SynthPops

Institute for Disease Modeling

Feb 03, 2021

CONTENTS

SynthPops is used construct synthetic networks of people that satisfy statistical properties of real-world populations (such as the age distribution, household size, etc.). SynthPops can create generic populations with different network characteristics, as well as synthetic populations that interact in different layers of a multilayer contact network. These synthetic populations can then be used with agent-based models like COVID-19 Agent-based Simulator (Covasim) to simulate epidemics. SynthPops is available on GitHub. For more information on Covasim see Covasim on GitHub.

CHAPTER ONE

INSTALLATION

Follow the instructions below to install SynthPops.

1.1 Requirements

Python 3.6 64-bit. (Note: Python 2 is not supported.)

We also recommend, but do not require, using Python virtual environments. For more information, see documentation for venv or Anaconda.

1.2 Installation

Complete the following steps to install SynthPops:

1. Fork and clone the SynthPops GitHub repository.
2. Open a command prompt and navigate to the SynthPops directory.
3. Run the following script:

```
python setup.py develop
```

Note: while *synthpops* can also be installed via pypi, this method does not currently include the data files which are required to function, and thus is not recommended.

1.3 Quick start guide

The following code creates a synthetic population for Seattle, Washington:

```
import synthpops as sp

sp.validate()

datadir = sp.datadir # this should be where your demographics data folder resides

location = 'seattle_metro'
state_location = 'Washington'
country_location = 'usa'
sheet_name = 'United States of America'
level = 'county'
```

(continues on next page)

(continued from previous page)

```
npop = 10000 # how many people in your population
sp.generate_synthetic_population(npop,datadir,location=location,
                                 state_location=state_location,country_
↪location=country_location,
                                 sheet_name=sheet_name,level=level)
```

CHAPTER

TWO

SYNTHPOPS OVERVIEW

Fundamentally, the population network can be considered a multilayer network with the following qualities:

- Nodes are people, with attributes like age.
- Edges represent interactions between people, with attributes like the setting in which the interactions take place (for example, household, school, or work). The relationship between the interaction setting and properties governing disease transmission, such as frequency of contact and risk associated with each contact, is mapped separately by Covasim or other *agent-based model*. SynthPops reports whether the edge exists or not.

If you are using SynthPops with Covasim, note that the relevant value in Covasim is the parameter **beta**, which captures the probability of transmission via a given edge per *time step*. The value of this parameter captures both number of effective contacts for disease transmission and transmission probability per contact.

The generated network is a multilayer network in the sense that it is possible for people to be connected by multiple edges each in different layers of the network. The layers are referred to as *contact layers*. For example, the *workplace contact layer* is a representation of all of the pairwise connections between people at work, and the *household contact layer* represents the pairwise connections between household members. Typically these networks are clustered; in other words, everyone within a household interacts with each other, but not with other households. However, they may interact with members of other households via their school or workplace. Some level of community contacts outside of these networks can be configured using Covasim or other model being used with SynthPops.

SynthPops functions in two stages:

1. Generate people living in households, and then assign individuals to workplaces and schools. Save the output to a cache file on disk. Implemented in `generate_synthetic_population()`.
2. Load the cached file and produce a dictionary that can be used by Covasim. Implemented in `make_population()`. Covasim assigns community contacts at random on a daily basis to reflect the random and stochastic aspect of contacts in many public spaces, such as shopping centers, parks, and community centers.

CHAPTER

THREE

SYNTHPOPS ALGORITHM

This topic describes the algorithm used by SynthPops to generate the connections between people in each of the *contact layers* for a given location in the real world. The fundamental algorithm is the same for homes, schools, and workplaces, but with some variations for each.

The method draws upon the following previously published models to infer high-resolution age-specific contact patterns in different physical settings and locations:

- Mossong et al. 2008
- Fumanelli et al. 2012
- Prem et al. 2017
- Mistry et al. 2020

The general idea is to use age-specific contact matrices that describe age mixing patterns for a specific population. By default, SynthPops uses Prem et al.'s (2017) matrices, which project inferred age mixing patterns from the POLYMOD study (Mossong et al. 2008) in Europe to other countries. However, user-specified contact matrices can also be implemented for customizing age mixing patterns for the household, school, and workplace settings (see the social contact data on Zenodo for other empirical contact matrices from survey studies).

The matrices represent the average number of contacts between people for different age bins (the default matrices use 5-year age bins). For example, a household of two individuals is relatively unlikely to consist of a 25-year-old and a 15-year-old, so for the 25-29 year age bin in the household layer, there are a low number of expected contacts with the 15-19 year age bin (c.f., Fig. 2c in Prem et al.).

CHAPTER

FOUR

USING SYNTHPOPS

The overall SynthPops workflow is contained in `generate_synthetic_population()` and is described below. The population is generated through households, not a pool of people.

You can provide required data to SynthPops in a variety of formats including .csv, .txt, or Microsoft Excel (.xlsx).

1. Instantiate a collection of households with sizes drawn from census data. Populations cannot be created outside of the *household contact layer*.
2. For each household, sample the age of a "reference" person from data that maps household size to a reference person in those households. The reference person may be referred to as the head of the household, a parent in the household, or some other definition specific to the data being used. If no data mapping household size to ages of reference individuals are available, then the age of the reference person is sampled from the age distribution of adults for the location.
3. The age bin of the reference people identifies the row of the contact matrix for that location. The remaining household members are then selected by sampling an age for the distribution of contacts for the reference person's age (in other words, normalizing the values of the row and sampling for a column) and assigning someone with that age to the household.
4. As households are generated, individuals are given IDs.
5. After households are constructed, students are chosen according to enrollment data by age to generate the *school contact layer*.
6. Students are assigned to schools using a similar method as above, where we select the age of a reference person and then select their contacts in school from an age-specific contact matrix for the school setting and data on school sizes.
7. With all students assigned to schools, teachers are selected from the labor force according to employment data.
8. The rest of the labor force are assigned to workplaces in the *workplace contact layer* by selecting a reference person and their contacts using an age-specific contact matrix and data on workplace sizes.

4.1 Examples

Examples live in the *examples* folder. These can be run as follows:

- `python examples/make_generic_contacts.py`

  Creates a dictionary of individuals, each of whom are represented by another dictionary with their contacts contained in the `contacts` key. Contacts are selected at random with degree distribution following the Erdos-Renyi graph model.

- `python examples/generate_contact_network_with_microstructure.py`

Creates and saves to file households, schools, and workplaces of individuals with unique IDs, and a table mapping IDs to ages. Two versions of each contact layer (households, schools, or workplaces) are saved; one with the unique IDs of each individual in each group (a single household, school or workplace), and one with their ages (for easy viewing of the age mixing patterns created).

- `python examples/load_contacts_and_show_some_layers.py`

  Loads a multilayer contact network made of three layers and shows the age and ages of contacts for the first 20 people.

In the *tests* folder, you can view the following to see examples of additional functionality.

- `test_synthpop.py`

  Reads in demographic data and generates populations matching those demographics.

- `test_contacts.py`

  Generates random contact networks with individuals matching demographic data or reads in synthetic contact networks with three layers (households, schools, and workplaces).

- `test_contact_network_generation.py`

  Generates synthetic contact networks in households, schools, and workplaces with Seattle Metro data (and writes to file).

The other topics in this section walk through the specific data sources and details about the settings for each of the *contact layers*.

4.1.1 Household contact layer

The *household contact layer* represents the pairwise connections between household members. The population is generated within this contact layer, not as a separate pool of people.

As locations, households are special in the following ways:

- Unlike schools and workplaces, everyone must be assigned to a household.
- The size of the household is important (for example, a 2-person household looks very different in comparison to a 5- or 6-person household) and some households only have 1 person.
- The reference person/head of the household can be well-defined by data.

Data needed

The following data sets are required for households:

1. **Age bracket distribution** specifying the distribution of people in age bins for the location. For example:

```
age_bracket , percent
0_4         , 0.0594714358950416
5_9         , 0.06031137308234759
10_14       , 0.05338015778985113
15_19       , 0.054500690394160285
20_24       , 0.06161403846144956
25_29       , 0.08899312471888453
30_34       , 0.0883533486774803
35_39       , 0.07780767611060545
40_44       , 0.07099017823587304
45_49       , 0.06996903280562596
```

(continues on next page)

(continued from previous page)

```
50_54       , 0.06655242534751997
55_59       , 0.06350008343899961
60_64       , 0.05761405140489549
65_69       , 0.04487122889235999
70_74       , 0.030964420778483555
75_100       , 0.05110673396642193
```

2. **Age distribution of the reference person for each household size**

   The distribution is what matters, so it doesn't matter if absolute counts are available or not, each *row* is normalized. If this is not available, default to sampling the age of the reference individual from the age distribution for adults:

```
family_size , 18-20 , 20-24 , 25-29 , 30-34 , 35-39 , 40-44 , 45-49 , 50-54 , 55-
↪64 , 65-74 , 75-99
2           , 163   , 999   , 2316  , 2230  , 1880  , 1856  , 2390  , 3118  ,␣
↪9528  , 9345  , 5584
3           , 115   , 757   , 1545  , 1907  , 2066  , 1811  , 2028  , 2175  ,␣
↪3311  , 1587  , 588
4           , 135   , 442   , 1029  , 1951  , 2670  , 2547  , 2368  , 1695  ,␣
↪1763  , 520   , 221
5           , 61    , 172   , 394   , 905   , 1429  , 1232  , 969   , 683   , 623␣
↪   , 235   , 94
6           , 25    , 81    , 153   , 352   , 511   , 459   , 372   , 280   , 280␣
↪   , 113   , 49
7           , 24    , 33    , 63    , 144   , 279   , 242   , 219   , 115   , 157␣
↪   , 80    , 16
```

3. **Distribution of household sizes**:

```
household_size , percent
1              , 0.2781590909877753
2              , 0.3443313103056699
3              , 0.15759535523004006
4              , 0.13654311541644018
5              , 0.050887858718118274
6              , 0.019738368167953997
7              , 0.012744901174002305
```

4. **Household contact matrix** specifying the number/weight of contacts by age bin:

```
        0-10        , 10-20       , 20-30
0-10    0.659867911 , 0.503965302 , 0.214772978
10-20   0.314776879 , 0.895460015 , 0.412465791
20-30   0.132821425 , 0.405073038 , 1.433888594
```

   By default, SynthPops uses matrices from a study (Prem et al. 2017) that projected inferred age mixing patterns from the POLYMOD study (Mossong et al. 2008) in Europe to other countries. SynthPops can take in user-specified contact matrices if other age mixing patterns are available for the household, school, and workplace settings (see the social contact data on Zenodo for other empirical contact matrices from survey studies).

   In theory, the household contact matrix varies with household size, but in general data at that resolution is unavailable.

Workflow

Use these SynthPops functions to instantiate households as follows:

1. Call `generate_synthetic_population()` and provide the binned age bracket distribution data described above. This wrapper function calls the following functions:

   1. From the binned age distribution, `get_age_n()` creates samples of ages from the binned distribution, and then normalizes to create a single-year distribution. This distribution can therefore be gathered using whatever age bins are present in any given dataset.

   2. `generate_household_sizes_from_fixed_pop_size()` generates empty households with known size based on the distribution of household sizes.

   3. `generate_all_households()` contains the core implementation and constructs households with individuals of different ages living together. It takes in the remaining data sources above, and then does the following:

      - Calls `generate_living_alone()` to populate households with 1 person (either from data on those living alone or, if unavailable, from the adult age distribution).

      - Calls `generate_larger_households()` repeatedly with with different household sizes to populate those households, first sampling the age of a reference person and then their household contacts as outlined above.

4.1.2 School contact layer

The *school contact layer* represents all of the pairwise connections between people in schools, including both students and teachers. Schools are special in that:

- Enrollment rates by age determine the probability of individual being a student given their age.
- Staff members such as teachers are chosen from individuals determined to be in the adult labor force.
- The current methods in SynthPops treat student and worker status as mutually exclusive. Many young adults may be both students and workers, part time or full time in either status. The ability to select individuals to participate in both activities will be introduced in a later version of the model.

Data needed

The following data is required for schools:

1. **School size distribution**:

```
school_size , percent
0-50        , 0.2
51-100      , 0.1
101-300     , 0.3
```

2. **Enrollment by age** specifying the percentage of people of each age attending school. See `get_school_enrollment_rates()`, but note that this mainly implements parsing a Seattle-specific data file to produce the following data structure, which could equivalently be read directly from a file:

```
age , percent
0   , 0
1   , 0
2   , 0
3   , 0.529
```

(continues on next page)

(continued from previous page)

```
4   , 0.529
5   , 0.95
6   , 0.95
7   , 0.95
8   , 0.95
9   , 0.95
10  , 0.987
11  , 0.987
12  , 0.987
13  , 0.987
```

3. **School contact matrix** specifying the number/weight of contacts by age bin. This is similar to the household contact matrix. For example:

```
        0-10        , 10-20       , 20-30
0-10    0.659867911 , 0.503965302 , 0.214772978
10-20   0.314776879 , 0.895460015 , 0.412465791
20-30   0.132821425 , 0.405073038 , 1.433888594
```

4. **Employment rates by age**, which is used when determining who is in the labor force, and thus which adults are available to be chosen as teachers:

```
Age , Percent
16  , 0.496
17  , 0.496
18  , 0.496
19  , 0.496
20  , 0.838
21  , 0.838
22  , 0.838
```

5. **Student teacher ratio**, which is the average ratio for the location. Methods to use a distribution or vary the ratio for different types of schools may come in later developments of the model:

```
student_teacher_ratio=30
```

Typically, contact matrices describing age-specific mixing patterns in schools include the interactions between students and their teachers. These patterns describe multiple types of schools, from possibly preschools to universities.

Workflow

Use these SynthPops functions to implement the school contact layer as follows:

1. `get_uids_in_school()` uses the enrollment rates to determine which people attend school. This then provides the number of students needing to be assigned to schools.
2. `generate_school_sizes()` generates schools according to the school size distribution until there are enough places for every student to be assigned a school.
3. `send_students_to_school()` assigns specific students to specific schools.
   - This function is similar to households in that a reference student is selected, and then the contact matrix is used to fill the remaining spots in the school.
   - Some particulars in this function deal with ensuring a teacher/adult is less likely to be selected as a reference person, and restricting the age range of sampled people relative to the reference person so that a

primary school age reference person will result in the rest of the school being populated with other primary school age children

4. `get_uids_potential_workers()` selects teachers by first getting a pool of working age people that are not students.
5. `get_workers_by_age_to_assign()` further filters this population by employment rates resulting in a collection of people that need to be assigned workplaces.
6. In `assign_teachers_to_work()`, for each school, work out how many teachers are needed according to the number of students and the student-teacher ratio, and sample those teachers from the pool of adult workers. A minimum and maximum age for teachers can be provided to select teachers from a specified range of ages (this can be used to account for the additional years of education needed to become a teacher in many places).

4.1.3 Workplace contact layer

The *workplace contact layer* represents all of the pairwise connections between people in workplaces, except for teachers working in schools. After some workers are assigned to the *school contact layer* as teachers, all remaining workers are assigned to workplaces. Workplaces are special in that there is little/no age structure so workers of all ages may be present in every workplace.

Again, note that work and school are currently exclusive, because the people attending schools are removed from the list of eligible workers. This doesn't necessarily need to be the case though. In fact, we know that in any countries and cultures around the world, people take on multiple roles as both students and workers, either part-time or full-time in one or both activities.

Data required

The following data are required for generating the workplace contact layer:

1. **Workplace size distribution** - again, this gets normalized so can be specified as absolute counts or as normalized values:

```
work_size_bracket , size_count
1-4               , 2947
5-9               , 992
10-19             , 639
20-49             , 430
50-99             , 140
100-249           , 83
250-499           , 26
500-999           , 13
1000-1999         , 12
```

2. **Work contact matrix** specifying the number/weight of contacts by age bin. This is similar to the household contact matrix. For example:

```
        20-30       , 30-40       , 40-50
20-30   0.659867911 , 0.503965302 , 0.214772978
30-40   0.314776879 , 0.895460015 , 0.412465791
40-50   0.132821425 , 0.405073038 , 1.433888594
```

Workflow

1. `generate_workplace_sizes()` generates workplace sizes according to the workplace size distribution until the number of workers is reached.
2. `assign_rest_of_workers()` populates workplaces just like for households and schools: randomly selecting the age of a reference person, and then sampling the rest of the workplace using the contact matrix.

CHAPTER

FIVE

API REFERENCE

5.1 Submodules

5.1.1 synthpops.base module

The module contains frequently-used functions that do not neatly fit into other areas of the code base.

`synthpops.base.`**`norm_dic`**`(`*dic*`)`

Normalize the dictionary `dic`.

Parameters **`dic`** (*`dict`*) – A dictionary with numerical values.

Returns A normalized dictionary.

`synthpops.base.`**`norm_age_group`**`(`*age_dic*, *age_min*, *age_max*`)`

Create a normalized dictionary for the range `age_min` to `age_max`, inclusive.

Parameters

- **`age_dic`** (*`dict`*) – A dictionary with numerical values.
- **`age_min`** (*`int`*) – The minimum value of the range for the dictionary.
- **`age_max`** (*`int`*) – The maximum value of the range for the dictionary.

Returns A normalized dictionary for keys in the range `age_min` to `age_max`, inclusive.

`synthpops.base.`**`get_index_by_brackets_dic`**`(`*brackets*`)`

Create a dictionary mapping each item in the value arrays to the key. For example, if brackets are age brackets, then this function will map each age to the age bracket or bin that it belongs to, so that the resulting dictionary will give by_brackets_dic[age_index] = age bracket of age_index.

Parameters **`brackets`** (*`dict`*) – A dictionary mapping bracket or bin keys to the array of values that belong to each bracket.

Returns A dictionary mapping indices to the brackets or bins each index belongs to.

Return type dict

`synthpops.base.`**`get_age_by_brackets_dic`**`(`*age_brackets*`)`

Create a dictionary mapping age to the age bracket it falls in.

Parameters **`age_brackets`** (*`dict`*) – A dictionary mapping age bracket keys to age bracket range.

Returns A dictionary of age bracket by age.

Example

```
age_brackets = sp.get_census_age_brackets(sp.datadir,state_location='Washington',
↪country_location='usa')
age_by_brackets_dic = sp.get_age_by_brackets_dic(age_brackets)
```

`synthpops.base.`**`get_ids_by_age_dic`**(*age_by_id_dic*)

Get lists of IDs that map to each age.

Parameters **`age_by_id_dic`** (*`dict`*) – A dictionary with the age of each individual by their ID.

Returns A dictionary listing IDs for each age from a dictionary that maps ID to age.

`synthpops.base.`**`get_aggregate_ages`**(*ages*, *age_by_brackets_dic*)

Create a dictionary of the count of ages by age brackets.

Parameters

- **`ages`** (*`dict`*) – A dictionary of age count by single year.
- **`age_by_brackets_dic`** (*`dict`*) – A dictionary mapping age to the age bracket range it falls within.

Returns A dictionary of aggregated age count for specified age brackets.

Example

```
aggregate_age_count = sp.get_aggregate_ages(age_count, age_by_brackets_dic)
aggregate_matrix = symmetric_matrix.copy()
aggregate_matrix = sp.get_aggregate_matrix(aggregate_matrix, age_by_brackets_dic)
```

`synthpops.base.`**`get_aggregate_matrix`**(*matrix*, *age_by_brackets_dic*)

Aggregate a symmetric matrix to fewer age brackets. Do not use for homogeneous mixing matrix.

Parameters

- **`matrix`** (*`np.ndarray`*) – A symmetric age contact matrix.
- **`age_by_brackets_dic`** (*`dict`*) – A dictionary mapping age to the age bracket range it falls within.

Returns A symmetric contact matrix (`np.ndarray`) aggregated to age brackets.

Example

```
age_brackets = sp.get_census_age_brackets(sp.datadir,state_location='Washington',
↪country_location='usa')
age_by_brackets_dic = sp.get_age_by_brackets_dic(age_brackets)

aggregate_age_count = sp.get_aggregate_ages(age_count, age_by_brackets_dic)
aggregate_matrix = symmetric_matrix.copy()
aggregate_matrix = sp.get_aggregate_matrix(aggregate_matrix, age_by_brackets_dic)

asymmetric_matrix = sp.get_asymmetric_matrix(aggregate_matrix, aggregate_age_
↪count)
```

`synthpops.base.`**`get_asymmetric_matrix`**(*symmetric_matrix*, *aggregate_ages*)

Get the contact matrix for the average individual in each age bracket.

Parameters

- **symmetric_matrix** (*np.ndarray*) – A symmetric age contact matrix.
- **aggregate_ages** (*dict*) – A dictionary mapping single year ages to age brackets.

Returns A contact matrix (np.ndarray) whose elements M_ij describe the contact frequency for the average individual in age bracket i with all possible contacts in age bracket j.

Example

```
age_brackets = sp.get_census_age_brackets(sp.datadir,state_location='Washington',
↪country_location='usa')
age_by_brackets_dic = sp.get_age_by_brackets_dic(age_brackets)

aggregate_age_count = sp.get_aggregate_ages(age_count, age_by_brackets_dic)
aggregate_matrix = symmetric_matrix.copy()
aggregate_matrix = sp.get_aggregate_matrix(aggregate_matrix, age_by_brackets_dic)

asymmetric_matrix = sp.get_asymmetric_matrix(aggregate_matrix, aggregate_age_
↪count)
```

5.1.2 synthpops.config module

This module sets the location of the data folder and other global settings.

To change the level of log messages displayed, use e.g.

sp.logger.setLevel('CRITICAL')

synthpops.config.**checkmem**(*unit='mb'*, *fmt='0.2f'*, *start=0*, *to_string=True*)
For use with logger, check current memory usage

synthpops.config.**set_nbrackets**(*n*)
Set the number of census brackets – usually 16 or 20.

synthpops.config.**validate**(*verbose=True*)
Check that the data folder can be found.

synthpops.config.**set_location_defaults**(*country=None*)

synthpops.config.**get_config_data**()

synthpops.config.**version_info**()

5.1.3 synthpops.contact_networks module

This module generates the household, school, and workplace contact networks.

`synthpops.contact_networks.`**`make_contacts_from_microstructure_objects`**(*age_by_uid_dic*, *homes_by_uids*, *schools_by_uids=None*, *teachers_by_uids=None*, *non_teaching_staff_uids=None*, *workplaces_by_uids=None*, *facilities_by_uids=None*, *facilities_staff_uids=None*, *use_two_group_reduction=Fals* *average_LTCF_degree=20*, *with_school_types=False*, *school_mixing_type='random'*, *average_class_size=20*, *inter_grade_mixing=0.1*, *average_student_teacher_ratio=20*, *average_teacher_teacher_degree=3* *average_student_all_staff_ratio=15* *average_additional_staff_degree=2* *school_type_by_age=None*, *workplaces_by_industry_codes=Non* *verbose=False*, *max_contacts=None*)

From microstructure objects (dictionary mapping ID to age, lists of lists in different settings, etc.), create a dictionary of individuals. Each key is the ID of an individual which maps to a dictionary for that individual with attributes such as their age, household ID (hhid), school ID (scid), workplace ID (wpid), workplace industry code (wpindcode) if available, and contacts in different layers.

Parameters

- **`age_by_uid_dic`** (*`dict`*) – dictionary mapping id to age for all individuals in the population
- **`homes_by_uids`** (*`list`*) – A list of lists where each sublist is a household and the IDs of the household members.
- **`schools_by_uids`** (*`list`*) – A list of lists, where each sublist represents a school and the ids of the students and teachers within it
- **`teachers_by_uids`** (*`list`*) – A list of lists, where each sublist represents a school and the ids of the teachers within it
- **`workplaces_by_uids`** (*`list`*) – A list of lists, where each sublist represents a workplace and the ids of the workers within it
- **`facilities_by_uids`** (*`list`*) – A list of lists, where each sublist represents a skilled

nursing or long term care facility and the ids of the residents living within it

- **facilities_staff_uids** (*list*) – A list of lists, where each sublist represents a skilled nursing or long term care facility and the ids of the staff working within it
- **non_teaching_staff_uids** (*list*) – None or a list of lists, where each sublist represents a school and the ids of the non teaching staff within it
- **use_two_group_reduction** (*bool*) – If True, create long term care facilities with reduced contacts across both groups
- **average_LTCF_degree** (*int*) – default average degree in long term care facilities
- **with_school_types** (*bool*) – If True, creates explicit school types.
- **school_mixing_type** (*str or dict*) – The mixing type for schools, 'random', 'age_clustered', or 'age_and_class_clustered' if string, and a dictionary of these by school type otherwise. 'random' means random graphs for each school, 'age_clustered' means random graphs but with students mostly mixing within the age/grade (inter_grade_mixing controls mixing between grades), 'age_and_grade_clustered' means students cohorted into classes with their own teachers.
- **average_class_size** (*float*) – The average classroom size.
- **inter_grade_mixing** (*float*) – The average fraction of mixing between grades in the same school for clustered school mixing types.
- **average_student_teacher_ratio** (*float*) – The average number of students per teacher.
- **average_teacher_teacher_degree** (*float*) – The average number of contacts per teacher with other teachers.
- **average_student_all_staff_ratio** (*float*) – The average number of students per staff members at school (including both teachers and non teachers).
- **average_additional_staff_degree** (*float*) – The average number of contacts per additional non teaching staff in schools.
- **school_type_by_age** (*dict*) – A dictionary of probabilities for the school type likely for each age.
- **workplaces_by_industry_codes** (*np.ndarray or None*) – array with workplace industry code for each workplace
- **verbose** (*bool*) – If True, print debugging statements.
- **trimmed_size_dic** (*dict*) – If supplied, trim contacts on creation rather than post hoc.

Returns A popdict of people with attributes. Dictionary keys are the IDs of individuals in the population and the values are a dictionary for each individual with their attributes, such as age, household ID (hhid), school ID (scid), workplace ID (wpid), workplace industry code (wpindcode) if available, and the IDs of their contacts in different layers. Different layers available are households ('H'), schools ('S'), and workplaces ('W'), and long term care facilities ('LTCF'). Contacts in these layers are clustered and thus form a network composed of groups of people interacting with each other. For example, all household members are contacts of each other, and everyone in the same school is considered a contact of each other. If use_two_group_reduction is True, then contracts within 'LTCF' are reduced from fully connected.

Notes

Methods to trim large groups of contacts down to better approximate a sense of close contacts (such as classroom sizes or smaller work groups are available via sp.trim_contacts() or sp.create_reduced_contacts_with_group_types(): see these methods for more details).

synthpops.contact_networks.**create_reduced_contacts_with_group_types**(*popdict, group_1, group_2, setting, average_degree=20, p_matrix=None, force_cross_edges=True*)

Create contacts between members of group 1 and group 2, fixing the average degree, and the probability of an edge between any two groups controlled by p_matrix if provided. Forces inter group edge for each individual in group 1 with force_cross_groups equal to True. This means not everyone in group 2 will have a contact with group 1.

Parameters

- **group_1** (*list*) – list of ids for group 1
- **group_2** (*list*) – list of ids for group 2
- **average_degree** (*int*) – average degree across group 1 and 2
- **p_matrix** (*np.ndarray*) – probability matrix for edges between any two groups
- **force_cross_groups** (*bool*) – If True, force each individual to have at least one contact with a member from the other group

Returns Popdict with edges added for nodes in the two groups.

Notes

This method uses the Stochastic Block Model algorithm to generate contacts both between nodes in different groups

and for nodes within the same group. In the current version, fixing the average degree and p_matrix, the matrix of probabilities for edges between any two groups is not supported. Future versions may add support for this.

5.1.4 synthpops.data_distributions module

Read in data distributions.

synthpops.data_distributions.**get_relative_path**(*datadir*)

synthpops.data_distributions.**get_nbrackets**()

synthpops.data_distributions.**get_age_brackets_from_df**(*ab_file_path*)

Create a dict of age bracket ranges from ab_file_path.

Parameters

- **ab_file_path** (*string*) – file path to get the ends of different age
- **from** (*brackets*) –

Returns A dictionary with a np.ndarray of the age range that maps to each age bracket key.

Examples:

```
get_age_brackets_from_df(ab_file_path) returns a dictionary
age_brackets, where age_brackets[0] is the age range for the first age
bracket, age_brackets[1] is the age range for the second age bracket,
etc.
```

`synthpops.data_distributions.`**`get_age_bracket_distr_path`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *nbrackets=None*)

Get file_path for age distribution by age brackets.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in

Returns A file path to the age distribution by age bracket data.

`synthpops.data_distributions.`**`read_age_bracket_distr`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *nbrackets=None*, *file_path=None*, *use_default=False*)

A dict of the age distribution by age brackets. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in
- **`file_path`** (*`string`*) – file path to user specified age bracket distribution data
- **`use_default`** (*`bool`*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from the default_location, default_state, default_country.

Returns A dictionary of the age distribution by age bracket. Keys map to a range of ages in that age bracket.

synthpops.data_distributions.**get_smoothed_single_year_age_distr**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *nbrackets=None*, *file_path=None*, *use_default=None*, *window_length=7*)

A smoothed dict of the age distribution by single years. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population. Using moving windows to smooth out the age distribution.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified age bracket distribution data
- **use_default** (*bool*) – If True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from the default_location, default_state, default_country.
- **window_length** (*int*) – length of window, in units of years, over which to average or smooth out age distribution

Returns A dictionary of the age distribution by age bracket. Keys map to a range of ages in that age bracket.

synthpops.data_distributions.**get_household_size_distr_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for household size distribution.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to the household size distribution data.

synthpops.data_distributions.**get_household_size_distr**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

A dictionary of the distribution of household sizes. If you don't give the file_path, then supply the location, state_location, and country_location strings. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified household size distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of the household size distribution data. Keys map to the household size as an integer, values are the percent of households of that size.

synthpops.data_distributions.**get_head_age_brackets_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for head of household age brackets.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in

Returns A file path to the age brackets for head of household distribution data.

synthpops.data_distributions.**get_head_age_brackets**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get a dictionary of head age brackets either from the file_path directly, or using the other parameters to figure out what the file_path should be. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in
- **file_path** (*string*) – file path to user specified head age brackets data

- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from the default_location, default_state, default_country.

Returns A dictionary of the age brackets for head of household distribution data. Keys map to the age bracket as an integer, values are the percent of households which head of household in that age bracket.

synthpops.data_distributions.**get_household_head_age_by_size_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for head of household age by size counts or distribution. If the data doesn't exist at the state level, only give the country_location.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in

Returns A file path to the head of household age by household size count or distribution data.

synthpops.data_distributions.**get_household_head_age_by_size_df**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Return a pandas df of head of household age by the size of the household. If the file_path is given return from there first. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in
- **file_path** (*string*) – file path to user specified data for the age of the head of the household by household size
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from the default_location, default_state, default_country.

Returns A file path to the head of household age by household size count or distribution data.

synthpops.data_distributions.**get_head_age_by_size_distr**(*datadir, location=None, state_location=None, country_location=None, file_path=None, household_size_1_included=False, use_default=False*)

Create an array of head of household age bracket counts (column) given by size (row). If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from the default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in
- **file_path** (*string*) – file path to user specified age of the head of the household by household size distribution data
- **household_size_1_included** – if True, age distribution for who lives alone is included in the head of household age by household size dataframe, so it will be used. Else, assume a uniform distribution for this among all ages of adults.
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns An array where each row s represents the age distribution of the head of households for households of size s-1.

synthpops.data_distributions.**get_census_age_brackets_path**(*datadir, location=None, state_location=None, country_location=None, nbrackets=None*)

Get file_path for census age brackets: will depend on the state or country of the source data on the age distribution and age specific contact patterns.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state_location is in

Returns A file path to the age brackets to be used with census age data in combination with the contact matrix data.

synthpops.data_distributions.**get_census_age_brackets**(*datadir, location=None, state_location=None, country_location=None, file_path=None, use_default=False, nbrackets=None*)

Get census age brackets: depends on the country or source of the age distribution and the contact pattern data. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state
- **`country_location`** (*`string`*) – name of the country the state_location is in
- **`file_path`** (*`string`*) – file path to user specified census age brackets
- **`use_default`** (*`bool`*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of the range of ages that map to each age bracket.

`synthpops.data_distributions.`**`get_contact_matrix`**(*datadir*, *setting_code*, *sheet_name=None*, *file_path=None*, *delimiter=' '*, *header=None*)

Get setting specific age contact matrix given sheet name to use. If file_path is given, then delimiter and header should also be specified.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`setting_code`** (*`string`*) – name of the physial contact setting: H for households, S for schools, W for workplaces, C for community or other
- **`sheet_name`** (*`string`*) – name of the sheet in the excel file with contact patterns
- **`file_path`** (*`string`*) – file path to user specified age contact matrix
- **`delimiter`** (*`string`*) – delimter for the contact matrix file
- **`header`** (*`int`*) – row number for the header of the file

Returns Matrix of contact patterns where each row i is the average contact patterns for an individual in age bracket i and the columns represent the age brackets of their contacts. The matrix element i,j is then the contact rate, number, or frequency for the average individual in age bracket i with all of their contacts in age bracket j in that physical contact setting.

`synthpops.data_distributions.`**`get_contact_matrix_dic`**(*datadir*, *sheet_name=None*, *file_path_dic=None*, *delimiter=' '*, *header=None*, *use_default=False*)

Create a dict of setting specific age contact matrices. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`setting_code`** (*`string`*) – name of the physial contact setting: H for households, S for schools, W for workplaces, C for community or other

- **sheet_name** (*string*) – name of the sheet in the excel file with contact patterns
- **file_path_dic** (*string*) – dictionary to file paths of user specified age contact matrix, where keys are "H", "S", "W", and "C".
- **delimiter** (*string*) – delimter for the contact matrix file
- **header** (*int*) – row number for the header of the file

Returns A dictionary of the different contact matrices for each population, given by the sheet name. Keys map to the different possible physical contact settings for which data are available.

synthpops.data_distributions.**get_school_enrollment_rates_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get a file_path for enrollment rates by age.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to the school enrollment rates.

synthpops.data_distributions.**get_school_enrollment_rates**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get dictionary of enrollment rates by age. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified school enrollment by age data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of school enrollment rates by age.

synthpops.data_distributions.**get_school_size_brackets_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for school size brackets specific to the location under study.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to school size brackets.

synthpops.data_distributions.**get_school_size_brackets**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get school size brackets: depends on the source/location of the data. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified school size brackets data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of school size brackets.

synthpops.data_distributions.**get_school_size_distr_by_brackets_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for the distribution of school size by brackets.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to the distribution of school sizes by bracket.

synthpops.data_distributions.**get_school_size_distr_by_brackets**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get distribution of school sizes by size bracket or bin. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified school size distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of the distribution of school sizes by bracket.

synthpops.data_distributions.**get_default_school_type_age_ranges**()

Define and return default school types and the age range for each.

Returns A dictionary of default school types and the age range for each.

synthpops.data_distributions.**get_default_school_types_distr_by_age**()

Define and return default probabilities of school type for each age.

Returns A dictionary of default probabilities for the school type likely for each age.

synthpops.data_distributions.**get_default_school_types_by_age_single**()

Define and return default school type by age by assigning the school type with the highest probability.

Returns A dictionary of default school type by age.

synthpops.data_distributions.**get_default_school_size_distr_brackets**()

Define and return default school size distribution brackets.

Returns A dictionary of school size brackets.

synthpops.data_distributions.**get_default_school_size_distr_by_type**()

Define and return default school size distribution for each school type. The school size distributions are binned to size groups or brackets.

Returns A dictionary of school size distributions binned by size groups or brackets for each type of default school.

synthpops.data_distributions.**write_school_type_age_ranges**(*datadir*, *location*, *state_location*, *country_location*, *school_type_age_ranges*)

Write to file the age range for each school type.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **school_type_age_ranges** (*dict*) – a dictionary with the age range for each school type

Returns None.

synthpops.data_distributions.**get_school_type_age_ranges_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for the age range by school type.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to the age range for different school types.

synthpops.data_distributions.**get_school_type_age_ranges**(*datadir*, *location*, *state_location*, *country_location*, *file_path=None*, *use_default=None*)

Get a dictionary of the school types and the age range for each for the location specified.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of default school types and the age range for each.

synthpops.data_distributions.**get_school_size_distr_by_type_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for the school size distribution by school type.

Parameters

- **datadir** (*string*) – file path to the data directory

- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in

Returns A file path to the school size distribution data by different school types for the region specified.

Return type str

`synthpops.data_distributions.`**`get_school_size_distr_by_type`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=None*)

Get the school size distribution by school types. If use_default, then we'll try to look for location specific data first, and if that's not available we'll use default data from the set default locations (see sp.config.py). This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in, which should be the 'usa'
- **`file_path`** (*string*) – file path to user specified distribution data
- **`use_default`** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of school size distributions binned by size groups or brackets for each type of default school.

`synthpops.data_distributions.`**`get_employment_rates_path`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for employment rates by age.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in

Returns A file path to employment rates by age.

`synthpops.data_distributions.`**`get_employment_rates`**(*datadir*, *location*, *state_location*, *country_location*, *file_path=None*, *use_default=False*)

Get employment rates by age. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in, which should be the 'usa'
- **`file_path`** (*`string`*) – file path to user specified employment by age data
- **`use_default`** (*`bool`*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of employment rates by age.

`synthpops.data_distributions.`**`get_workplace_size_brackets_path`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for workplace size brackets.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in

Returns A file path to workplace size brackets.

`synthpops.data_distributions.`**`get_workplace_size_brackets`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get workplace size brackets. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in, which should be the 'usa'
- **`file_path`** (*`string`*) – file path to user specified workplace size brackets data

- **`use_default`** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of workplace size brackets.

`synthpops.data_distributions.`**`get_workplace_size_distr_by_brackets_path`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for the distribution of workplace size by brackets.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in

Returns A file path to the distribution of workplace sizes by bracket.

`synthpops.data_distributions.`**`get_workplace_size_distr_by_brackets`**(*datadir*, *location=None*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=False*)

Get the distribution of workplace size by brackets. If use_default, then we'll first try to look for location specific data and if that's not available we'll use default data from default_location, default_state, default_country. This may not be appropriate for the population under study so it's best to provide as much data as you can for the specific population.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in
- **`file_path`** (*string*) – file path to user specified workplace size distribution data
- **`use_default`** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from default_location, default_state, default_country.

Returns A dictionary of the distribution of workplace sizes by bracket.

`synthpops.data_distributions.`**`get_state_postal_code`**(*state_location*, *country_location*)

Get the state postal code.

Parameters

- **`state_location`** (*string*) – name of the state

- **country_location** (*string*) – name of the country the state is in

Returns A postal code for the state_location.

Return type str

synthpops.data_distributions.**get_usa_long_term_care_facility_path**(*datadir*, *state_location=None*, *country_location=None*, *part=None*)

Get file_path for state level data on Long Term Care Providers for the US from 2015-2016.

Parameters

- **datadir** (*string*) – file path to the data directory
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state is in
- **part** (*int*) – part 1 or 2 of the table

Returns A file path to data on Long Term Care Providers from 'Long-Term Care Providers and Services Users in the United States - State Estimates Supplement: National Study of Long-Term Care Providers, 2015-2016'. Part 1 or 2 are available.

Return type str

synthpops.data_distributions.**get_usa_long_term_care_facility_data**(*datadir*, *state_location=None*, *country_location=None*, *part=None*, *file_path=None*, *use_default=False*)

Get state level data table from National survey on Long Term Care Providers for the US from 2015-2016.

Parameters

- **datadir** (*string*) – file path to the data directory
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **part** (*int*) – part 1 or 2 of the table
- **file_path** (*string*) – file path to user specified LTCF distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A file path to data on the size distribution of residents per facility for Long Term Care Facilities.

Return type str

synthpops.data_distributions.**get_long_term_care_facility_residents_path**(*datadir*, *location=None*, *state_location=None*, *country_location=None*)

Get file_path for the size distribution of residents per facility for Long Term Care Facilities.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to data on the size distribution of residents per facility for Long Term Care Facilities.

synthpops.data_distributions.**get_long_term_care_facility_residents_distr**(*datadir, location=None, state_location=None, country_location=None, file_path=None, use_default=None*)

Get size distribution of residents per facility for Long Term Care Facilities.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified LTCF resident size distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of the distribution of residents per facility for Long Term Care Facilities.

synthpops.data_distributions.**get_long_term_care_facility_residents_distr_brackets_path**(*datad lo- ca- tion= state_ coun try_l*

Get file_path for the size bins for the distribution of residents per facility for Long Term Care Facilities.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to data on the size bins for the distribution of residents per facility for Long Term Care Facilities.

synthpops.data_distributions.**get_long_term_care_facility_residents_distr_brackets**(*datadir*, *lo- ca- tion=None*, *state_locatio coun- try_location= file_path=No use_default=*

Get size bins for the distribution of residents per facility for Long Term Care Facilities.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in, which should be the 'usa'
- **file_path** (*string*) – file path to user specified LTCF resident size brackets data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of size brackets or bins for residents per facility.

synthpops.data_distributions.**get_long_term_care_facility_resident_to_staff_ratios_path**(*datad lo- ca- tion= state_ coun try_l*

Get file_path for the distribution of resident to staff ratios per facility for Long Term Care Facilities.

Parameters

- **datadir** (*string*) – file path to the data directory
- **location** (*string*) – name of the location
- **state_location** (*string*) – name of the state the location is in
- **country_location** (*string*) – name of the country the location is in

Returns A file path to data on the distribution of resident to staff ratios per facility for Long Term Care Facilities.

synthpops.data_distributions.**get_long_term_care_facility_resident_to_staff_ratios_distr**(*data lo- ca- tion stat cou try_ file_ use_*

Get size distribution of resident to staff ratios per facility for Long Term Care Facilities.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in
- **`file_path`** (*string*) – file path to user specified resident to staff ratio distribution data
- **`use_default`** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of the distribution of residents per facility for Long Term Care Facilities.

`synthpops.data_distributions.`**`get_long_term_care_facility_resident_to_staff_ratios_brackets_`**

Get file_path for the size bins for the distribution of residents to staff ratios per facility for Long Term Care Facilities.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in

Returns A file path to data on the size bins for the distribution of resident to staff ratios per facility for Long Term Care Facilities.

Return type str

`synthpops.data_distributions.`**`get_long_term_care_facility_resident_to_staff_ratios_brackets`**

Get size bins for the distribution of resident to staff ratios per facility for Long Term Care Facilities.

Parameters

- **`datadir`** (*string*) – file path to the data directory
- **`location`** (*string*) – name of the location
- **`state_location`** (*string*) – name of the state the location is in
- **`country_location`** (*string*) – name of the country the location is in, which should be the 'usa'
- **`file_path`** (*string*) – file path to user specified resident to staff ratio brackets data

- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of size brackets or bins for resident to staff ratios per facility.

synthpops.data_distributions.**get_long_term_care_facility_use_rates_path**(*datadir*, *state_location=None*, *country_location=None*)

Get file_path for Long Term Care Facility use rates by age for a state.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in

Returns A file path to the data on the Long Term Care Facility usage rates by age.

Return type str

Note: Currently only available for the United States.

synthpops.data_distributions.**get_long_term_care_facility_use_rates**(*datadir*, *state_location=None*, *country_location=None*, *file_path=None*, *use_default=None*)

Get Long Term Care Facility use rates by age for a state.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **file_path** (*string*) – file path to user specified gender by age bracket distribution data
- **use_default** (*bool*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns A dictionary of the Long Term Care Facility usage rates by age.

Return type dict

Note: Currently only available for the United States.

5.1.5 synthpops.households module

Functions for generating households

`synthpops.households.`**`generate_household_sizes_from_fixed_pop_size`**(*N*, *hh_size_distr*)

Given a number of people and a household size distribution, generate the number of homes of each size needed to place everyone in a household.

Parameters

- **`N`** (*`int`*) – The number of people in the population.
- **`hh_size_distr`** (*`dict`*) – The distribution of household sizes.

Returns An array with the count of households of size s at index s-1.

`synthpops.households.`**`generate_household_head_age_by_size`**(*hha_by_size_counts*, *hha_brackets*, *hh_size*, *single_year_age_distr*)

Generate the age of the head of the household, also known as the reference person of the household, conditional on the size of the household.

Parameters

- **`hha_by_size_counts`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`hh_size`** (*`int`*) – The household size.
- **`single_year_age_distr`** (*`dict`*) – The age distribution.

Returns Age of the head of the household or reference person.

`synthpops.households.`**`generate_living_alone`**(*hh_sizes*, *hha_by_size_counts*, *hha_brackets*, *single_year_age_distr*)

Generate the ages of those living alone.

Parameters

- **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.
- **`hha_by_size_counts`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`single_year_age_distr`** (*`dict`*) – The age distribution.

Returns An array of households of size 1 where each household is a row and the value in the row is the age of the household member.

`synthpops.households.`**`assign_uids_by_homes`**(*homes*, *id_len=16*, *use_int=True*)

Assign IDs to everyone in order by their households.

Parameters

- **`homes`** (*`array`*) – The generated synthetic ages of household members.
- **`id_len`** (*`int`*) – The length of the UID.
- **`use_int`** (*`bool`*) – If True, use ints for the uids of individuals; otherwise use strings of length 'id_len'.

Returns A copy of the generated households with IDs in place of ages, and a dictionary mapping ID to age.

`synthpops.households.`**`generate_age_count`**`(`*n*, *age_distr*`)`

Generate a stochastic count of people for each age given the age distribution (age_distr) and number of people to generate (n).

Parameters

- **`n`** (*`int`*) – number of people to generate
- **`age_distr`** (*`list or np.ndarray`*) – single year age distribution

Returns A dictionary with the count of people to generate for each age given an age distribution and the number of people to generate.

Return type dict

`synthpops.households.`**`generate_living_alone_method_2`**`(`*hh_sizes*, *hha_by_size*, *hha_brackets*, *age_count*`)`

Generate the ages of those living alone.

Parameters

- **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.
- **`hha_by_size`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`age_distr`** (*`dict`*) – The age distribution.

Returns An array of households of size 1 where each household is a row and the value in the row is the age of the household member.

`synthpops.households.`**`generate_larger_household_sizes`**`(`*hh_sizes*`)`

Create a list of the households larger than 1 in random order so that as individuals are placed by age into homes running out of specific ages is not systemically an issue for any given household size unless certain sizes greatly outnumber households of other sizes.

Parameters **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.

Returns An array of household sizes to be generated and place people into households.

Return type Np.array

`synthpops.households.`**`generate_larger_households_head_ages`**`(`*larger_hh_size_array*, *hha_by_size*, *hha_brackets*, *ages_left_to_assign*`)`

Generate the ages of the heads of households for households larger than 2.

`synthpops.households.`**`generate_larger_households_method_2`**`(`*larger_hh_size_array*, *larger_hha_chosen*, *hha_brackets*, *cm_age_brackets*, *cm_age_by_brackets_dic*, *household_matrix*, *ages_left_to_assign*, *homes_dic*`)`

Assign people to households larger than one person (excluding special residences like long term care facilities or agricultural workers living in shared residential quarters.

Parameters

- **hh_sizes** (*array*) – The count of household size s at index s-1.
- **hha_by_size** (*matrix*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **hha_brackets** (*dict*) – The age brackets for the heads of household.
- **cm_age_brackets** (*dict*) – The age brackets for the contact matrix.
- **cm_age_by_brackets_dic** (*dict*) – A dictionary mapping age to the age bracket range it falls within.
- **household_matrix** (*dict*) – The age-specific contact matrix for the household ontact setting.
- **larger_homes_age_count** (*dict*) – Age count of people left to place in households larger than one person.

Returns A dictionary of households by age indexed by household size.

Return type dict

synthpops.households.**get_all_households**(*homes_dic*)

Get all households in a list, randomly assorted.

Parameters **homes_dic** (*dict*) – A dictionary of households by age indexed by household size

Returns A random ordering of households with the ages of the individuals.

Return type list

5.1.6 synthpops.ltcfs module

Modeling Seattle Metro Long Term Care Facilities

synthpops.ltcfs.**generate_ltcfs**(*n*, *with_facilities*, *datadir*, *country_location*, *state_location*, *location*, *use_default*, *smooth_ages*, *window_length*)

Generate residents living in long term care facilities and their ages.

Parameters

- **n** (*int*) – The number of people to create.
- **with_facilities** (*bool*) – If True, create long term care facilities, currently only available for locations in the US.
- **datadir** (*string*) – The file path to the data directory.
- **country_location** (*string*) – name of the country the location is in
- **state_location** (*string*) – name of the state the location is in
- **location** – name of the location
- **use_default** (*bool*) – If True, try to first use the other parameters to find data specific to the location under study; otherwise, return default data drawing from default_location, default_state, default_country.
- **smooth_ages** (*bool*) – If True, use smoothed out age distribution.
- **window_length** (*int*) – length of window over which to average or smooth out age distribution

Returns The number of people expected to live outside long term care facilities, age_brackets, age_by_brackets dictionary, age distribution adjusted for long term care facility residents already sampled, and facilities with people living in them.

`synthpops.ltcfs.`**`assign_facility_staff`**(*datadir*, *location*, *state_location*, *country_location*, *ltcf_staff_age_min*, *ltcf_staff_age_max*, *facilities*, *workers_by_age_to_assign_count*, *potential_worker_uids_by_age*, *potential_worker_uids*, *facilities_by_uids*, *age_by_uid_dic*, *use_default=False*)

Assign Long Term Care Facility staff to the generated facilities with residents.

Parameters

- **`datadir`** (*`string`*) – The file path to the data directory.
- **`location`** – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in
- **`ltcf_staff_age_min`** (*`int`*) – Long term care facility staff minimum age.
- **`ltcf_staff_age_max`** (*`int`*) – Long term care facility staff maximum age.
- **`facilities`** (*`list`*) – A list of lists where each sublist is a facility with the resident ages
- **`workers_by_age_to_assign_count`** (*`dict`*) – A dictionary mapping age to the count of employed individuals of that age.
- **`potential_worker_uids`** (*`dict`*) – dictionary of potential workers mapping their id to their age
- **`facilities`** – A list of lists where each sublist is a facility with the resident IDs
- **`age_by_uid_dic`** (*`dict`*) – dictionary mapping id to age for all individuals in the population
- **`use_default`** (*`bool`*) – If True, try to first use the other parameters to find data specific to the location under study; otherwise, return default data drawing from default_location, default_state, default_country.

Returns A list of lists with the facility staff IDs for each facility.

Return type list

`synthpops.ltcfs.`**`remove_ltcf_residents_from_potential_workers`**(*facilities_by_uids*, *potential_worker_uids*, *potential_worker_uids_by_age*, *workers_by_age_to_assign_count*, *age_by_uid_dic*)

Remove facilities residents from potential workers

`synthpops.ltcfs.`**`ltcf_resample_age`**(*exp_age_distr*, *a*)

Resampling younger ages to better match data

Parameters

- **`exp_age_distr`** (*`dict`*) – age distribution
- **`age`** (*`int`*) – age as an integer

Returns Resampled age as an integer.

Notes

This is not always necessary, but is mostly used to smooth out sharp edges in the age distribution when spsamp.resample_age() produces too many of one year and under produces the surrounding ages. For example, new borns (0 years old) may be over produced, and 1 year olds under produced, so this function can be customized to correct for that. It is currently customized to model well the age distribution for Seattle, Washington.

`synthpops.ltcfs.`**`generate_larger_households_method_1`**(*size*, *hh_sizes*, *hha_by_size_counts*, *hha_brackets*, *cm_age_brackets*, *cm_age_by_brackets_dic*, *contact_matrix_dic*, *single_year_age_distr*)

Generate ages of those living in households of greater than one individual. Reference individual is sampled conditional on the household size. All other household members have their ages sampled conditional on the reference person's age and the age mixing contact matrix in households for the population under study.

Parameters

- **`size`** (*`int`*) – The household size.
- **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.
- **`hha_by_size_counts`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`cm_age_brackets`** (*`dict`*) – The dictionary mapping age bracket keys to age bracket range matching the household contact matrix.
- **`cm_age_by_brackets_dic`** (*`dict`*) – The dictionary mapping age to the age bracket range it falls within matching the household contact matrix.
- **`contact_matrix_dic`** (*`dict`*) – A dictionary of the age-specific contact matrix for different physical contact settings.
- **`single_year_age_distr`** (*`dict`*) – The age distribution.

Returns An array of households for size `size` where each household is a row and the values in the row are the ages of the household members. The first age in the row is the age of the reference individual.

`synthpops.ltcfs.`**`generate_all_households_method_1`**(*N*, *hh_sizes*, *hha_by_size_counts*, *hha_brackets*, *cm_age_brackets*, *cm_age_by_brackets_dic*, *contact_matrix_dic*, *single_year_age_distr*)

Generate the ages of those living in households together. First create households of people living alone, then larger households. For households larger than 1, a reference individual's age is sampled conditional on the household size, while all other household members have their ages sampled conditional on the reference person's age and the age mixing contact matrix in households for the population under study.

Parameters

- **`N`** (*`int`*) – The number of people in the population.
- **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.

- **`hha_by_size_counts`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`cm_age_brackets`** (*`dict`*) – The dictionary mapping age bracket keys to age bracket range matching the household contact matrix.
- **`cm_age_by_brackets_dic`** (*`dict`*) – The dictionary mapping age to the age bracket range it falls within matching the household contact matrix.
- **`contact_matrix_dic`** (*`dict`*) – The dictionary of the age-specific contact matrix for different physical contact settings.
- **`single_year_age_distr`** (*`dict`*) – The age distribution.

Returns An array of all households where each household is a row and the values in the row are the ages of the household members. The first age in the row is the age of the reference individual. Households are randomly shuffled by size.

Note: This method is not guaranteed to model the population age distribution well automatically. The method called inside, generate_larger_households_method_1 uses the method ltcf_resample_age to fit Seattle, Washington populations with long term care facilities generated. For a method that matches the age distribution well for populations in general, please use generate_all_households_methods_2.

`synthpops.ltcfs.`**`generate_all_households_method_2`**(*n_nonltcf*, *hh_sizes*, *hha_by_size*, *hha_brackets*, *cm_age_brackets*, *cm_age_by_brackets_dic*, *contact_matrix_dic*, *ltcf_adjusted_age_distr*)

Generate the ages of those living in households together. First create households of people living alone, then larger households. For households larger than 1, a reference individual's age is sampled conditional on the household size, while all other household members have their ages sampled conditional on the reference person's age and the age mixing contact matrix in households for the population under study. Fix the count of ages in the population before placing individuals in households so that the age distribution of the generated population is fixed to closely match the age distribution from data on the population.

Parameters

- **`n_nonltcf`** (*`int`*) – The number of people in the population not living in long term care facilities.
- **`hh_sizes`** (*`array`*) – The count of household size s at index s-1.
- **`hha_by_size_counts`** (*`matrix`*) – A matrix in which each row contains the age distribution of the reference person for household size s at index s-1.
- **`hha_brackets`** (*`dict`*) – The age brackets for the heads of household.
- **`cm_age_brackets`** (*`dict`*) – The dictionary mapping age bracket keys to age bracket range matching the household contact matrix.
- **`cm_age_by_brackets_dic`** (*`dict`*) – The dictionary mapping age to the age bracket range it falls within matching the household contact matrix.
- **`contact_matrix_dic`** (*`dict`*) – The dictionary of the age-specific contact matrix for different physical contact settings.
- **`ltcf_adjusted_age_distr`** (*`dict`*) – The age distribution.

Returns An array of all households where each household is a row and the values in the row are the ages of the household members. The first age in the row is the age of the reference individual. Households are randomly shuffled by size.

5.1.7 synthpops.plotting module

This module plots the age-specific contact matrix in different settings.

`synthpops.plotting.`**`calculate_contact_matrix`**(*population*, *density_or_frequency='density'*, *setting_code='H'*)

Calculate the symmetric age-specific contact matrix from the connections for all people in the population. density_or_frequency sets the type of contact matrix calculated.

When density_or_frequency is set to 'frequency' each person is assumed to have a fixed amount of contact with others they are connected to in a setting so each person will split their contact amount equally among their connections. This means that if a person has links to 4 other individuals then 1/4 will be added to the matrix element matrix[age_i][age_j] for each link, where age_i is the age of the individual and age_j is the age of the contact. This follows the mass action principle such that increased density or number of people a person is in contact with leads to decreased per-link or connection contact rate.

When density_or_frequency is set to 'density' the amount of contact each person has with others scales with the number of people they are connected to. This means that a person with connections to 4 individuals has a higher total contact rate than a person with connection to 3 individuals. For this definition if a person has links to 4 other individuals then 1 will be added to the matrix element matrix[age_i][age_j] for each contact. This follows the 'pseudo'mass action principle such that the per-link or connection contact rate is constant.

Parameters

- **`population`** (*`dict`*) – A dictionary of a population with attributes.
- **`density_or_frequency`** (*`str`*) – option for the type of contact matrix calculated.
- **`setting_code`** (*`str`*) – name of the physial contact setting: H for households, S for schools, W for workplaces, C for community or other, and 'lTCF' for long term care facilities

Returns Symmetric age specific contact matrix.

`synthpops.plotting.`**`plot_contacts`**(*population*, *setting_code='H'*, *aggregate_flag=True*, *logcolors_flag=True*, *density_or_frequency='density'*, *cmap='cmr.freeze_r'*, *fontsize=16*, *rotation=50*, *title_prefix=None*, *fig=None*, *ax=None*, *do_show=True*)

Plot the age mixing matrix for a specific setting.

TODO: rename setting_code to layer

Parameters

- **`population`** (*`dict`*) – population to be plotted, if None, code will generate it
- **`setting_code`** (*`str`*) – name of the physial contact setting: H for households, S for schools, W for workplaces, C for community or other
- **`aggregate_flag`** (*`bool`*) – If True, plot the contact matrix for aggregate age brackets, else single year age contact matrix.
- **`logcolors_flag`** (*`bool`*) – If True, plot heatmap in logscale
- **`density_or_frequency`** (*`str`*) – If 'density', then each contact counts for 1/(group size -1) of a person's contact in a group, elif 'frequency' then count each contact. This means that more people in a group leads to higher rates of contact/exposure.

- **cmap** (*str or matplotlib colormap*) – colormap
- **fontsize** (*int*) – base font size
- **rotation** (*int*) – rotation for x axis labels
- **title_prefix** (*str*) – optional title prefix for the figure
- **fig** (*Figure*) – if supplied, use this figure instead of generating one
- **ax** (*Axes*) – if supplied, use these axes instead of generating one
- **do_show** (*bool*) – whether to show the plot

Returns A fig object.

5.1.8 synthpops.pop module

This module provides the layer for communicating with the agent-based model Covasim.

class synthpops.pop.**Pop**(*n=None*, *max_contacts=None*, *ltcf_pars=None*, *school_pars=None*, *with_industry_code=False*, *with_facilities=False*, *use_default=False*, *use_two_group_reduction=True*, *average_LTCF_degree=20*, *ltcf_staff_age_min=20*, *ltcf_staff_age_max=60*, *with_school_types=False*, *school_mixing_type='random'*, *average_class_size=20*, *inter_grade_mixing=0.1*, *average_student_teacher_ratio=20*, *average_teacher_teacher_degree=3*, *teacher_age_min=25*, *teacher_age_max=75*, *with_non_teaching_staff=False*, *average_student_all_staff_ratio=15*, *average_additional_staff_degree=20*, *staff_age_min=20*, *staff_age_max=75*, *rand_seed=None*, *country_location=None*, *state_location=None*, *location=None*, *sheet_name=None*, *household_method='infer_ages'*, *smooth_ages=False*, *window_length=7*, *do_make=True*)

Bases: `sciris.sc_utils.prettyobj`

generate(*verbose=False*)

Actually generate the network.

Parameters **verbose** (*bool*) – If True, print statements about the population and networks as they're being generated.

Returns A dictionary of the full population with ages, connections, and other attributes.

Return type network (dict)

to_dict()

Export to a dictionary – official way to get the popdict

Example:

```
popdict = pop.to_dict()
```

to_json(*filename*, *indent=2*, ***kwargs*)

Export to a JSON file

Example:

```
pop.to_json('my-pop.json')
```

save(*filename*, ***kwargs*)

Save population to an binary, gzipped object file

Example:

```
pop.save('my-pop.pop')
```

static load(*filename*, **args*, ***kwargs*)
Load from disk from a gzipped pickle.

Parameters

- **filename** (*str*) – the name or path of the file to load from
- **kwargs** – passed to sc.loadobj()

Example:

```
pop = sp.Pop.load('my-pop.pop')
```

plot_people(**args*, ***kwargs*)
Placeholder example of plotting the people in a population

plot_contacts(**args*, ***kwargs*)
Plot matrices of the contacts for a given layer or layers

synthpops.pop.**make_population**(**args*, ***kwargs*)
Interface to sp.Pop().to_dict(). Included for backwards compatibility.

synthpops.pop.**generate_synthetic_population**(**args*, ***kwargs*)
For backwards compatibility only.

5.1.9 synthpops.process_census module

This module provides functions that process data tables from the US Census Bureau into simple distribution tables that SynthPops functions can talk to.

Also includes functions to process data tables from the National survey on Long Term Care Providers in the US to convert those into rates by age for each US state using SynthPops functions.

synthpops.process_census.**process_us_census_age_counts**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)
Process American Community Survey data for a given year to get an age count for the location binned into 18 age brackets.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location** (*str*) – name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **year** (*int*) – the year for the American Community Survey
- **acs_period** (*int*) – the number of years for the American Community Survey

Returns A dictionary with the binned age count and a dictionary with the age bracket ranges.

`synthpops.process_census.`**`process_us_census_age_counts_by_gender`**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)

Process American Community Survey data for a given year to get an age count by gender for the location binned into 18 age brackets.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location`** (*`str`*) – name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in
- **`year`** (*`int`*) – the year for the American Community Survey
- **`acs_period`** (*`int`*) – the number of years for the American Community Survey

Returns A dictionary with the binned age count by gender and a dictionary with the age bracket ranges.

`synthpops.process_census.`**`process_us_census_population_size`**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)

Process American Community Survey data for a given year to get the population size for the location.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location`** (*`str`*) – name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in
- **`year`** (*`int`*) – the year for the American Community Survey
- **`acs_period`** (*`int`*) – the number of years for the American Community Survey

Returns The population size of the location for a given year estimated from the American Community Survey.

Return type int

`synthpops.process_census.`**`process_us_census_household_size_count`**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)

Process American Community Survey data for a given year to get a household size count for the location. The last bin represents households of size 7 or higher.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory

- **location** (*str*) – name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **year** (*int*) – the year for the American Community Survey
- **acs_period** (*int*) – the number of years for the American Community Survey

Returns A dictionary with the household size count.

synthpops.process_census.**process_us_census_employment_rates**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)

Process American Community Survey data for a given year to get employment rates by age as a fraction.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location** (*str*) – name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **year** (*int*) – the year for the American Community Survey
- **acs_period** (*int*) – the number of years for the American Community Survey

Returns A dictionary with the employment rates by age as a fraction.

synthpops.process_census.**process_us_census_enrollment_rates**(*datadir*, *location*, *state_location*, *country_location*, *year*, *acs_period*)

Process American Community Survey data for a given year to get enrollment rates by age as a fraction.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location** (*str*) – name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **year** (*int*) – the year for the American Community Survey
- **acs_period** (*int*) – the number of years for the American Community Survey

Returns A dictionary with the enrollment rates by age as a fraction.

synthpops.process_census.**process_us_census_workplace_sizes**(*datadir*, *location*, *state_location*, *country_location*, *year*)

Process American Community Survey data for a given year to get a count of workplace sizes as the number of employees per establishment.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location** (*str*) – name of the location

- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **year** (*int*) – the year for the American Community Survey

Returns A dictionary with the workplace or establishment size distribution as a count.

synthpops.process_census.**process_long_term_care_facility_rates_by_age**(*datadir*, *state_location*, *country_location*)

Process the National Long Term Care Providers state data tables from 2016 to get the estimated user rates by age.

Parameters

- **datadir** (*string*) – file path to the data directory
- **state_location** (*string*) – name of the state
- **country_location** (*string*) – name of the country the state is in

Returns A dictionary with the estimated rates of Long Term Care Facility usage by age for the state in 2016.

Return type dict

synthpops.process_census.**process_usa_ltcf_resident_to_staff_ratios**(*datadir*, *country_location*, *state_location*, *location_alias*, *location_list=['']*, *save=False*)

Process the Kaiser Health News (KHN) dashboard data on the ratios by facility to estimate the ratios for all facilities in the area. from 2016 to get the estimated user rates by age. Then write to file.

Parameters

- **datadir** (*string*) – file path to the data directory
- **country_location** (*string*) – name of the country
- **state_location** (*string*) – name of the state
- **location_alias** (*str*) – more commonly known name of the location
- **location_list** (*list*) – list of locations to include
- **save** (*bool*) – If True, save to file.

Returns A dictionary with the probability of resident to staff ratios and the bins.

Return type dict

synthpops.process_census.**write_age_bracket_distr_18**(*datadir*, *location_alias*, *state_location*, *country_location*, *age_bracket_count*, *age_brackets*)

Write age bracket distribution binned to 18 age brackets.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **age_bracket_count** (*dict*) – dictionary of the age count given by 18 brackets
- **age_brackets** (*dict*) – dictionary of the age range for each bracket

Returns None.

synthpops.process_census.**write_age_bracket_distr_16**(*datadir*, *location_alias*, *state_location*, *country_location*, *age_bracket_count*, *age_brackets*)

Write age bracket distribution binned to 16 age brackets.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **age_bracket_count** (*dict*) – dictionary of the age count given by 18 brackets
- **age_brackets** (*dict*) – dictionary of the age range for each bracket

Returns None.

synthpops.process_census.**write_gender_age_bracket_distr_18**(*datadir*, *location_alias*, *state_location*, *country_location*, *age_bracket_count_by_gender*, *age_brackets*)

Write age bracket by gender distribution data binned to 18 age brackets.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **age_bracket_distr** (*dict*) – dictionary of the age count by gender given by 18 brackets
- **age_brackets** (*dict*) – dictionary of the age range for each bracket

Returns None.

synthpops.process_census.**write_gender_age_bracket_distr_16**(*datadir*, *location_alias*, *state_location*, *country_location*, *age_bracket_count_by_gender*, *age_brackets*)

Write age bracket by gender distribution binned to 16 age brackets.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location_alias`** (*`str`*) – more commonly known name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in
- **`age_bracket_distr`** (*`dict`*) – dictionary of the age count by gender given by 18 brackets
- **`age_brackets`** (*`dict`*) – dictionary of the age range for each bracket

Returns None.

`synthpops.process_census.`**`read_household_size_count`**(*datadir*, *location_alias*, *state_location*, *country_location*)

Get household size count dictionary.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location_alias`** (*`str`*) – more commonly known name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in

Returns A dictionary of the household size count.

Return type dict

`synthpops.process_census.`**`write_household_size_count`**(*datadir*, *location_alias*, *state_location*, *country_location*, *household_size_count*)

Write household size count.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location_alias`** (*`str`*) – more commonly known name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in
- **`household_size_count`** (*`dict`*) – dictionary of the household size count.

Returns None.

`synthpops.process_census.`**`write_household_size_distr`**(*datadir*, *location_alias*, *state_location*, *country_location*, *household_size_count*)

Write household size distribution.

Parameters

- **`datadir`** (*`str`*) – file path to the data directory
- **`location_alias`** (*`str`*) – more commonly known name of the location
- **`state_location`** (*`str`*) – name of the state the location is in
- **`country_location`** (*`str`*) – name of the country the location is in

- **household_size_count** (*dict*) – dictionary of the household size count.

Returns None.

`synthpops.process_census.`**`write_employment_rates`**(*datadir*, *location_alias*, *state_location*, *country_location*, *employment_rates*)

Write employment rates by age as a fraction.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **employment_rates** (*dict*) – dictionary of the employment rates by age as a fraction.

Returns None.

`synthpops.process_census.`**`write_enrollment_rates`**(*datadir*, *location_alias*, *state_location*, *country_location*, *enrollment_rates*)

Write employment rates by age as a fraction.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **enrollment_rates** (*dict*) – dictionary of the enrollment rates by age as a fraction.

Returns None.

`synthpops.process_census.`**`write_long_term_care_facility_use_rates`**(*datadir*, *state_location*, *country_location*, *ltcf_rates_by_age*)

Write Long Term Care Facility usage rates by age as a fraction for a state in the United States.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **ltcf_rates_by_age** (*dict*) – dictionary of the long term care facility use rates by age as a fraction.

Returns None.

`synthpops.process_census.`**`write_workplace_size_counts`**(*datadir*, *location_alias*, *state_location*, *country_location*, *size_label_mappings*, *establishment_size_counts*)

Write workplace or establishment size count distribution.

Parameters

- **datadir** (*str*) – file path to the data directory
- **location_alias** (*str*) – more commonly known name of the location
- **state_location** (*str*) – name of the state the location is in
- **country_location** (*str*) – name of the country the location is in
- **size_label_mappings** (*dict*) – dictionary of the size labels mapping to the size bin
- **establishment_size_counts** (*dict*) – dictionary of the count of workplaces by size label

Returns None.

5.1.10 synthpops.sampling module

Sample distributions, either from real world data or from uniform distributions.

synthpops.sampling.**set_seed**(*seed=None*)
Reset the random seed – complicated because of Numba.

synthpops.sampling.**fast_choice**(*weights*)
Choose an option – quickly – from the provided weights. Weights do not need to be normalized.

Reimplementation of random.choices(), removing everything inessential.

Example

fast_choice([0.1,0.2,0.3,0.2,0.1]) # might return 2

synthpops.sampling.**sample_single_dict**(*distr_keys*, *distr_vals*)
Sample from a distribution.

Parameters **distr** (*dict or np.ndarray*) – distribution

Returns A single sampled value from a distribution.

synthpops.sampling.**sample_single_arr**(*distr*)
Sample from a distribution.

Parameters **distr** (*dict or np.ndarray*) – distribution

Returns A single sampled value from a distribution.

synthpops.sampling.**resample_age**(*age_dist_vals*, *age*)
Resample age from single year age distribution.

Parameters

- **single_year_age_distr** (*arr*) – age distribution, ordered by age
- **age** (*int*) – age as an integer

Returns Resampled age as an integer.

synthpops.sampling.**sample_from_range**(*distr*, *min_val*, *max_val*)
Sample from a distribution from min_val to max_val, inclusive.

Parameters

- **distr** (*dict*) – distribution with integer keys

- **`min_val`** (*int*) – minimum of the range to sample from
- **`max_val`** (*int*) – maximum of the range to sample from

Returns A sampled number from the range min_val to max_val in the distribution distr.

5.1.11 synthpops.schools module

This module generates school contacts by class and grade in flexible ways. Contacts can be clustered into classes and also mixed across the grade and across the school.

H. Guclu et. al (2016) shows that mixing across grades is low for public schools in elementary and middle schools. Mixing across grades is however higher in high schools.

Functions in this module are flexible to allow users to specify the inter-grade mixing (for 'age_clustered' school_mixing_type), and to choose whether contacts are clustered within a grade. Clustering contacts across different grades is not supported because there is no data to suggest that this happens commonly.

`synthpops.schools.`**`get_uids_in_school`**(*datadir, n, location, state_location, country_location, age_by_uid_dic=None, homes_by_uids=None, folder_name=None, use_default=False*)

Identify who in the population is attending school based on enrollment rates by age.

Parameters

- **`datadir`** (*string*) – The file path to the data directory.
- **`n`** (*int*) – The number of people in the population.
- **`location`** (*string*) – The name of the location.
- **`state_location`** (*string*) – The name of the state the location is in.
- **`country_location`** (*string*) – The name of the country the location is in.
- **`age_by_uid_dic`** (*dict*) – A dictionary mapping ID to age for all individuals in the population.
- **`homes_by_uids`** (*list*) – A list of lists where each sublist is a household and the IDs of the household members.
- **`folder_name`** (*string*) – The name of the folder the location is in, e.g. 'contact_networks'
- **`use_default`** (*bool*) – If True, try to first use the other parameters to find data specific to the location under study; otherwise, return default data drawing from default_location, default_state, default_country.

Returns A dictionary of students in schools mapping their ID to their age, a dictionary of students in school mapping age to the list of IDs with that age, and a dictionary mapping age to the number of students with that age.

`synthpops.schools.`**`send_students_to_school_with_school_types`**(*school_size_distr_by_type, school_size_brackets, uids_in_school, uids_in_school_by_age, ages_in_school_count, school_types_distr_by_age, school_type_age_ranges, verbose=False*)

A method to send students to school together. This method uses the dictionaries school_types_distr_by_age, school_type_age_ranges, and school_size_distr_by_type to first determine the type of school based on the age of a sampled reference student. Then the school type is used to determine the age range of the school. After that, the size of the school is then sampled conditionally on the school type and then the rest of the students are chosen from the lists of students available in the dictionary uids_in_school_by_age. This method is not perfect and requires a strict definition of school type by age. For now, it is not able to model mixed school types such as schools with Kindergarten through Grade 8 (K-8), or Kindergarten through Grade 12. These mixed types of schools may be common in some settings and this feature may be added later.

Parameters

- **school_size_distr_by_type** (*dict*) – A dictionary of school size distributions binned by size groups or brackets for each school type.
- **school_size_brackets** (*dict*) – A dictionary of school size brackets.
- **uids_in_school** (*dict*) – A dictionary of students in school mapping ID to age.
- **uids_in_school_by_age** (*dict*) – A dictionary of students in school mapping age to the list of IDs with that age.
- **ages_in_school_count** (*dict*) – A dictionary mapping age to the number of students with that age.
- **school_types_distr_by_age** (*dict*) – A dictionary of the school type for each age.
- **school_type_age_ranges** (*dict*) – A dictionary of the age range for each school type.
- **verbose** (*bool*) – If True, print statements about the generated schools as they're being generated.

Returns Two lists of lists and third flat list, the first where each sublist is the ages of students in the same school, and the second is the same list but with the IDs of each student in place of their age. The third is a list of the school types for each school, where each school has a single string to represent it's school type.

synthpops.schools.**add_contacts_from_edgelist**(*popdict*, *edgelist*, *setting*)

Add contacts to popdict from edges in an edgelist. Note that this simply adds to the contacts already in the layer and does not overwrite the contacts.

Parameters

- **popdict** (*dict*) – dict of people
- **edgelist** (*list*) – list of edges
- **setting** (*str*) – social setting layer

Returns Updated popdict.

synthpops.schools.**add_contacts_from_group**(*popdict*, *group*, *setting*)

Add contacts to popdict from fully connected group. Note that this simply adds to the contacts already in the layer and does not overwrite the contacts.

Parameters

- **popdict** (*dict*) – dict of people
- **group** (*list*) – list of people in group
- **setting** (*str*) – social setting layer

Returns Updated popdict.

synthpops.schools.**generate_random_contacts_for_additional_school_members**(*school_uids*, *ad- di- tional_school_member_ui av- er- age_additional_school_m*

Generate random contacts for additional school members. This might be people like non teaching staff such as principals, administrative staff, cleaning staff, or school nurses.

Parameters

- **school_uids** (*list*) – list of uids of individuals already in the school
- **additional_school_member_uids** (*list*) – list of uids of the additional school member who do not have contacts yet or for whom more contacts are needed
- **average_additional_school_members_degree** (*float*) – average degree for the additional school members

Returns List of edges for the additional school members in school.

synthpops.schools.**generate_random_classes_by_grade_in_school**(*syn_school_uids*, *syn_school_ages*, *age_by_uid_dic*, *grade_age_mapping*, *age_grade_mapping*, *aver- age_class_size=20*, *in- ter_grade_mixing=0.1*, *verbose=False*)

Generate edges for contacts mostly within the same age/grade. Edges are randomly distributed so that clustering is roughly average_class_size/size of the grade. Inter grade mixing is done by rewiring edges, specifically swapping endpoints of pairs of randomly sampled edges.

Parameters

- **syn_school_uids** (*list*) – list of uids of students in the school
- **syn_school_ages** (*list*) – list of the ages of the students in the school
- **age_by_uid_dic** (*dict*) – dict mapping uid to age
- **grade_age_mapping** (*dict*) – dict mapping grade to an age
- **age_grade_mapping** (*dict*) – dict mapping age to a grade
- **average_class_size** (*float*) – average class size
- **inter_grade_mixing** (*float*) – percent of edges that rewired to create edges across grades in schools when school_mixing_type is 'age_clustered'
- **verbose** (*bool*) – print statements throughout

Returns List of edges between students in school.

synthpops.schools.**generate_clustered_classes_by_grade_in_school**(*syn_school_uids*, *syn_school_ages*, *age_by_uid_dic*, *grade_age_mapping*, *age_grade_mapping*, *average_class_size=20*, *return_edges=False*, *verbose=False*)

Generate edges for contacts mostly within the same age/grade. Edges are randomly distributed so that clustering is roughly average_class_size/size of the grade.

Parameters

- **syn_school_uids** (*list*) – list of uids of students in the school
- **syn_school_ages** (*list*) – list of the ages of the students in the school
- **age_by_uid_dic** (*dict*) – dict mapping uid to age
- **grade_age_mapping** (*dict*) – dict mapping grade to an age
- **age_grade_mapping** (*dict*) – dict mapping age to a grade
- **average_class_size** (*float*) – average class size
- **return_edges** (*bool*) – If True, return edges, else return two groups of contacts - students and teachers for each class
- **verbose** (*bool*) – print statements throughout

Returns List of edges between students in school or groups of contacts.

synthpops.schools.**generate_edges_between_teachers**(*teachers*, *average_teacher_teacher_degree*)

Generate edges between teachers.

Parameters

- **teachers** (*list*) – a list of teachers
- **average_teacher_teacher_degree** (*int*) – average number of contacts with other teachers

Returns List of edges between teachers.

synthpops.schools.**generate_edges_for_teachers_in_random_classes**(*syn_school_uids*, *syn_school_ages*, *teachers*, *age_by_uid_dic*, *average_student_teacher_ratio=20*, *average_teacher_teacher_degree=4*, *verbose=False*)

Generate edges for teachers, including to both students and other teachers at the same school. Well mixed contacts within the same age/grade, some cross grade mixing. Teachers are clustered by grade mostly.

Parameters

- **syn_school_uids** (*list*) – list of uids of students in the school

- **syn_school_ages** (*list*) – list of the ages of the students in the school
- **teachers** (*list*) – list of teachers in the school
- **age_by_uid_dic** (*dict*) – dict mapping uid to age
- **grade_age_mapping** (*dict*) – dict mapping grade to an age
- **age_grade_mapping** (*dict*) – dict mapping age to a grade
- **average_student_teacher_ratio** (*float*) – average number of students per teacher
- **average_teacher_teacher_degree** (*float*) – average number of contacts with other teachers
- **verbose** (*bool*) – print statements throughout

Returns List of edges connected to teachers.

synthpops.schools.**generate_edges_for_teachers_in_clustered_classes**(*groups, teachers, average_student_teacher_ratio=20, average_teacher_teacher_degree=4, return_edges=False, verbose=False*)

Generate edges for teachers, including to both students and other teachers at the same school. Students and teachers are clustered into disjoint classes.

Parameters

- **groups** (*list*) – list of lists of students, clustered into groups mostly by grade
- **teachers** (*list*) – list of teachers in the school
- **average_student_teacher_ratio** (*float*) – average number of students per teacher
- **average_teacher_teacher_degree** (*float*) – average number of contacts with other teachers
- **return_edges** (*bool*) – If True, return edges, else return two groups of contacts - students and teachers for each class
- **verbose** (*bool*) – print statements throughout

Returns List of edges connected to teachers.

synthpops.schools.**generate_random_contacts_across_school**(*all_school_uids, average_class_size*)

Generate edges for contacts in a school where everyone mixes randomly. Assuming class and thus class size determines effective contacts.

Parameters

- **all_school_uids** (*list*) – list of uids of individuals in the school
- **average_class_size** (*int*) – average class size or number of contacts in school
- **verbose** (*bool*) – If True, print some edges

Returns List of edges between individuals in school.

synthpops.schools.**add_school_edges**(*popdict*, *syn_school_uids*, *syn_school_ages*, *teachers*, *non_teaching_staff*, *age_by_uid_dic*, *grade_age_mapping*, *age_grade_mapping*, *average_class_size=20*, *inter_grade_mixing=0.1*, *average_student_teacher_ratio=20*, *average_teacher_teacher_degree=3*, *average_additional_staff_degree=20*, *school_mixing_type='random'*, *verbose=False*)

Generate edges for teachers, including to both students and other teachers at the same school. When school_mixing_type is 'age_clustered' then inter_grade_mixing will rewire a fraction of the edges between students in the same age or grade to be edges with any other student in the school. When school_mixing_type is 'random' or 'age_and_class_clustered', inter_grade_mixing has no effect.

Parameters

- **popdict** (*dict*) – dictionary of people
- **syn_school_uids** (*list*) – list of uids of students in the school
- **syn_school_ages** (*list*) – list of the ages of the students in the school
- **teachers** (*list*) – list of teachers in the school
- **non_teaching_staff** (*list*) – list of non teaching staff in the school
- **age_by_uid_dic** (*dict*) – dict mapping uid to age
- **grade_age_mapping** (*dict*) – dict mapping grade to an age
- **age_grade_mapping** (*dict*) – dict mapping age to a grade
- **average_class_size** (*float*) – average class size
- **inter_grade_mixing** (*float*) – percent of edges that rewired to create edges across grades in schools when school_mixing_type is 'age_clustered'
- **average_student_teacher_ratio** (*float*) – average number of students per teacher
- **average_teacher_teacher_degree** (*float*) – average number of contacts with other teachers
- **average_additional_staff_degree** (*float*) – The average number of contacts per additional non teaching staff in schools.
- **school_mixing_type** (*str*) – 'random' for well mixed schools, 'age_clustered' for well mixed within the same grade and some intermixing with other grades, 'age_and_class_clustered' for disjoint classes in a school by age or grade
- **verbose** (*bool*) – print statements throughout

Returns Updated popdict.

synthpops.schools.**get_school_types_distr_by_age**(*school_type_age_ranges*)

Return probabilities of school type for each age. For now assuming no overlapping of grades between school types.

Returns A dictionary of default probabilities for the school type likely for each age.

synthpops.schools.**get_school_types_by_age_single**(*school_types_distr_by_age*)

Return school type by age by assigning the school type with the highest probability.

Returns A dictionary of default school type by age.

`synthpops.schools.`**`get_school_type_data`**(*datadir*, *location*, *state_location*, *country_location*, *use_default=False*)

Get location specific distributions on school type data if it's available for all the distributions of interest, otherwise return default data if use_default.

Parameters

- **`datadir`** (*`string`*) – file path to the data directory
- **`location`** (*`string`*) – name of the location
- **`state_location`** (*`string`*) – name of the state the location is in
- **`country_location`** (*`string`*) – name of the country the location is in
- **`use_default`** (*`bool`*) – if True, try to first use the other parameters to find data specific to the location under study, otherwise returns default data drawing from Seattle, Washington.

Returns 3 dictionaries necessary to generate schools by the type of school (i.e. elementary, middle, high school, etc.).

`synthpops.schools.`**`assign_teachers_to_schools`**(*syn_schools*, *syn_school_uids*, *employment_rates*, *workers_by_age_to_assign_count*, *potential_worker_uids*, *potential_worker_uids_by_age*, *potential_worker_ages_left_count*, *average_student_teacher_ratio=20*, *teacher_age_min=25*, *teacher_age_max=75*, *verbose=False*)

Assign teachers to each school according to the average student-teacher ratio.

Parameters

- **`syn_schools`** (*`list`*) – list of lists where each sublist is a school with the ages of the students within
- **`syn_school_uids`** (*`list`*) – list of lists where each sublist is a school with the ids of the students within
- **`employment_rates`** (*`dict`*) – employment rates by age
- **`workers_by_age_to_assign_count`** (*`dict`*) – dictionary of the count of workers left to assign by age
- **`potential_worker_uids`** (*`dict`*) – dictionary of potential workers mapping their id to their age
- **`potential_worker_uids_by_age`** (*`dict`*) – dictionary mapping age to the list of worker ids with that age
- **`potential_worker_ages_left_count`** (*`dict`*) – dictionary of the count of potential workers left that can be assigned by age
- **`average_student_teacher_ratio`** (*`float`*) – The average number of students per teacher.
- **`teacher_age_min`** (*`int`*) – The minimum age for teachers.
- **`teacher_age_max`** (*`int`*) – The maximum age for teachers.
- **`verbose`** (*`bool`*) – If True, print statements about the generated schools as teachers are being added to each school.

Returns List of lists of schools with the ages of individuals in each, lists of lists of schools with the ids of individuals in each, dictionary of potential workers mapping id to their age, dictionary mapping age to the list of potential workers of that age, dictionary with the count of workers left to assign for each age after teachers have been assigned.

`synthpops.schools.`**`assign_additional_staff_to_schools`**(*syn_school_uids*, *syn_teacher_uids*, *workers_by_age_to_assign_count*, *potential_worker_uids*, *potential_worker_uids_by_age*, *potential_worker_ages_left_count*, *average_student_teacher_ratio=20*, *average_student_all_staff_ratio=15*, *staff_age_min=20*, *staff_age_max=75*, *with_non_teaching_staff=False*, *verbose=True*)

Assign additional staff to each school according to the average student to all staff ratio.

Parameters

- **`syn_school_uids`** (*`list`*) – list of lists where each sublist is a school with the ids of the students within
- **`syn_teacher_uids`** (*`list`*) – list of lists where each sublist is a school with the ids of the teachers within
- **`workers_by_age_to_assign_count`** (*`dict`*) – dictionary of the count of workers left to assign by age
- **`potential_worker_uids`** (*`dict`*) – dictionary of potential workers mapping their id to their age
- **`potential_worker_uids_by_age`** (*`dict`*) – dictionary mapping age to the list of worker ids with that age
- **`potential_worker_ages_left_count`** (*`dict`*) – dictionary of the count of potential workers left that can be assigned by age
- **`average_student_teacher_ratio`** (*`float`*) – The average number of students per teacher.
- **`average_student_all_staff_ratio`** (*`float`*) – The average number of students per staff members at school (including both teachers and non teachers).
- **`staff_age_min`** (*`int`*) – The minimum age for non teaching staff.
- **`staff_age_max`** (*`int`*) – The maximum age for non teaching staff.
- **`with_non_teaching_staff`** (*`bool`*) – If True, includes non teaching staff.
- **`verbose`** (*`bool`*) – If True, print statements about the generated schools as teachers are being added to each school.

Returns List of lists of schools with the ids of non teaching staff for each school, dictionary of potential workers mapping id to their age, dictionary mapping age to the list of potential workers of that age, dictionary with the count of workers left to assign for each age after teachers have been assigned.

`synthpops.schools.`**`add_random_contacts_from_graph`**(*G*, *expected_average_degree*)

Add additional edges at random to achieve the expected or desired average degree.

Parameters

- **`G`** (*`networkx Graph`*) – networkx Graph object
- **`expected_average_degree`** (*`int`*) – expected or desired average degree

Returns Updated networkx Graph object with additional edges added at random.

`synthpops.schools.`**`generate_school_sizes`**(*school_size_distr_by_bracket*, *school_size_brackets*, *uids_in_school*)

Given a number of students in school, generate a list of school sizes to place everyone in a school.

Parameters

- **`school_size_distr_by_bracket`** (*`dict`*) – The distribution of binned school sizes.
- **`school_size_brackets`** (*`dict`*) – A dictionary of school size brackets.
- **`uids_in_school`** (*`dict`*) – A dictionary of students in school mapping ID to age.

Returns A list of school sizes whose sum is the length of `uids_in_school`.

`synthpops.schools.`**`send_students_to_school`**(*school_sizes*, *uids_in_school*, *uids_in_school_by_age*, *ages_in_school_count*, *age_brackets*, *age_by_brackets_dic*, *contact_matrix_dic*, *verbose=False*)

A method to send students to school together. Using the matrices to construct schools is not a perfect method so some things are more forced than the matrix method alone would create. This method models schools using matrices and so it does not create explicit school types.

Parameters

- **`school_sizes`** (*`list`*) – A list of school sizes.
- **`uids_in_school`** (*`dict`*) – A dictionary of students in school mapping ID to age.
- **`uids_in_school_by_age`** (*`dict`*) – A dictionary of students in school mapping age to the list of IDs with that age.
- **`ages_in_school_count`** (*`dict`*) – A dictionary mapping age to the number of students with that age.
- **`age_brackets`** (*`dict`*) – A dictionary mapping age bracket keys to age bracket range.
- **`age_by_brackets_dic`** (*`dict`*) – A dictionary mapping age to the age bracket range it falls within.
- **`contact_matrix_dic`** (*`dict`*) – A dictionary of age specific contact matrix for different physical contact settings.
- **`verbose`** (*`bool`*) – If True, print statements about the generated schools as they're being generated.

Returns Two lists of lists and third flat list, the first where each sublist is the ages of students in the same school, and the second is the same list but with the IDs of each student in place of their age. The third is a list of the school types for each school, where each school has a single string to represent it's school type.

5.1.12 synthpops.version module

5.1.13 synthpops.workplaces module

synthpops.workplaces.**get_uids_potential_workers**(*syn_school_uids*, *employment_rates*, *age_by_uid_dic*)

Get IDs for everyone who could be a worker by removing those who are students and those who can't be employed officially.

Parameters

- **syn_school_uids** (*list*) – A list of lists where each sublist represents a school with the IDs of students in the school.
- **employment_rates** (*dict*) – The employment rates by age.
- **age_by_uid_dic** (*dict*) – A dictionary mapping ID to age for individuals in the population.

Returns A dictionary of potential workers mapping their ID to their age, a dictionary mapping age to the list of IDs for potential workers with that age, and a dictionary mapping age to the count of potential workers left to assign to a workplace for that age.

synthpops.workplaces.**generate_workplace_sizes**(*workplace_size_distr_by_bracket*, *workplace_size_brackets*, *workers_by_age_to_assign_count*)

Given a number of individuals employed, generate a list of workplace sizes to place everyone in a workplace.

Parameters

- **workplace_size_distr_by_bracket** (*dict*) – The distribution of binned workplace sizes.
- **worplace_size_brackets** (*dict*) – A dictionary of workplace size brackets.
- **workers_by_age_to_assign_count** (*dict*) – A dictionary mapping age to the count of employed individuals of that age.

Returns A list of workplace sizes.

synthpops.workplaces.**get_workers_by_age_to_assign**(*employment_rates*, *potential_worker_ages_left_count*, *uids_by_age_dic*)

Get the number of people to assign to a workplace by age using those left who can potentially go to work and employment rates by age.

Parameters

- **employment_rates** (*dict*) – A dictionary of employment rates by age.
- **potential_worker_ages_left_count** (*dict*) – A dictionary of the count of workers to assign by age.
- **uids_by_age_dic** (*dict*) – A dictionary mapping age to the list of ids with that age.

Returns A dictionary with a count of workers to assign to a workplace.

synthpops.workplaces.**assign_rest_of_workers**(*workplace_sizes*, *potential_worker_uids*, *potential_worker_uids_by_age*, *workers_by_age_to_assign_count*, *age_by_uid_dic*, *age_brackets*, *age_by_brackets_dic*, *contact_matrix_dic*, *verbose=False*)

Assign the rest of the workers to non-school workplaces.

Parameters

- **`workplace_sizes`** (*`list`*) – list of workplace sizes
- **`potential_worker_uids`** (*`dict`*) – dictionary of potential workers mapping their id to their age
- **`potential_worker_uids_by_age`** (*`dict`*) – dictionary mapping age to the list of worker ids with that age
- **`workers_by_age_to_assign_count`** (*`dict`*) – dictionary of the count of workers left to assign by age
- **`age_by_uid_dic`** (*`dict`*) – dictionary mapping id to age for all individuals in the population
- **`age_brackets`** (*`dict`*) – dictionary mapping age bracket keys to age bracket range
- **`age_by_brackets_dic`** (*`dict`*) – dictionary mapping age to the age bracket range it falls in
- **`contact_matrix_dic`** (*`dict`*) – dictionary of age specific contact matrix for different physical contact settings
- **`verbose`** (*`bool`*) – If True, print statements about the generated schools as teachers are being added to each school.

Returns List of lists where each sublist is a workplace with the ages of workers, list of lists where each sublist is a workplace with the ids of workers, dictionary of potential workers left mapping id to age, dictionary mapping age to a list of potential workers left of that age, dictionary mapping age to the count of workers left to assign.

CHAPTER

SIX

GLOSSARY

contact layers Each of the layers of the population network that is a representation of all of the pairwise connections between people in a given location, such as school, work, or households.

node In network theory, the discrete object being represented. In SynthPops, nodes represent people and can have attributes like age assigned.

edge In network theory, the interactions between discrete objects. In SynthPops, edges represent interactions between people, with attributes like the setting in which the interactions take place (for example, household, school, or work). The relationship between the interaction setting and properties governing disease transmission, such as frequency of contact and risk associated with each contact, is mapped separately by Covasim or other *agent-based model*. SynthPops reports whether the edge exists or not.

agent-based model A type of simulation that models the actions and interactions of autonomous agents (both individual and collective entities such as organizations or groups).

time step A discrete number of hours or days in which the "simulation states" of all "simulation objects" (interventions, infections, immune systems, or individuals) are updated in a simulation. Each time step will complete processing before launching the next one. For example, a time step would process the migration data for populations moving between nodes via rail, airline, and road. The migration of individuals between nodes is the last step of the time step after updating states.

household contact layer The layer in the population network that represents all of the pairwise connections between people in households. All people must be part of the household contact layer, though some households may consist of a single person.

school contact layer The layer in the population network that represents all of the pairwise connections between people in schools. This includes both students and teachers. The school and workplace contact layers are mutually exclusive, someone cannot be both a student and a worker.

workplace contact layer The layer in the population network that represents all of the pairwise connections between people in workplaces excluding teachers in schools. The school and workplace contact layers are mutually exclusive, someone cannot be both a student and a worker.

PYTHON MODULE INDEX

INDEX