emodpy-covid

Institute for Disease Modeling

Mar 08, 2022

CONTENTS

emodpy-covid is a collection of Python scripts and utilities created to streamline user interactions with EMOD and idmtools for modeling COVID-19. Much of the functionality is inherited from the emod_api package and emodpy package.

Additional information about how to use idmtools can be found at in Welcome to idmtools. Additional information about EMOD generic parameters can be found in EMOD parameter reference.

CHAPTER

ONE

EMODPY-COVID INSTALLATION

Follow the steps below to install emodpy-covid.

1.1 Prerequisites

First, ensure the following prerequisites are met.

- Windows 10 Pro or Enterprise, Linux, or Mac
- Python 3.9 64-bit (https://www.python.org/downloads/release)
- A file that indicates the pip index-url:
  - Windows
  - Linux

  In C:\Users\Username\pip\pip.ini, containing the following:

```
[global]
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

  In $HOME/.config/pip/pip.conf, containing the following:

```
[global]
index-url = https://packages.idmod.org/api/pypi/pypi-production/simple
```

1.2 Installation instructions

1. Open a command prompt and create a virtual environment in any directory you choose. The command below names the environment "v-emodpy-covid", but you may use any desired name:

```
python -m venv v-emodpy-covid
```

2. Activate the virtual environment:
   - Windows
   - Linux

   Enter the following:

```
v-emodpy-covid\Scripts\activate
```

Enter the following:

```
source v-emodpy-covid/bin/activate
```

3. Install emodpy-covid packages:

```
pip install emodpy-covid
```

If you are on Linux, also run:

```
pip install keyrings.alt
```

4. When you are finished, deactivate the virtual environment by entering the following at a command prompt:

```
deactivate
```

CHAPTER

TWO

FREQUENTLY ASKED QUESTIONS

As you get started with emodpy-covid, you may have questions. The most common questions will be added below. For questions related to functionality in related packages, see the following documentation:

- Frequently asked questions for EMOD
- Frequently asked questions for idmtools
- Frequently asked questions for emod-api
- Frequently asked questions for emodpy

CHAPTER

THREE

API REFERENCE

3.1 emodpy_covid package

3.1.1 Subpackages

emodpy_covid.demographics package

Submodules

emodpy_covid.demographics.CovidDemographics module

emodpy_covid.demographics.CovidDemographics.**mat_magic**(*geog='AFRO:GHA'*, *spike=False*, *data_dir=None*)

HINT matrix construction :param geog: Geography Key. :type geog: str :param spike: Contact spike :type spike: bool

Returns tuple

Return type HINT setup inputs

class emodpy_covid.demographics.CovidDemographics.**CovidDemographics**(*nodes*, *idref='Gridded world grump2.5arcmin'*, *base_file=None*)

Bases: emod_api.demographics.Demographics.Demographics

This class is derived from emod_api.demographics' Demographics class so that we can set certain defaults for Covid in construction. Keen observers will note thatt SetDefaultProperties does not look like a Covid-specific function. But as we add other disease types the generalizations and speicfics will become clearer. The architectural point is solid.

emodpy_covid.demographics.CovidDemographics.**from_synth_pop**(*tot_pop=1000000.0*, *num_nodes=100*, *frac_rural=0.3*, *id_ref='from_synth_pop'*)

This function creates a multi-node CovidDemographics instance from the params you give it.

Parameters

- **tot_pop** – Total population
- **num_nodes** – Number of nodes

- **frac_rural** – Floating point fraction that determines in a single param how population is spread across the nodes
- **id_ref** – Tag that appears in demographics file.

Returns GENERIC_SIM demographics instance which can be customized and/or written to file.

emodpy_covid.demographics.CovidDemographics.**fromBasicNode**(*lat=0*, *lon=0*, *pop=1000000.0*, *name=1*, *forced_id=1*)

This function creates a single-node CovidDemographics instance from the params you give it.

Parameters

- **lat** – latitude (not really used)
- **lon** – longitude (not really used)
- **pop** – population. Defaults to 1 million.
- **name** – node name (not really used)
- **forced_id** – node id (not really used)

Returns GENERIC_SIM demographics instance which can be customized and/or written to file.

emodpy_covid.demographics.contact_mat module

emodpy_covid.interventions package

Submodules

emodpy_covid.interventions.complex_import module

emodpy_covid.interventions.complex_import.**ComplexImportationEvent**(*dips=[0.00012919896640826872]*, *durs=[100000]*, *timestep=1*, *nods=[]*)

Import the Disease

Parameters

- **DailyImportPressures** – a rate of per-day importation for each node that the intervention is distributed to
- **Durations** – The durations over which to apply import pressure.
- **timestep** – Day to start campaign event

Be careful when configuring import pressure in multi-node simulations. Daily_Import_Pressures defines a rate of per-day importation for each node that the intervention is distributed to. In a 10 node simulation with Daily_Import_Pressures = [0.1, 5.0], the total importation rate summed over all nodes will be 1/day and 50/day during the two time periods. You must divide the per-day importation rates by the number of nodes.

emodpy_covid.interventions.covid_vaccine module

emodpy_covid.interventions.covid_vaccine.**SimpleSample1**(*timestep*, *coverage=1.0*)

emodpy_covid.interventions.covid_vaccine.**SimpleSample2**(*timestep*, *coverage=1.0*)

emodpy_covid.interventions.covid_vaccine.**new_intervention_as_file**(*timestep*, *filename=None*)

emodpy_covid.interventions.matrix_swap module

emodpy_covid.interventions.matrix_swap.**MatrixSwap**(*new_hint_matrix*, *timestep=1*)

emodpy_covid.interventions.mevacc module

emodpy_covid.interventions.mevacc.**MEVacc**(*coverage=1.0*, *qual_ac=0*, *qual_trn=1.0*, *group_names=''*, *timestep=1*, *nods=[]*)

emodpy_covid.interventions.quarantine module

emodpy_covid.interventions.quarantine.**Quarantine**(*coverage=1.0*, *delay=0*, *quality=1.0*, *trigger='NoTrigger'*, *timestep=1*, *nods=[]*)

emodpy_covid.microstructure package

Submodules

emodpy_covid.microstructure.add_hh_to_demo module

emodpy_covid.microstructure.change_ser_pop module

emodpy_covid.microstructure.hh_gen_toy module

emodpy_covid.microstructure.test_change_ser_pop module

emodpy_covid.microstructure.test_spops_interface module

emodpy_covid.microstructure.test_toy_pop module

emodpy_covid.mid package

Submodules

emodpy_covid.mid.dtk_in_process_close_schools module

emodpy_covid.mid.dtk_in_process_close_schools.**debug = False**

default = 30. Only used if timestep passed in is -1. min_school_id: the minimum of the group ids for schools. Defaults to 1. max_school_id: the maximum of the group ids for schools. Defaults to 21, but that is very much dependent on your configuration. school_closure_duration: the duration of the school closure. It is completely relaxed after this. Defaults to 60. debug = False

Type school_closure_timestep

emodpy_covid.mid.dtk_in_process_close_schools.**app**(*timestep*)

emodpy_covid.mid.dtk_in_process_isolate_infecteds module

emodpy_covid.mid.dtk_in_process_isolate_infecteds.**app**(*timestep*)

emodpy_covid.mid.dtk_in_process_isolate_symptomatic_nosql module

emodpy_covid.mid.dtk_in_process_isolate_symptomatic_nosql.**get_event**(*timestep*, *route*, *efficacy*, *intervention_name*)

emodpy_covid.mid.dtk_in_process_isolate_symptomatic_nosql.**app**(*timestep*)

emodpy_covid.mid.dtk_in_process_outbreak module

emodpy_covid.mid.dtk_in_process_outbreak.**app**(*timestep*, *cases=1*)

emodpy_covid.mid.dtk_in_process_qtine_fams module

emodpy_covid.mid.dtk_in_process_qtine_fams.**get_events**(*timestep*)

emodpy_covid.mid.dtk_in_process_qtine_fams.**app**(*timestep*)

emodpy_covid.mid.dtk_in_process_ten_four module

emodpy_covid.mid.dtk_in_process_ten_four.**app**(*timestep*, *cases=1*)

So here's the plan. Divide population in two. Let's try male and female for now. Group 0 (male) goes to work/school on Day 1, for 4 days, then returns home for 10 days. Group 1 (female) goes to work/school on Day 8, for 4 days, then returns home for 10 days. Implement this as RASV, CONTACT, on day 5 for group 0 with 10 day duration, and RASV, CONTACT, on day 12 for group 0 with 10 day duration. Repeat every 14 days.

emodpy_covid.mid.dtk_in_process_trace_and_isolate_infecteds module

emodpy_covid.mid.dtk_in_process_trace_and_isolate_infecteds.**app**(*timestep*)

emodpy_covid.mid.dtk_in_process_wfh module

emodpy_covid.mid.dtk_in_process_wfh.**app**(*timestep*)

emodpy_covid.mid.schema_to_class module

class emodpy_covid.mid.schema_to_class.**ReadOnlyDict**

Bases: collections.OrderedDict

set_schema(*schema*)

finalize()

emodpy_covid.mid.schema_to_class.**get_default_campaign_event**()

emodpy_covid.mid.schema_to_class.**get_class_with_defaults**(*classname*, *schema_path='schema.json'*)

Very handy little class – that needs a lot more error handling and robustification – that gets you the default config for an event coordinator or intervention from the schema.

emodpy_covid.post package

Submodules

emodpy_covid.post.dtk_post_process_add_cum_inf module

emodpy_covid.post.dtk_post_process_newinf module

emodpy_covid.post.dtk_post_process_newinf_sql2csv module

emodpy_covid.post.dtk_post_process_trace module

3.1.2 Submodules

emodpy_covid.basic_demographics module

emodpy_covid.config module

emodpy_covid.config.**app**(*eradication_path*)

We expect this to be executed module-style (as a tool, not a script). This script creates a default covid-19 config, using emod_api.

emodpy_covid.demog module

emodpy_covid.demog.**get_demog**()

This code is expected to be run module-style, not from code. It createa a default demographics.json file that can be consumed by DTK for covid-19 modeling.

PYTHON MODULE INDEX

e

INDEX

F

G

M

N

Q

R

S