the U.S. have been transitioning to disaggregate, activity-based travel demand forecasting models (ABMs) that make use of demand microsimulation, which estimates travel at the level of the individual decision maker (persons and households), and which allows one to better represent the interdependencies between activities, trips, persons, time, and space. By simulating activities at the level of the individual traveler, ABMs are better able to account for policies such as flexible working arrangements/telework, pricing policies, equity analyses, and representation of non-motorized transportation modes. ABMs require disaggregate household and person socio-economic data for the *entire population* of the modeled area. Unfortunately, such data is typically available for *only a subset* of the population (e.g. U.S. Census Public Use Microdata Sample [PUMS] data or data from a regional household travel survey). Thus, to use an ABM, one needs to synthesize a regional population from known statistical distributions of the population. **Population synthesis** is the process of generating a synthetic population by expanding the disaggregate sample data to mirror known aggregate distributions of household and person variables of interest.<sup>1</sup> Population synthesis typically involves two data sources: 1) Disaggregate data for a sample of the population (e.g., PUMS or travel surveys), often called the seed data; 2) Marginal distributions for the entire modeled area (e.g., Census summary files or aggregate land use forecasts created by a regional agency). Initially, population synthesis routines made use of a technique known as iterative proportional fitting (IPF),<sup>2</sup> and allowed the user to specify controls using only one level of geography (e.g., zones). Now, as noted below, population synthesis routines make use of more complex algorithms and techniques, such as list balancing and entropy maximization, which allow the user to specify controls using multiple levels of geography.

The Metropolitan Washington Council of Governments (MWCOC) Gen3 Travel Demand Forecasting Model is being developed using the open-source ActivitySim modeling platform.<sup>3</sup> ActivitySim is a disaggregate modeling platform that simulates the travel choices of households and persons. This requires a synthetic population of households and persons representing the entire population of the modeled region. PopulationSim<sup>4</sup> was used to generate the synthetic

---

<sup>1</sup> Ram M. Pendyala and Karthik C. Konduri, "Population Synthesis for Travel Demand Modeling: Data Needs and Application Case Studies" (Using Census Data for Transportation Applications Conference, Irvine, California, October 25-27, 2011, Irvine, California, October 2011), 6, [http://onlinepubs.trb.org/onlinepubs/Conferences/2011/Census/Presentations/10-26\\_830-10amKillough/3Pendyala.pdf](http://onlinepubs.trb.org/onlinepubs/Conferences/2011/Census/Presentations/10-26_830-10amKillough/3Pendyala.pdf).

<sup>2</sup> See, for example, Richard J. Beckman, Keith A. Baggerly, and Michael D. McKay, "Creating Synthetic Baseline Populations," *Transportation Research Part A: Policy and Practice* 30, no. 6 (November 1996): 415–29, [https://doi.org/10.1016/0965-8564\(96\)00004-3](https://doi.org/10.1016/0965-8564(96)00004-3).

<sup>3</sup> ActivitySim, 2020: <https://activitysim.github.io/>

<sup>4</sup> PopulationSim 0.4.2 Wiki, 2020: <https://activitysim.github.io/populationsim/>

population for the Gen3 Model. PopulationSim is a state-of-the-art population synthesizer initially developed for the Oregon Department of Transportation (ODOT) and its partner agencies. PopulationSim uses an entropy-maximization-based, list balancing approach to speed and improve convergence to marginal controls, allows the user to specify controls at multiple levels of geography, and uses linear programming-based techniques to generate synthetic households and persons, which eliminates drawing errors (a.k.a. Monte-Carlo error). PopulationSim has been implemented in the open-source ActivitySim framework and offers detailed documentation and instructions on the installation and configuration of the software. The PopulationSim source code and technical documentation are available at the following public GitHub repository: <https://github.com/ActivitySim/populationsim>.

The MWCOC Population Synthesizer, developed based on PopulationSim, derives households and persons from the American Community Survey (ACS) Public Use Microdata Sample (PUMS) database, Decennial Census Data, and MWCOC's Round 9.1a Cooperative Forecasts of population, households, group quarters, and employment. This document describes the implementation of MWCOC's Population Synthesizer using PopulationSim. The implementation has been automated using Python scripts. All the data processing scripts are written in the Python programming language, including those associated with downloading and formatting the PUMS sample and the Census/ACS marginal totals, post-processing of outputs, and creation of validation charts and summaries. The following sections describe the software and configuration details, data preparation and application, and validation results. In particular, Software setup Section 2.1 provides instructions on the installation of the software. First-time users should read this sub-section when setting up PopulationSim. Users might also want to revisit this section in the future when a software update is available.

For readers who are not familiar with population synthesis techniques in general, or PopulationSim software specifically, we suggest they first read the PopulationSim wiki located at <https://activitysim.github.io/populationsim/>.

We also suggest that analysts who will be modifying PopulationSim inputs or settings first complete the tutorial available here:

[https://github.com/RSGInc/odot\\_PopulationSim\\_resources/blob/master/training\\_package/TrainingPackage.7z](https://github.com/RSGInc/odot_PopulationSim_resources/blob/master/training_package/TrainingPackage.7z).

## 2.0 MWCOC POPULATION SYNTHESIZER

---

This section describes the implementation and configuration of the MWCOC Population Synthesizer. First, the software setup is described followed by configuration details and instructions on running the software.

### 2.1 SOFTWARE SETUP

The MWCOC Population Synthesizer is based on PopulationSim which is implemented in the Python-based ActivitySim framework. In addition to being used for population synthesis, Python scripts are also used to automate the downloading and preparation of input data as well as the post-processing of the outputs and generation of validation summaries. These installation instructions presume a computer running the Microsoft Windows operating system. The Population Synthesizer runs in a conda environment<sup>5</sup>. This is an execution environment with a sandboxed version of Python and Python libraries needed to run PopulationSim. The creation and use of conda environments require an installation of Anaconda<sup>6</sup> (Note that Miniconda<sup>7</sup> may be used interchangeably as well). The MWCOC Population Synthesizer was created and tested on Python 3 using both Anaconda and Miniconda. Python version 2 is not supported.

**Install Anaconda3** using recommended default settings. Downloads are available at <https://www.anaconda.com/products/individual>. When installing Anaconda, the default recommended settings do not add Anaconda to the system PATH nor register Anaconda Python as the system Python. Those settings will work for the Population Synthesizer but require the use of the Anaconda Prompt (included with Anaconda) as the command shell. A user may choose to register Anaconda Python as the system Python if there are no other versions of Python on that computer. This will allow the user to call Population Synthesizer using the Windows Command Prompt. However, it is common to have multiple versions of Python on the same computer. For example, the computer might have a software-specific (e.g., ArcGIS) version of Python distribution. In such cases, it is recommended to not add Anaconda to the System PATH, as it might interfere with other software requiring Python. Population Synthesizer also might not be able to find the correct version of Python if run using Windows Command Prompt. Therefore, we recommend not add Anaconda to the system PATH nor register Anaconda Python as the system Python.

---

<sup>5</sup> Anaconda Environments: <https://docs.conda.io/projects/conda/en/latest/user-guide/concepts/environments.html>

<sup>6</sup> Anaconda Download Page: <https://www.anaconda.com/products/individual>

<sup>7</sup> Miniconda Documentation: <https://docs.conda.io/en/latest/miniconda.html>



**Obtain a U.S. Census application program interface (API) key. This is necessary to download Census data<sup>8</sup>. See the Census Data API User Guide<sup>9</sup> for instructions. The validation\_R/ directory**

This directory contains the R script to generate validation charts and summaries described in the *Validation procedures* section. Please note that the MWCOC Population Synthesizer setup automatically calls a Python-based validation script (validation.py) to generate validation charts and summaries. However, the R script produces better visuals and provides easy configuration options. Therefore, the validation R script and associated configuration files were added to the setup. The R script has been configured to work with the MWCOC Population Synthesizer setup and includes directions to update user inputs. The R script needs to be executed separately to generate validation charts and summaries for specified PopulationSim runs.

```
validation_R/
├── validation_PopulationSim.R      R validation scripts for PopulationSim.
├── columnMapPopSim_MWCOG_gq.csv   config file for group quarter run
└── columnMapPopSim_MWCOG_res.csv  config file for residential run
```

Configuration section details where to specify the Census API key in the Population Synthesizer software.

**Download MWCOC Population Synthesizer Setup.** Unzip the contents of the RSG PopulationSim Deliverable (MWCOG\_PopulationSynthesizer.zip) into a directory with read/write permissions. In the remainder of this document, we refer to the top-level of this directory as the “root” directory. All the code and necessary data are included in the zip file. Please note that the MWCOC Population Synthesizer will download and generate additional data. The final size of the directory will be **10-15 GB** depending on the number of runs performed (approximately 2 GB per additional run).

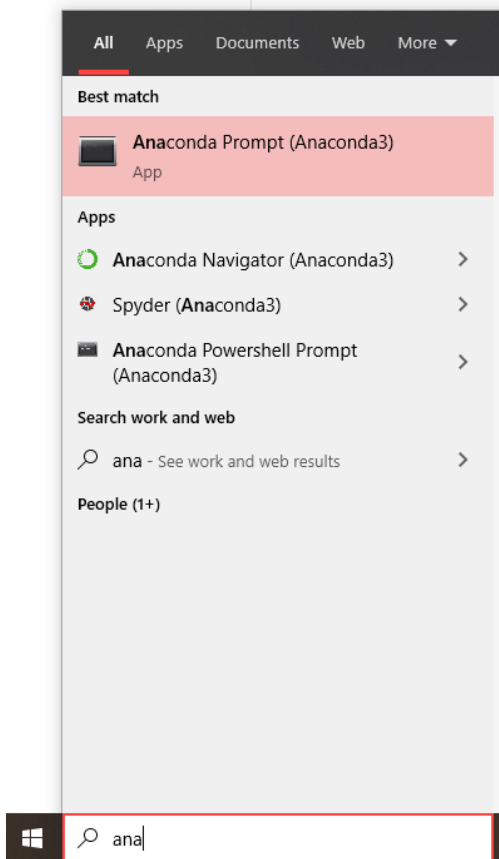
**Install the PopulationSim package.** Follow these steps to install the PopulationSim package:

1. *Start Anaconda Prompt:* On the Windows Operating System, click on the Start Menu (Windows icon on the lower left side of Desktop) and start typing “Anaconda” in the search bar. All matching entries will be shown. Select “Anaconda Prompt (Anaconda3)”.

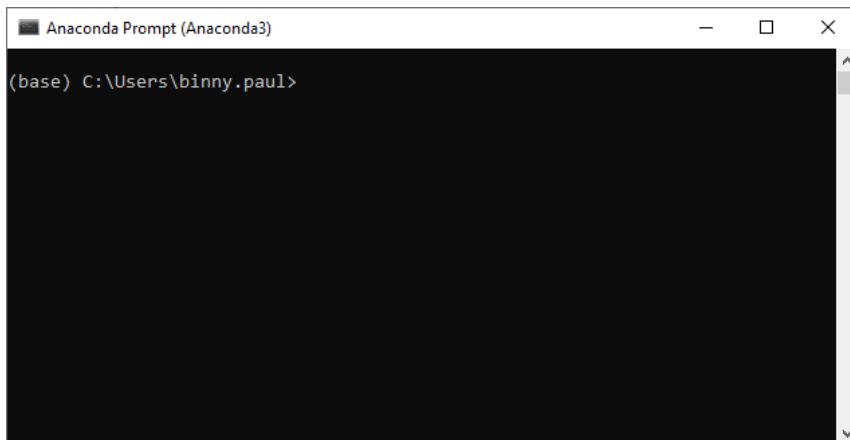
---

<sup>8</sup> Census data refers to any data obtained from the Census Bureau including the American Community Survey (ACS) datasets and Decennial Census datasets.

<sup>9</sup> Census Data API User Guide: <https://www.census.gov/data/developers/guidance/api-user-guide.html>



A new command window, named “Anaconda Prompt”, will open as shown below:



2. *Navigate to the root directory:* At the Anaconda prompt, navigate to the root MWCOG Population Synthesis directory with `cd /d C:\path\to\MWCOG Population Synthesis`. (Anaconda Prompt requires the use of backslashes in directory specifications).
3. *Create a conda environment and install PopulationSim:* Type `conda env create -f configs\popsim_env.yaml`. This will create a conda environment called 'popsim' and install the populationsim package and other dependencies as specified in the `popsim_env.yaml` file. Please note that, if the user has installed the PopulationSim software



before (e.g., the user may have already installed PopulationSim following the steps outlined in the PopulationSim documentation<sup>10</sup>), it is possible that the user has already created a conda environment named 'popsim'. In that case, the MWCOG Population Synthesizer may not be able to run in the existing 'popsim' environment, unless all the required Python packages listed as 'dependencies' in the `configs\popsim_env.yaml` file have been installed in this environment.<sup>11</sup> Thus, it is strongly recommended that the user first remove the existing 'popsim' environment by executing `conda remove --name popsim --all` and then recreate the 'popsim' environment by executing the `conda env create -f configs\popsim_env.yaml` command. If there is a need to keep the existing 'popsim' environment, the user may consider either renaming the existing environment or creating an environment with a different name for the MWCOG Population Synthesizer (by changing the environment name in the `popsim_env.yaml` file).

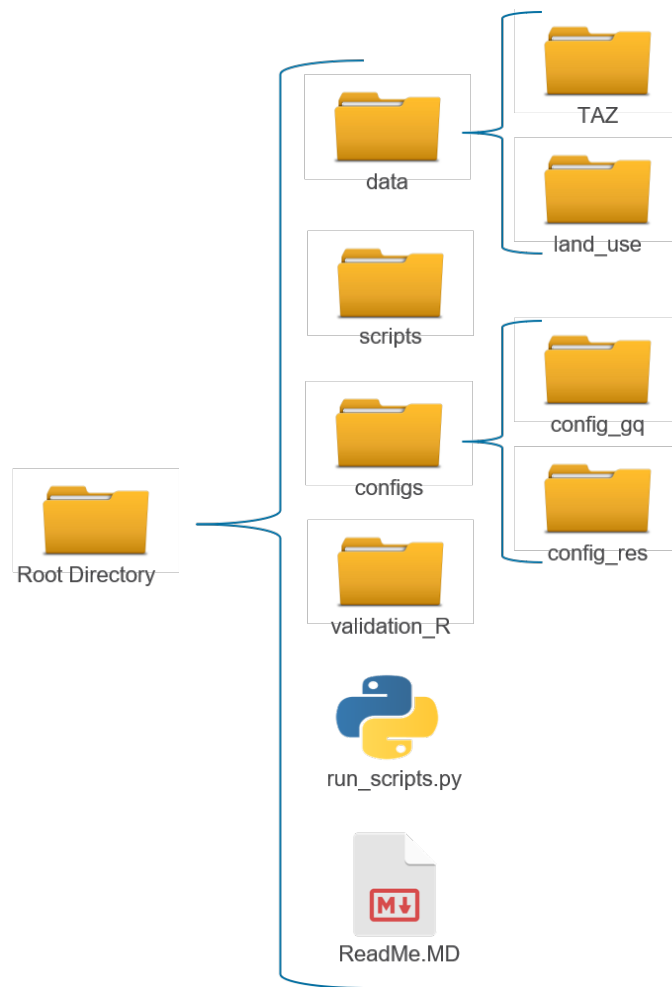
## Directory Setup

Figure 1 shows the directory structure of the MWCOG Population Synthesizer setup. A brief description of the components of the MWCOG Population Synthesizer directory setup follows.

---

<sup>10</sup> [https://activitysim.github.io/populationsim/getting\\_started.html](https://activitysim.github.io/populationsim/getting_started.html)

<sup>11</sup> During some of their tests, MWCOG staff failed to install the last the last two packages listed in the `popsim_env.yaml` file, namely, 'autocensus' and 'us', but were still able to run the software. This indicates that the two packages may be optional for running the software.



**FIGURE 1 MWCOG POPULATION SYNTHESIZER DIRECTORY STRUCTURE**

### ***Root directory***

Within the root directory, there are four folders and two files. The folders are `configs`, `data`, `scripts`, and `validation_R`. Each is described in more detail below. The files are `README.md` (a Markdown version of this document) and `run_scripts.py`, which is the master script that runs all the sub-scripts used in the MWCOG Population Synthesizer.

```

root/
├── configs/
├── data/
├── scripts/
├── validation_R/
├── README.md
└── run_scripts.py
  
```

### ***configs/ directory***

PopulationSim requires a large number of parameters and settings, which are described in text files, in CSV, YAML and text formats. All of the following files can be opened and viewed with

any standard text editor. Note that there are separate configuration directories for residential population synthesis (config\_res) and group quarters synthesis (config\_gq).

```
configs/
├── census_variables_needed.csv  file with details on census data download
├── config_gq/                  folder for group quarters synthesis config.
│   ├── controls.csv           file to specify PopulationSim controls
│   ├── logging.yaml           file for setting logging for PopulationSim
│   ├── settings.yaml:         file for PopulationSim config. parameters
│   └── validation.yaml:       file for the validation script
├── config_res/ :              folder for residential synthesis config.
│   ├── controls.csv           as above
│   ├── logging.yaml           as above
│   ├── settings.yaml          as above
│   └── validation.yaml        as above
├── census-api-key.txt:         text file with census api key
└── popsim_env.yaml:           file with instructions for conda environment
```

### **data/ directory**

Most of the data required by PopulationSim are automatically downloaded according to the requirements in configs/census\_variables\_needed.csv. All data files used in population synthesis are stored in this directory. Initially, the directory contains a shapefile delineating the transportation analysis zone (TAZ) polygons<sup>12</sup> and land use data<sup>13</sup> for forecast years.

```
data/
├── TAZ/                        Contains TAZ geography from MWCOG
│   ├── TPBTAZ3722_TPBMod.dbf
│   ├── TPBTAZ3722_TPBMod.prj
│   ├── TPBTAZ3722_TPBMod.sbn
│   ├── TPBTAZ3722_TPBMod.sbx
│   ├── TPBTAZ3722_TPBMod.shp
│   ├── TPBTAZ3722_TPBMod.shp.xml
│   ├── TPBTAZ3722_TPBMod.shx
│   └── TPBTAZ3722_TPBMod.vpr
├── land_use/                   Contains MWCOG socioeconomic forecasts
│   ├── LU_taz3722_rnd91a_2015_adj.dbf
│   ├── LU_taz3722_rnd91a_2018_adj.dbf
│   ├── LU_taz3722_rnd91a_2020_adj.dbf
│   ├── LU_taz3722_rnd91a_2025_adj.dbf
│   ├── LU_taz3722_rnd91a_2030_adj.dbf
│   ├── LU_taz3722_rnd91a_2035_adj.dbf
│   ├── LU_taz3722_rnd91a_2040_adj.dbf
│   └── LU_taz3722_rnd91a_2045_adj.dbf
```

<sup>12</sup> MWCOG TAZ Shape File (2018, 3722 TAZ):  
<https://app.box.com/s/ayw9d5kbwx5wbftak7f30dnjleepwqm1>

<sup>13</sup> MWCOG Land Use Data (2015 – 2045, 3722 TAZ):  
<https://app.box.com/folder/123653270372?s=6g78tis7d8zd8v5t31gg8m19hk80tegb>

### ***scripts/ directory***

The Python scripts in the `scripts/` directory are all designed to be called by the `root/run_scripts.py` Python script. This script passes the parameters needed for each of the sub-scripts in turn. Numbered scripts are called by `run_scripts.py` directly, in order. Unnumbered scripts are called by a numbered script as indicated below.

scripts/		
—	01_get_pums.py	downloads PUMS seed data and PUMA geography.
—	02_get_census_geography.py	downloads geog. of Census tracts and blocks.
—	03_get_census.py	downloads Census data for the study region.
—	04_create_crosswalk.py	creates crosswalks between levels
—	05_create_seed_sample.py	process the PUMS sample for PopulationSim
—	06_create_controls.py	processes and rescales Census data
—	07_run_populationsim.py	creates synthetic data
—	08_postprocessing.py	Runs post-processing needed on synthetic data
—	run_populationsim.py	file called by 07_run_populationsim.py
—	validation.py	file called by 07_run_populationsim.py

### ***validation\_R/ directory***

This directory contains the R script to generate validation charts and summaries described in the *Validation procedures* section. Please note that the MWCOC Population Synthesizer setup automatically calls a Python-based validation script (`validation.py`) to generate validation charts and summaries. However, the R script produces better visuals and provides easy configuration options. Therefore, the validation R script and associated configuration files were added to the setup. The R script has been configured to work with the MWCOC Population Synthesizer setup and includes directions to update user inputs. The R script needs to be executed separately to generate validation charts and summaries for specified PopulationSim runs.

validation_R/		
—	validation_PopulationSim.R	R validation scripts for PopulationSim.
—	columnMapPopSim_MWCOG_gq.csv	config file for group quarter run
—	columnMapPopSim_MWCOG_res.csv	config file for residential run

## **2.2 CONFIGURATION**

Configuring the MWCOC Population Synthesizer involves setting up the core PopulationSim software and specifying the user inputs in the `run_scripts.py` Python script and the PopulationSim settings files, as necessary. The user must copy the Census API key to the `configs/census-api-key.txt` file. The text file must not contain any other text besides the Census API key.

### **Configuring PopulationSim**

PopulationSim is configured using YAML and CSV files in `config_res/` and `config_gq/` for residential and group quarters population synthesis, respectively. The primary file is `settings.yaml` in which the user has the option to specify algorithm functionality, list

geographies, specify input tables and filenames, select variables to include in the final synthetic population, and list the steps to run. Fully configured `settings.yaml` files have been delivered as part of the final deliverable package. For detailed instructions on configuring a PopulationSim run, please refer to

[https://activitysim.github.io/populationsim/application\\_configuration.html#configuration](https://activitysim.github.io/populationsim/application_configuration.html#configuration).

The MWCOG Population Synthesizer works with three geographic levels: Region, Public Use Microdata Area (PUMA), and TAZs. The `settings.yaml` files were configured to work with these three geographies. The pre-processing Python scripts (numbers 01 through 06 as listed above in **scripts/ directory**) download and process the raw ACS PUMS and the MWCOG's TAZ-level socioeconomic data to produce inputs for PopulationSim. These inputs include the household and person seed sample, a TAZ-level controls file, and a geographic crosswalk file<sup>14</sup>. The input filenames and field names in the `settings.yaml` files were specified to match the outputs from the pre-processing scripts.

In addition to the `settings.yaml` file, the user needs to specify the controls for PopulationSim in the `controls.csv` file. The final versions of the `controls.csv` file for the MWCOG implementation of PopulationSim were also included in the final deliverable package. For detailed instructions on how to specify controls, please refer to

[https://activitysim.github.io/populationsim/application\\_configuration.html#specifying-controls](https://activitysim.github.io/populationsim/application_configuration.html#specifying-controls).

The importance factor values on each control were decided after trying out multiple configurations and choosing the ones that resulted in the best overall validation results. The maximum expansion factor which dictates the ratio of the final weight of a household record to its initial weight was set at 30.0 for this implementation. This value was also chosen after testing multiple configuration settings.

The final versions of `settings.yaml` and `controls.csv` were included in the final RSG PopulationSim deliverable. In general, these settings need not be changed except a) when there is a change in the input data that results in poor performance, b) when some controls are added or excluded, or c) when additional variables need to be included in the final synthetic population.

### **`run_scripts.py`**

In addition to the files described above, the configuration settings in `run_scripts.py` also need to be updated when running a new scenario or when setting up MWCOG Population Synthesizer on a new computer.

There are two main parameters in `run_scripts.py` that need to be set before starting a run:

- 1) The years for the simulation (`sim_years`)
- 2) Whether to overwrite PopulationSim output that has already been produced (`overwrite_output`)

---

<sup>14</sup> The geographic crosswalk file specifies the crosswalk between the following three geographies – Region, PUMA, and TAZ.

The following shows the relevant section of the file, lines 26-37.

```
# %% User parameters-----  
  
# Years for which to synthesize data  
sim_years = [2018, 2030, 2045]  
  
# Should existing PopSim output be overwritten?  
# Does not affect downloaded input data.  
overwrite_output = False  
  
# Path to your census API key  
census_key_path = 'configs/census-api-key.txt '
```

**NOTE:** the `overwrite_output` switch applies only to the synthetic population files and other output from PopulationSim. It does not apply to previously downloaded and processed input PUMS and ACS data, which will be used if present. If it is desirable to update or replace the downloaded and processed input data, please delete the outdated input data under the `data/` directory. The scripts will recreate them with newly downloaded and processed data.

Other parameters that can be modified if necessary (lines 40-61) include the locations of various input files as well as details of the PUMS and the ACS data.

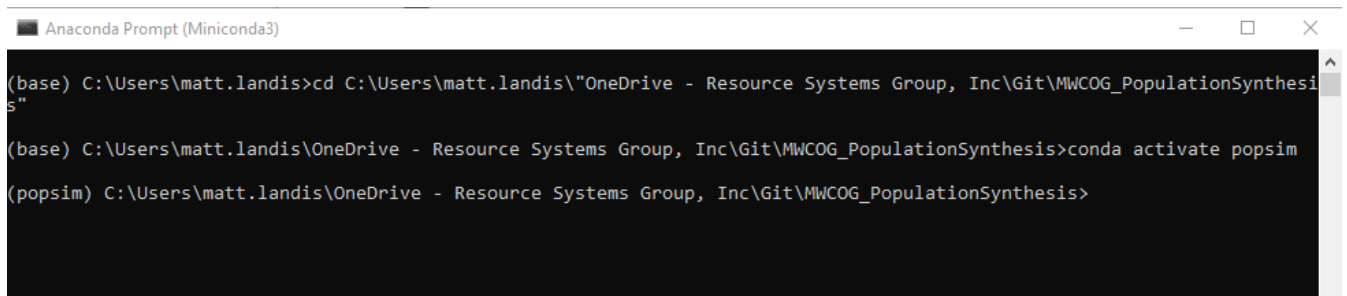
```
# %% Other parameters -----  
  
# TAZ path  
# Downloaded from Box:  
# https://app.box.com/s/ayw9d5kbwx5wbftak7f30dnjleepwqm1  
taz_path = 'data/TAZ/TPBTAZ3722_TPBMod.shp'  
  
# Census data needed  
path_vars_needed = 'configs/census_variables_needed.csv'  
  
# root directory is assumed to be this script's parent directory  
data_dir = 'data'  
  
# PUMS data download settings  
pums_dir = os.path.join(data_dir, 'PUMS') # Path to save the PUMS data  
pums_year = 2018  
pums_period = '5-Year' # '5-Year' or '1-Year'  
  
# ACS data download settings  
census_dir = os.path.join(data_dir, 'Census')  
census_year = pums_year  
census_period = pums_period
```



## 2.3 RUNNING THE SOFTWARE

Before launching the Population Synthesizer, the user must ensure that all configuration settings have been specified appropriately. To launch a new run, follow these steps:

1. **Open an Anaconda prompt** for typing commands from the Start menu as described earlier.
2. **Navigate to the root directory.** At the Anaconda prompt, navigate to the root MWCOG Population Synthesis directory with `cd /d C:\path\to\MWCOG Population Synthesis`. (Anaconda Prompt requires the use of backslashes in directory specifications).
3. **Activate the conda environment.** At the Anaconda prompt, type `conda activate popsim`. This will activate the conda environment created above. The prompt will indicate the active environment with “(popsim)” at the beginning of each command prompt line.



```
Anaconda Prompt (Miniconda3)
(base) C:\Users\matt.landis>cd C:\Users\matt.landis\OneDrive - Resource Systems Group, Inc\Git\MWCOG_PopulationSynthesis
(base) C:\Users\matt.landis\OneDrive - Resource Systems Group, Inc\Git\MWCOG_PopulationSynthesis>conda activate popsim
(popsim) C:\Users\matt.landis\OneDrive - Resource Systems Group, Inc\Git\MWCOG_PopulationSynthesis>
```

5. **Run MWCOG Population Synthesizer.** Once the configuration settings have been specified appropriately, the MWCOG Population Synthesizer run can be started by typing the following command in the Anaconda prompt: `python run_scripts.py` to call `run_scripts.py`

This script runs each of the following files in turn:

- `01_get_pums.py`. Downloads PUMS data for the specified years and subsets it to the PUMAs included in the modeling region. Output files saved for households (h) and persons (p) separately as `data\PUMS\puma_[pums year]_[pums period]_study_region_[h|p].csv`.
- `02_get_census_geography.py`. Downloads various geographic shapefiles from <https://www2.census.gov/geo/tiger/>. These files are needed for processing ACS and Census data at various spatial scales including state, tract, block group and block. Files are saved in `data\Census`.
- `03_get_census.py`. Downloads Census data and subsets it to the study region. Most variables needed for the controls are downloaded at the tract level, but some variables related to group quarters are needed at the block level (see below at **Marginal controls**).

- `04_create_crosswalk.py`. Creates crosswalk tables between levels of geography: Census block to TAZ, Census tract to TAZ and TAZ to PUMA. Files are saved to the `data\xwalk` directory.
- `05_create_seed_sample.py`. Processes PUMS data to create the seed sample data files used by PopulationSim as inputs. Processing steps include creation of the necessary input variables and separation of the data into residential and group quarters data files. See below at **Seed sample** for more detail. Writes out four files in the `data\` directory, one for each combination of household/person and residential/group quarters (`seed_hh_gq.csv`, `seed_per_gq.csv`, `seed_hh_res.csv`, `seed_per_res.csv`).
- `06_create_controls.py`. Creates the marginal controls. This script performs the following actions.
  - Reclassifies control variables as needed
  - Disaggregates tract-level data to TAZ
  - Scales data to MWCOC Round 9.1a Forecasts
  - Integerizes scaled counts
  - Saves output file as `data\control_taz_[year].csv`

See **Marginal controls** below for more detail.

- `07_run_populationsim.py`. This is the script that runs the PopulationSim software to produce the synthetic households. It runs the population synthesis separately for group quarters and residential populations by calling the script `run_populationsim.py` with the appropriate configuration directory and output directory arguments. After the population synthesis, it also runs `validation.py` to produce the validation graphics. The outputs are saved to `output\output_[gq|res]_[year]\synthetic_[households|persons].csv`.
- `08_postprocessing.py`. The final file in the sequence combines the group quarters and residential synthetic populations into a single file for either of households and persons, for each simulation year. The files are saved to `output\combined_synthetic_[hh|per]_[year].csv`.

Simultaneously running the MWCOC Population Synthesizer software from different user accounts on a modeling server may cause conflicts and fatal errors (crashes) as all the concurrent runs may operate on the same 'popsim' environment. It remains to be tested whether simultaneous runs in different 'popsim' environments are viable.

## 2.4 POPULATION SYNTHESIS OUTPUTS

As the scripts run, data will be downloaded, processed, and stored in additional folders within the root directory.

PopulationSim input data (seed sample data, marginal controls, and raw downloads from the US Census Bureau) are stored in the `data/` directory. The following shows the appearance of



the directory after a base year run (2018). The size of this directory following the downloading and processing of the data is approximately **2 GB**.

```
data/
├── Census/           Raw data from US Census Bureau
├── PUMS/             Raw data from ACS PUMS
├── TAZ/              TAZ shapefile from MWCOG
├── land_use/         Round 9.1a Cooperative Forecast DBF files
├── xwalk/            Geographic crosswalks needed for pre-processing
├── control_taz_2018.csv Marginal controls at TAZ level
├── geo_cross_walk.csv TAZ to PUMA crosswalk required by PopulationSim
├── seed_hh.csv        Residential seed sample household data
├── seed_hh_gq.csv     Group quarter seed sample household data
├── seed_per.csv       Residential seed sample person data
└── seed_per_gq.csv    Group quarter seed sample person data
```

PopulationSim output data appears in the `output/` directory. This includes the raw output from PopulationSim, zip archives of the output, and the combined residential/group quarters synthetic population data. The following shows the appearance of the directory after a base year run (2018).

```
output/
├── output_gq_2018/      group quarters output + validation
├── output_res_2018/     residential output + validation
├── combined_synthetic_hh_2018.csv synthetic data
├── combined_synthetic_per_2018.csv synthetic data
└── popsim_output_2018_20201001_1916.zip archive of PopulationSim output
```

After one run of PopulationSim, the size of this directory is approximately **2 GB**.

## 3.0 DATA PREPARATION AND APPLICATION

---

There are two main data processing steps in the MWCOC Population Synthesizer: 1) the preparation of inputs for the PopulationSim software (“pre-processing”) and 2) the processing of the PopulationSim outputs (“post-processing”), such as combining Group Quarters and Residential outputs and preparation of validation charts and summaries. The following sub-sections present the details of data preparation and application of the MWCOC Population Synthesizer.

### 3.1 INPUT DATA PREPARATION

The main data inputs to PopulationSim are:

- A disaggregate population sample (seed sample)
- Marginal control distributions (control variables)<sup>15</sup>

PopulationSim can work with both household-level and person-level controls. The controls can also be specified at multiple geographic levels. The geographic resolution of the seed sample is referred to as the “seed” geography. The marginal controls can be specified at the level of the seed geography or any number of sub-seed geographies. The marginal controls can also be specified at a “meta” geographic level which is above the seed geography. For the MWCOC implementation, the PUMS dataset was used as the seed sample, which is available at the Public Use Microdata Area (PUMA) level. The marginal controls are generated at the TAZ level from Census data (ACS and Decennial Census) and multi-year land use forecasts. As a result, the hierarchical geographic structure for the MWCOC implementation is defined as follows:

- Region
- PUMA
- TAZ

A geographic crosswalk file defines the hierarchical structure of the various geographies. The crosswalk between TAZ and PUMA is created by one of the pre-processing scripts, `04_create_crosswalk.py`, and formatted for PopulationSim in `06_create_controls.py`.

#### Seed sample

The seed sample data are generated by the script `05_create_seed_sample.py`. The main requirement for the seed sample is that it should be representative of the modeling region. The seed sample must contain the appropriate fields needed to specify various marginal controls. Also, it must contain variables that are needed for the ABM but that are not specified as controls. The ACS PUMS data satisfies these requirements and was used as the seed sample.

---

<sup>15</sup> In the context of List Balancing or Iterative Proportional Fitting, marginal controls refer to the row and column totals of the seed table.

The 2014-2018 5-year ACS PUMS data is the most recent vintage of the PUMS sample and the closest to the selected base year (2018). PUMS data for District of Columbia, Maryland, Virginia, and West Virginia were used. The pre-processing scripts filter the PUMS sample to include only those records belonging to the PUMAs overlapping the modeling region.

## Marginal controls

The control data are generated by the script `06_create_controls.py`. The residential marginal controls are produced from the 2018 ACS 5-year dataset downloaded at the tract level, based on specifications in `configs/census_variables_needed.csv` (see also `03_get_census.py`). The tract-level data are aggregated to TAZ level based on the area fraction of each Census tract that overlaps each TAZ. The group quarters data are not available at the sub-state level in the ACS 2018 sample so the 2010 Summary File 1 Census data were used at the block level. The block-level data were aggregated to TAZ by summing blocks within each TAZ. Finally, the Census/ACS counts were scaled to the Round 9.1a Cooperative Forecasts for each simulation year and each TAZ by calculating the ratio between the forecast data total and the Census/ACS data total within TAZ, and multiplying each Census/ACS count by this adjustment factor. Table 1 presents the list of control variables specified in the MWCOG Population Synthesizer.

**TABLE 1. DATA SOURCES FOR SEED SAMPLE AND MARGINAL CONTROLS**

Variable	Categories	PUMS Field	Control Source
Total HH		WGTP	Round 9.1a Forecasts
HH Size	1, 2, 3, 4+	NP	2018 ACS 5-year. Census Tract [Table S2501]
HH Income	0-\$25K, \$25K-\$50K, \$50K-\$100K, \$100k-\$150K, \$150k-\$200K, \$200K+	HINCP	2018 ACS 5-year. Census Tract [Table B19001]
Number of Workers	0, 1, 2, 3+	ESR	2018 ACS 5-year. Census Tract [Table B08202]
Presence of Children	0, 1	HUPAC	2018 ACS 5-year. Census Tract [Table S1101]
Person Age	0-4, 5-19, 20-34, 35-64, 65+	AGEP	2018 ACS 5-year. Census Tract [Table S0101]
Person Race	White, Hispanic, Black, Asian, Other	HISP, RAC1P	2018 ACS 5-year. Census Tract [Table DP05]
Total GQ units		TYPE	Round 9.1a Forecasts
GQ Type	University, Military, Other Non-Institutional (group homes, missions, shelters, etc.)	SCHG, MIL, TYPE	2010 Census SF1 [Table P042]

### HH Size Control Adjustments

The household size controls were further adjusted to resolve the inconsistencies between the TAZ-level population inferred from the Census/ACS household size distributions and the TAZ-

level household and population estimates from the Round 9.1a Cooperative Forecasts. A lookup table between average household size and household size distribution (1,2,3, 4+ categories) was computed from the Tract level Census/ACS data. The household size controls were recomputed for TAZs with problematic household size controls. To identify problematic TAZs, minimum and average implied populations were computed for each TAZ using the existing household size controls. The minimum implied population is computed by counting only 4 persons for the 4-plus household size category. For the average implied population, the average number of persons in a 4-plus person household is used. Problematic TAZs are the ones for which either of the following conditions is true.

1. Round 9.1a Cooperative Forecasts population < minimum implied population
2. Round 9.1a Cooperative Forecast population > 1.5 \* average implied population

Using the above rules, about 600 TAZs were tagged. The household size controls were recomputed for these TAZs using the household size distribution lookup table. This fixed the inconsistencies for more than 90% of the problematic TAZs. Table 2 shows some example TAZs that failed one of the above checks and how their household size controls were adjusted.

**TABLE 2: EXAMPLE HOUSEHOLD SIZE CONTROL ADJUSTMENTS**

	Round 9.1a CF			PopulationSim Controls						Data Checks		
TAZ	Total HH	HH persons	Avg HH Size	hh_1	hh_2	hh_3	hh_4p	Min implied persons	Avg implied persons	Check 1	Check 2	Version
159	1,211	4,098	3	502	420	87	202	2,411	2,546	✓	✗	Initial
159	1,211	4,098	3	171	335	244	461	3,417	3,726	✓	✓	Adjusted
411	660	1,396	2	190	155	140	175	1,620	1,737	✗	✓	Initial
411	660	1,396	2	283	227	80	70	1,257	1,304	✓	✓	Adjusted
1500	1,578	2,402	2	855	539	139	45	2,530	2,560	✗	✓	Initial
1500	1,578	2,402	2	1,018	459	69	32	2,271	2,292	✓	✓	Adjusted
3669	2,242	6,065	3	371	761	252	858	6,081	6,656	✗	✓	Initial
3669	2,242	6,065	3	586	745	383	528	5,337	5,691	✓	✓	Adjusted

## Scenario Applications

The marginal controls for PopulationSim need to be updated for modeling scenarios involving a change in demographics. To prepare the synthetic population for such scenarios, the user must create appropriate marginal control data. For example, when modeling an aging population scenario, the person-age controls must be adjusted to represent an aging population. For such a scenario, however, just updating the age controls will not be sufficient. The aging population will likely impact other distributions such as household income and auto ownership. The user must evaluate such effects and updated all marginal distributions to represent demographic distributions under an aging population scenario. The modified controls can be specified at a different geographic level compared to the base year. This depends on the availability of the control data at that geographic level and the accuracy of the data. Typically, the seed data remains the same.





## 3.2 CONTROLS AND SETTINGS

This section presents the final settings for the MWCOC Population Synthesizer. Different settings and configurations were tried during the initial testing of PopulationSim, such as varying the maximum expansion factor, combining controls, and altering importance factors on controls. The settings and configuration that resulted in the best overall validation performance were retained as the final version. The final version of the MWCOC PopulationSim uses a maximum expansion factor of 30. This is the default maximum expansion factor used by other agencies such as the Metropolitan Transportation Council (MTC), Oregon Department of Transportation, Portland Metro, Fresno Council of Governments, and Metropolitan Council (Minneapolis) in their PopulationSim implementation. Table 3 (residential) and Table 4. MWCOC PopulationSim Marginal Controls, Group Quarters (group quarters) present the final set of controls and importance factors.

**TABLE 3. MWCOC POPULATIONSIM MARGINAL CONTROLS, RESIDENTIAL**

Target	Geography	Seed Table	Importance
hh_total	TAZ	households	1,000,000,000
hh_size_1	TAZ	households	5000
hh_size_2	TAZ	households	5000
hh_size_3	TAZ	households	5000
hh_size_4_plus	TAZ	households	5000
hh_inc_0_25	TAZ	households	1000
hh_inc_25_50	TAZ	households	1000
hh_inc_50_100	TAZ	households	1000
hh_inc_100_150	TAZ	households	1000
hh_inc_150_200	TAZ	households	1000
hh_inc_200_plus	TAZ	households	1000
hh_worker_0	TAZ	households	5000
hh_worker_1	TAZ	households	5000
hh_worker_2	TAZ	households	5000
hh_worker_3plus	TAZ	households	5000
hh_w_kid	TAZ	households	1000
hh_wo_kid	TAZ	households	1000
per_age_0_4	TAZ	persons	1000
per_age_5_19	TAZ	persons	1000
per_age_20_34	TAZ	persons	1000
per_age_35_64	TAZ	persons	1000
per_age_65plus	TAZ	persons	1000

per_race_anyhispanic	TAZ	persons	5000
per_race_white	TAZ	persons	5000
per_race_black	TAZ	persons	5000
per_race_asian	TAZ	persons	5000
per_race_other	TAZ	persons	5000

**TABLE 4. MWCOG POPULATIONSIM MARGINAL CONTROLS, GROUP QUARTERS**

Target	Geography	Seed Table	Importance
gq_noninst	TAZ	households	1,000,000,000
gq_univ	TAZ	persons	1000
gq_mil	TAZ	persons	1000
gq_other	TAZ	persons	1000

## 4.0 VALIDATION

One of the most critical steps in population synthesis is validating the final synthetic population. Validation can give clues about inconsistencies among controls, data processing errors, or misspecification of any settings. This section describes the validation procedures and then presents the validation results from the final base-year (2018) run and future-year (2045) run.

### 4.1 VALIDATION PROCEDURES

PopulationSim reports the difference between the synthesized totals and the control totals for all the controls at each geographic level. The Python validation script generates advanced summary statistics and validation plots. These are described briefly below.

#### Validation summary statistics

The validation script reports the following information for each control: the total number of records (household/person) desired by the control, the total number of records synthesized, the difference between the synthesized total and the control total, and the percentage difference. Statistics that inform us of convergence at a more disaggregate level are also computed – please note that these statistics are computed for the geography at which the controls are specified *i.e.* TAZ or Region as the case might be. The following three statistics are computed as a part of this exercise:

1. the average percentage difference between the control totals and the synthesized totals,

$$\text{Average Percentage Difference (APD)} = \frac{\sum_{i=1}^N (\text{PercDiff}_i)}{N}$$

Where, *PercDiff<sub>i</sub>* is the percentage difference between the control totals and the synthesized totals at each geography and *N* is the total number of geographies.

2. the standard deviation (STDEV) of the percentage difference – this measure informs us of how much dispersion from the average exists,

$$\text{STDEV} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\text{PercDiff}_i - \text{APD})^2}$$

3. the percentage root mean square error (PRMSE) - an indicator of the proximity of synthesized and control totals.

$$\text{PRMSE} = \left( \frac{\sqrt{\frac{\sum_{i=1}^N (\text{Diff}_i)^2}{N}}}{\sum_{i=1}^N \text{Control}_i} \right) * N * 100$$

Where,  $Diff_i$  is the difference between the control totals and the synthesized totals at each geography and  $N$  is the total number of geographies.

4. The number of geographies for which the control is non-zero ( $N$ ) is also reported.

Traditionally, the performance of population synthesis is assessed at the regional level. These disaggregate statistics provide a tool to investigate data inconsistencies and misspecification errors. In the absence of any data inconsistencies and errors, the average percentage differences are expected to be close to zero across all controls. However, the observed data usually has some inconsistencies. Therefore, an average percentage difference in the range of -5% to +5% is considered acceptable. Please note that the control on the total number of households, in any case, should match perfectly. The sum of average percentage differences across all control categories should also be close to zero. For example, the average percentage difference on the 1-person household control may not be close to zero but the sum across all household size categories should be close to zero. A systematic pattern across all control categories may indicate issues in the control data or a specification error. The STDEV and PRMSE should ideally be in the -20% to +20% range. However, these statistics are very sensitive to the actual control values. For example, a control on a minority segment of the population may have a very low control value for each geography. A difference of 5 on a control value of 100 results in a percentage difference of 5% while the same difference on a control value of 20 results in a percentage difference of 20%. A higher percentage difference is generally acceptable for zones with a low control value, but the high percentage differences increase the STDEV and PRMSE. Therefore, a higher STDEV or PRMSE may be acceptable for controls with low control values that are hard to match exactly.

## Validation chart

The validation chart is a visualization of the disaggregate summary statistics – mean percentage difference, STDEV, and PRMSE of percentage differences. A form of dot and whisker plot is generated for each control where the dots are the mean percentage differences and horizontal bars are twice the STDEV or PRMSE centered around zero.

## Frequency distribution plots<sup>16</sup>

These are simply frequency distribution plots of differences between control and synthesized values across the geography at which the controls were specified.

## Expansion factor distribution<sup>17</sup>

While a synthetic population may match the controls well, it is important to know how uniform the household weights are, and how different they are from the initial weights. The closer the final weights are to the initial PUMS weight, the higher is the chance of matching the distribution

---

<sup>16</sup> Frequency distribution plot example:

<https://activitysim.github.io/populationsim/validation.html#frequency-distribution-plots>

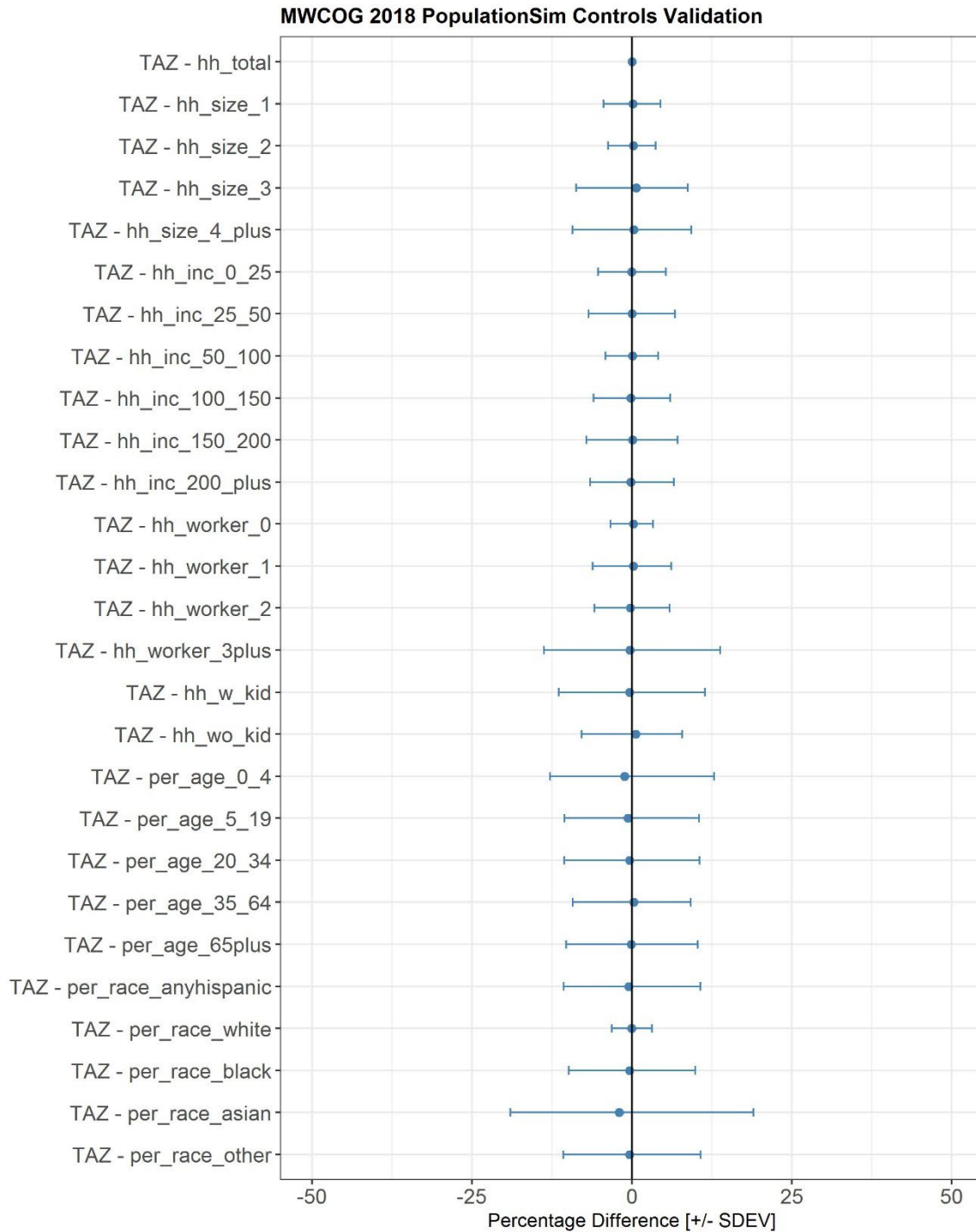
<sup>17</sup> Expansion factor distribution example:

<https://activitysim.github.io/populationsim/validation.html#expansion-factor-distributions>

of uncontrolled variables. An expansion factor is computed for each record in the PUMS data as total final weight/initial weight. A distribution plot of these expansion factors is created for each PUMA. A good synthetic population would have most of these expansion factors as close to one as possible.

## 4.2 BASE YEAR (2018) VALIDATION

Figure 2 presents the residential validation results of the MWCOG Population Synthesizer for the base year 2018. The validation results indicate that PopulationSim performs well overall, as can be observed from the close to zero mean percentage differences across all controls. The total number of households (top row of Figure 2) is matched perfectly across all TAZs as required. Overall, the standard deviation is also quite low for all household-level controls. The standard deviations on the person age controls are slightly higher than household-level controls but within reasonable limits. Typically, it is difficult to match the controls such as person race at TAZ-level and standard deviations are higher. The controls are matched closely when the control totals are consistent among themselves and across geographies. The TAZ-level person race controls are generated by applying the Tract-level person race distribution from Census/ACS data to the TAZ-level population. This is a reasonable approach for larger TAZs with a higher population, but for some small TAZs with a low population, it can create a set of person race controls that are very hard to match. For small TAZs, the control totals are small in value and even a small mismatch in control totals and synthesized totals results in a high percentage difference at the TAZ level. High percentage differences at TAZ-level lead to higher average percentage difference and standard deviation. Therefore, for controls such as person race, it is reasonable to exclude smaller TAZs from the validation statistics to get a better picture of that control's overall performance. Accordingly, TAZs with less than 25 households were excluded from the validation statistics for the person race control. The validation chart shows a reasonable performance for person race controls.

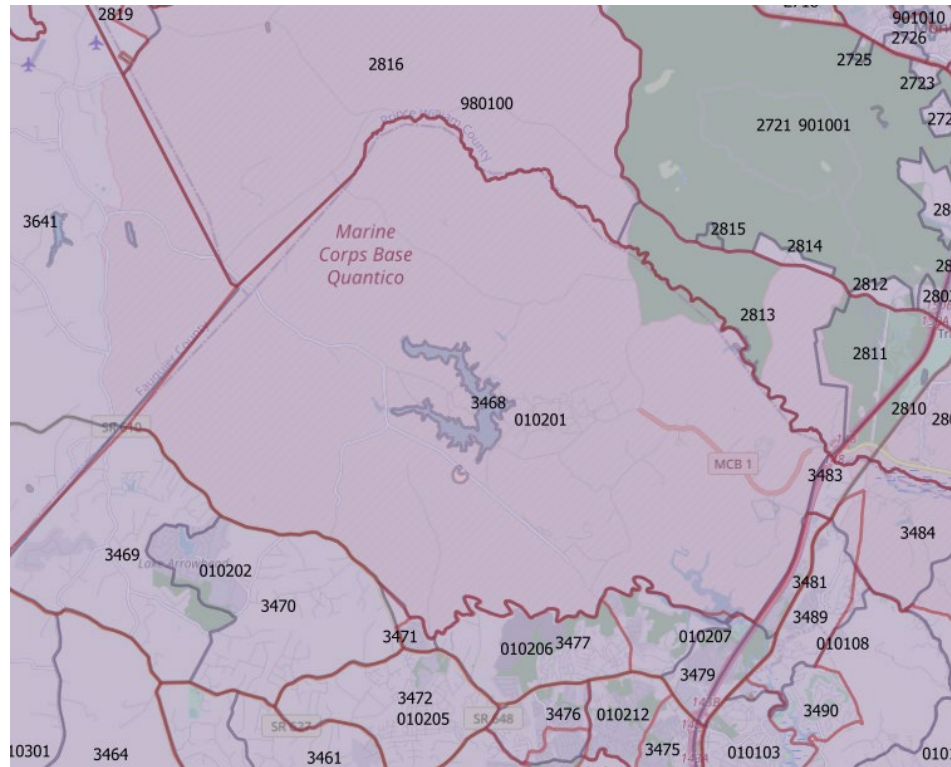


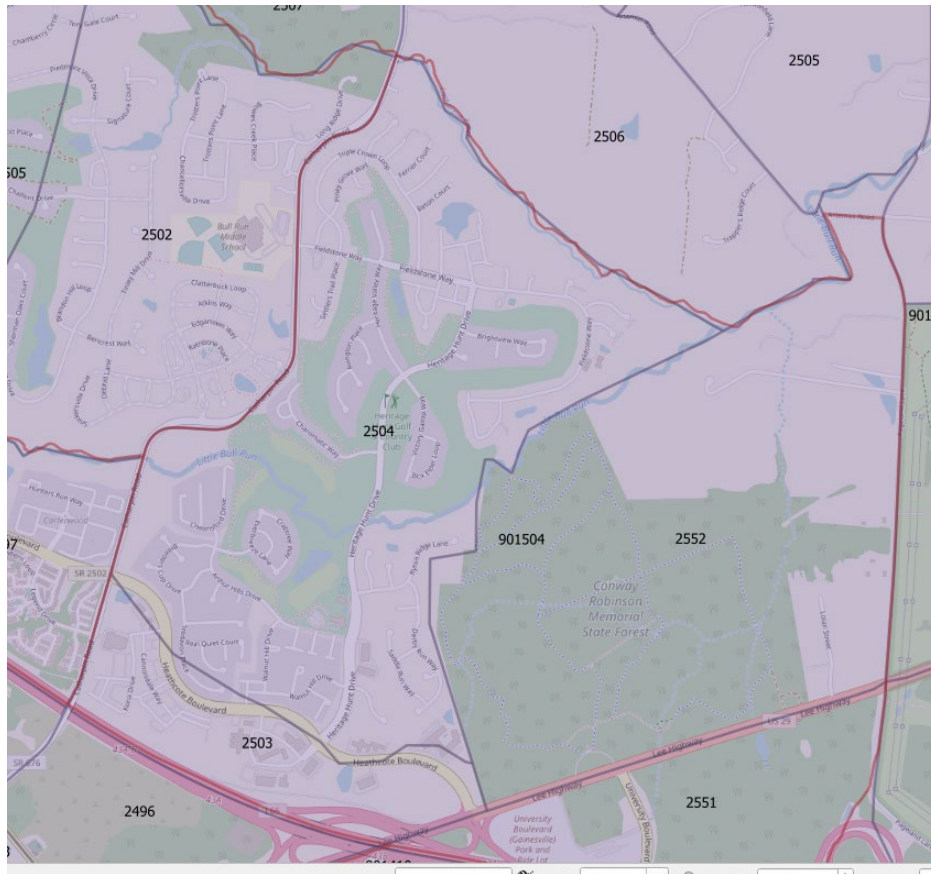
**FIGURE 2 MWCOG RESIDENTIAL POPULATIONSIM VALIDATION – 2018**

To investigate higher standard deviations on some controls, we spot-checked a few outliers at the TAZ-level. Generally, a higher variance is a result of inconsistencies among the controls or a data processing error. We identified two TAZs with inconsistencies between Census data and



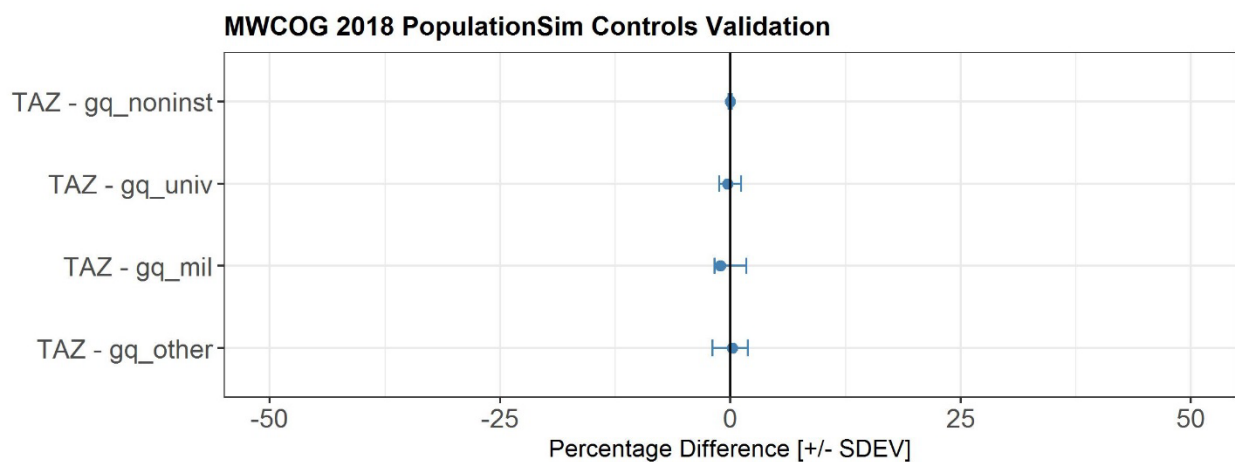
<sup>18</sup> TAZ boundaries are shown in grey and Census Tract boundaries are shown in red.

 26



**FIGURE 4 OUTLIER TAZ – 2504**

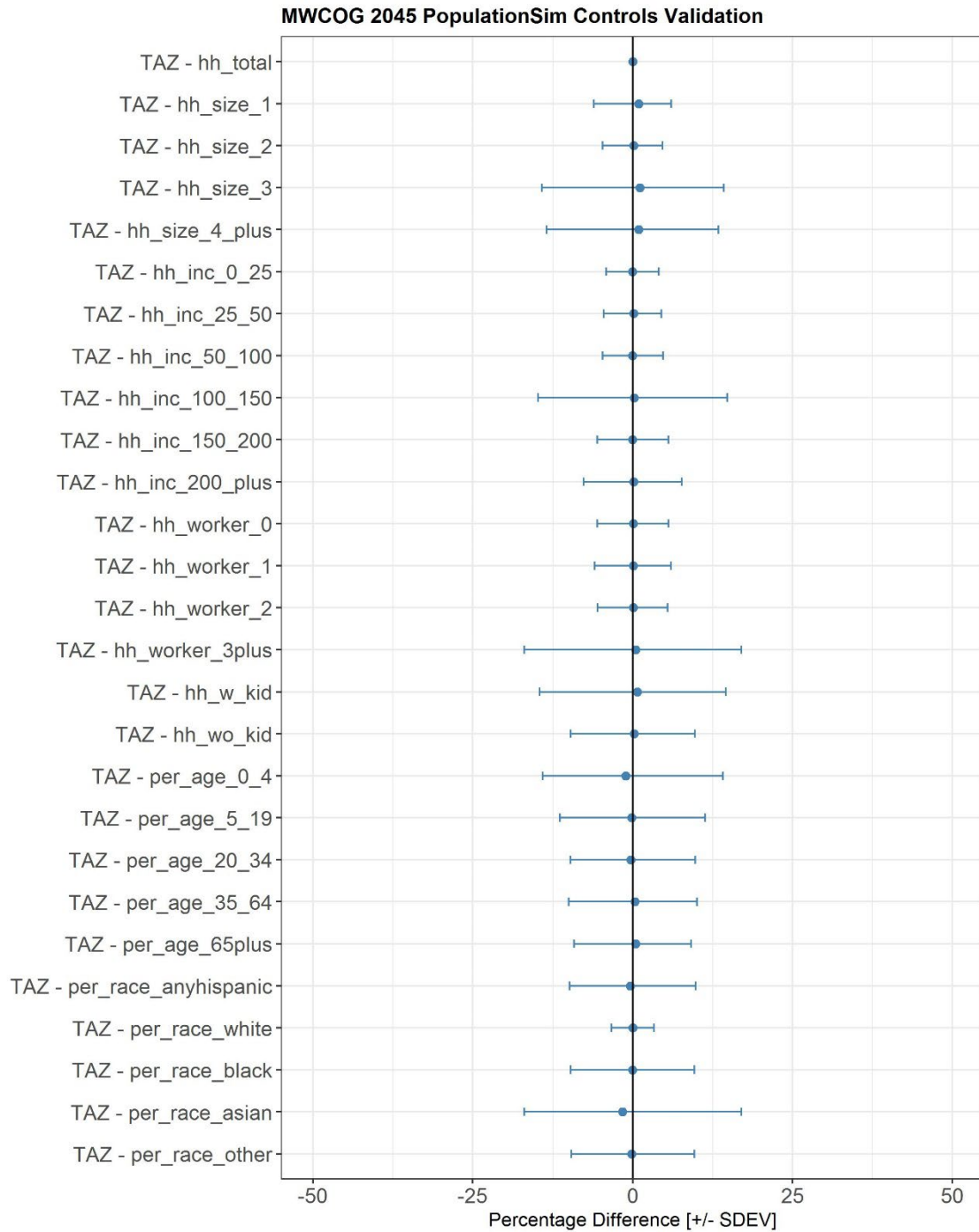
Figure 5 presents the group quarters validation result for the year 2018. PopulationSim matches the group quarters controls very well at the TAZ level as evidenced by the low percentage differences and variances.



**FIGURE 5 MWCOG GQ POPULATIONSIM VALIDATION - 2018**

## 4.3 FUTURE YEAR VALIDATION

Figure 6 presents the residential validation result for the year 2045. These results are very similar to the base year results with close to zero average percentage differences across all the controls. Again, the standard deviations are lower for household-level controls and slightly higher for person-level controls. Like base year, TAZs with less than 25 households were excluded from validation statistics for person race controls.



**FIGURE 6 MWCOG RESIDENTIAL POPULATIONSIM VALIDATION - 2045**