Bucky

Release .1

Matt Kinsey, Kate Tallaksen, Freddy Obrecht

Oct 27, 2020

CONTENTS:

CHAPTER

ONE

QUICKSTART

1.1 Requirements

The Bucky model currently supports Linux and OSX and includes GPU support for accelerated modeling and processing. Anaconda environment files are provided for installation of dependencies.

1.2 Installation

Clone the repo using git:

```
git clone https://github.com/mattkinsey/bucky.git
```

Create Anaconda environment using *environment.yml* or *environment_gpu.yml* if using the GPU.

```
conda env create --file environment[_gpu].yml
conda activate bucky[_gpu]
```

Optional: Data and output directory default locations are defined in *config.yml*. Edit this file to change these.

Download the required US data using the provided shell script:

```
./get_US_data.sh
```

1.3 Running the Model

In order to illustrate how to run the model, this section contains the commands needed to run a small simulation. First, create the intermediate graph format used by the model. This graph contains county-level data on the nodes and mobility information on the edges. The command below creates a US graph for a simulation that will start on October 1, 2020.

```
./bmodel make_input_graph -d 2020-10-01
```

After creating the graph, run the model with 100 iterations and 20 days:

```
./bmodel model -n 100 -d 20
```

This will create a folder in the *raw_output* directory with the unique run ID. The script *postprocess* processes and aggregates the Monte Carlo runs. This script by default postprocesses the most recent data in the *raw_output* directory and aggregates at the national, state, and county level.

```
./bmodel postprocess
```

1.4 Visualizing Results

To create plots:

```
./bmodel viz.plot
```

Like postprocessing, this script by default creates plots for the most recently processed data. Plots will be located in *output/<run_id>/plots*. These plots can be customized to show different columns and historical data. See the documentation for more.

1.5 Lookup Tables

During postprocessing, the graph file is used to define geographic relationships between administrative levels (e.g. counties, states). In some cases, a user may want to define custom geographic groupings for visualization and analysis. For example, the National Capital Region includes counties from Maryland and Virginia along with Washington, DC. An example lookup table for this region (also known as the DMV) is included in the repo, *DMV.lookup*.

To aggregate data with this lookup table, use the flag *–lookup* followed by the path to the lookup file:

```
./bmodel postprocess --lookup DMV.lookup
```

This will create a new directory with the prefix *DMV_* in the default output directory (output/DMV_<run_id>/). To plot:

```
./bmodel model viz.plot --lookup DMV.lookup
```

CHAPTER

TWO

INSTALLATION GUIDE

1. To begin, first checkout the code from GitLab:

```
git clone https://gitlab.com/kinsemc/bucky.git
```

2. Next set up the enviroment required to run the model, first making sure Anaconda is installed.

Note: Anaconda can be downloaded from https://docs.anaconda.com/anaconda/install/

Included in the repository are two yaml formatted Anaconda enviroment specs:

- **enviroment.yml**: Contains the standard packages required to run the model
- **enviroment_gpu.yml**: Standard enviroment + CUDA/CuPy for GPU acceleration. CuPy will be used to replace all references to numpy in the model itself.

Note: CuPy requires an NVIDIA GPU and will only increase performance for model runs over large geographic area (e.g. the whole US)

To install and activate the appropriate enviroment:

```
cd bucky
conda env create --file enviroment.yml
conda activate bucky
```

or

```
cd bucky
conda env create --file enviroment_gpu.yml
conda activate bucky_gpu
```

3. Finally, if you wish to use custom paths to store the data associated with the model (either inputs or outputs), simply edit the contents of config.yml in the root of the repository

Note: It is recommended to use high speed storage for <raw_output_dir> if possible as that will have an impact on runtimes.

2.1 Downloading Input Datasets

The model depends on a number of input datasets being available in the <data_dir> specified in config.yml. To automatically download them just using the *get_US_data.sh* script provided in the root of the repository (this will take some time for the initial download):

```
chmod +x ./get_US_data.sh
./get_US_data.sh
```

The following datasets will be automatically downloaded:

- **COVID-19 Data Repository by the Center for Systems Science and Engineering at Johns Hopkins University**
  - COVID-19 Case and death data on the county level
  - GitHub
- **Descartes Labs: Data for Mobility Changes in Response to COVID-19**
  - State and county-level mobility statistics
  - GitHub
- **COVID Exposure Indices from PlaceIQ movement data**
  - State and county-level location exposure indices
  - Reference: *Measuring movement and social contact with smartphone data: a real-time application to COVID-19* by Couture, Dingel, Green, Handbury, and Williams Link
  - GitHub
- **The COVID Tracking Project at The Atlantic**
  - COVID-19 case and death data at the state level
  - GitHub
- **US TIGER shapefiles from the US Census**
  - Link
- **US Census Bridged-Race Population estimates**
  - Link
- **Social Contact Matrices for 152 Countries**
  - *Projecting social contact matrices in 152 countries using contact surveys and demographic data*, Prem et al.
  - Paper
- **USAFacts Coronavirus Stats and Data**
  - County-level coronavirus cases and deaths
  - Link

CHAPTER

THREE

MODEL INFORMATION

3.1 Model Description

The JHUAPL-Bucky model is a COVID-19 metapopulation compartment model initially designed to estimate medium-term (on the order of weeks) case incidence and healthcare usage at the second administrative (admin-2, ADM2) level (counties in the United States; cities or districts in various countries). These ADM2 regions are all coupled using mobility information to approximate the both inter- and intra-regional contacts between the members of the populations. Using the historical case and death data, local demographic data (see *Graph Information*), and a set of *parameters* derived from empirical studies, the model infers a number of localized features (see table below) that are related to spread of COVID-19. Projecting forward in time, Bucky then utilizes an age stratified compartment model to not only estimate the case load but additionally provide outputs relating to the healthcare burden of each locality.

These time forecasts are performed a large number of times (Monte Carlo experiments), with each individual simulation using minor modifications to the input parameters at random, scaled to the uncertainty of the estimates. The resulting collection of simulations is then used to obtain probabilistic estimates for all output variables.

3.1.1 Model Overview

At its base, the Bucky model is a spatially distributed SEIR model. SEIR models are a class of deterministic models used to model infectious diseases that are spread by person-to-person transmission in a population. The simplest versions of such models are systems of ordinary differential equations and are analysed mathematically [Het89].

Within the context of an SEIR model, disease dynamics are modeled over time by moving the population through a series of compartments (otherwise known as "bins" or "states"). Those states are as follows:

- susceptible (S): the fraction of the population that could be potentially subjected to the infection;
- exposed (E): the fraction of the population that has been infected but does not show symptoms yet;
- infectious (I): the fraction of the population that is infective after the latent period;
- recovered (R): The fraction of the population that have been infected and recovered from the infection.

The total population is represented by the sum of the compartments. Basic assumptions of this type of model include:

- Once the model is initialized, no individuals are added to the susceptible group. It follows that births and natural deaths are unaccounted for, migration in/out of the region is frozen for the duration of a simulation, and none of the population has been vaccinated or is immune to the pathogen;
- The population within each strata is uniform and each pair of individuals within the strata are equally likely to interact;
- The probability of interaction between individuals in the population is not rare;
- Once infected, an individual cannot be reinfected with the virus.

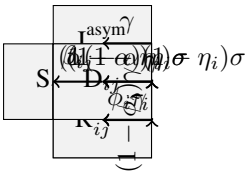

Fig. 1: Model

Note: The compartments E, I^{asym}, I^{mild}, I^{hosp} and R^{hosp} are gamma-distributed with shape parameters specified in the configuration file.

Variable	Description
S_{ij}	Proportion of individuals who are susceptible to the virus
E_{ij}	Proportion of individuals who have been exposed to the virus
I_{ij}^{hosp}	Proportion of individuals that are exhibiting severe disease symptoms and are in need of hospitalization
I_{ij}^{mild}	Proportion of individuals that are exhibiting mold disease symptoms
I_{ij}^{asymp}	Proportion of individuals who are infected but asymptomatic
R_{ij}	Proportion of individuals who have recovered from the virus and are no longer capable of infecting other individuals
R_{ij}^{hosp}	Proportion of individuals who have recovered from the virus after a period of time in a hospital
D_{ij}	Proportion of individuals who have succumbed as a direct result of the virus

Parameter	Description
β_{ij}	Force of infection on a member of age group *i* in location *j*
$\frac{1}{\sigma}$	Viral latent period
α	Rate of infections that are asymptomatic
η_i	Fraction of cases necessitating hospitalization
ϕ_i	Case fatality rate for age group *i*
$\frac{1}{\gamma}$	Infectious period
τ_i	Recovery period from severe infection for age group *i*

The Bucky model consists of a collection of coupled and stratified SEIR models. Since COVID-19 exhibits heavily age dependent properties, wherein a majority of severe cases are in older individuals, SEIR models are stratified via the age demographic structure of a geographic region in order to get accurate estimates of case severity and deaths. Additionally, to model the spatial dynamics of COVID spread, we consider a set of SEIR sub-models at the smallest geographic level for which we have appropriate data.

The basic structure of the model is displayed in the diagram above. Age is denoted by index *i*, and geographic regions are denoted by index *j*. Within each strata, Bucky models the susceptible and exposed populations, followed by one of three possible infected states: asymptomatic (I^{asym}), mild (I^{mild}), and severe (I^{hosp}). Members of the population who are either asymptomatic or exhibit mild symptoms recover from the virus at a rate γ. Those who exhibit severe symptoms and are in need of healthcare support will either recover after a period of illness at rate $1/\tau_i$ or expire as a result of the virus at rate $\phi_i\gamma$.

A critical component of the Bucky model is the parameterization of the model. A number of parameters must be derived and/or estimated from their original data sources. These include, but are not limited to those listed in tables above as well as local estimates of local case doubling time, case reporting rate, case fatality rate, and the case hospitalization rate. Further details of these quantities as well as how they are estimated are given in the *Model Input*

and Ouput section. All parameter estimation for the model includes the basic assumption that, once estimated and initialized, these parameters remain constant during the simulation period.

Coupling individual age and geographically stratified sub-models occurs across a number of dimensions including disease state. Sub-models are coupled together using both the spatial mobility matrix and age-based contact matrices. Modeling of the overall interaction rates between geographic locations and age groups is an important component in accurately modeling non-pharmaceutical Interventions (NPIs). Bucky accounts for the implementation of NPIs (e.g. school closures, border closures, face mask wearing) via modifying either the social contact matrices or the basic reproductive number, R_0. For further details, see *Non-pharmaceutical Interventions*.

All together, these components contribute to a model that is adaptable to a number of contexts. Bucky is calibrated to the uncertainties in both the case data and the disease parameters, leading to a model that is robust to both the quality and resolution of available input data.

3.2 Model Input and Output

3.2.1 Input

The Bucky model uses two main sources of input: the input graph and CDC-recommended parameters.

Input Graph

The input graph contains data regarding demographics, geographic information, and historical case information. For details, see *Graph Information*.

CDC-Recommended Parameters

The Centers for Disease Control and Prevention (CDC) has published pandemic planning scenarios [fDCP+20] which contain recommended parameters describing biological and epidemiological parameters. Of these five planning scenarios, Bucky utilizes scenario 5, which contains the CDC's current best estimates for disease severity and transmission. These parameters are described in detail, based on information available from the CDC, and summarized in the table below. CDC-recommended parameters are controlled by parameter files located in the `par` directory.

Parameter Description	Bucky Variable Name	Value (Interquartile Range)
Mean generation interval	T_g	7.5 (5.5, 8.5)
Mean serial interval	T_s	6 (5, 7)
Fraction of infections that are asymptomatic	asym_frac	0.4
Relative infectiousness of asymptomatic individuals	rel_inf_asym	0.75
Percentage of transmission prior to symptom onset	frac_trans_before_sym	0.5
Case fatality ratio	F	• 0-49 years: 0.0005 • 50-64 years: 0.002 • 65+ years: 0.013
Case hospitalization ratio	H	• 0-49 years: 0.017 • 50-64 years: 0.045 • 65+ years: 0.074
TIme from symptom onset to hospitalization	I_TO_H_TIME	• 0-49 years: 6 days • 50-64 years: 6 days • 65+ years: 4 days
Duration of hospitalization	H_TIME	• 0-49 years: 4.9 days • 50-64 years: 7.6 days • 65+ years: 8.1 days
Time between death and reporting	D_REPORT_TIME	• 0-49 years: 7.1 days • 50-64 years: 7.2 days • 65+ years: 6.6

Disease Transmission

The following parameters describe the transmissibility of the virus. The **percentage of infections that are asymptomatic**, `asym_frac`, refers to the percentage of infections that will never develop symptoms. This is a difficult parameter to estimate due to logistical complications (individuals would need to be tested to ensure they remain asymptomatic while infectious) and because the level of asymptomatic infections varies by age. The best estimate for this parameter is the midpoint between the lower bound of [BCB+20], the upper bound of [PTC+20], which corresponds to the estimates from [OT20].

The **relative infectiousness of asymptomatic individuals** compared to symptomatic individuals `rel_inf_asym` is calculated using upper and lower bounds on the difference in viral dynamics between asymptomatic and symptomatic cases. The lower bound is derived from data indicating that more severe cases have higher viral loads [LYW+20] and a study that indicates symptomatic cases shed for longer and have higher viral loads than asymptomatic cases [NYS+20]. Other studies indicate that both symptomatic and asymptomatic cases have similar duration and viral shedding [LKL+20], which is used as the upper bound.

The final parameter relating to disease transmission is the **fraction of transmission prior to symptom onset** `frac_trans_before_sym` which corresponds to the percentage of new cases that were caused by transmission from an individual before they become symptomatic. The lower bound is derived from [HLW+20], with the upper bound derived from [CGM+20].

Disease Characteristics and Severity

The mean serial interval, `Ts`, is the time in days from exposure to onset of symptoms and is taken from [MCH+20]. The mean generation interval, `Tg`, is the period of time (in days) between symptom onset for one individual and symptom onset for a person they have infected. This value is from [HLW+20].

The case fatality ratio (**CFR**) is the number of individuals who will die of the disease; the case hospitalization-severity ratio (**CHR**) corresponds to the number of cases that are severe and necessitate hospitalization. Within the context of the United States, this ratio corresponds to the individuals admitted to a hospital. In a context where access to medical care is limited, this ratio corresponds to the ratio of individuals who exhibit severe disease symptoms.

Hospital-related parameters are derived using data from COVID-Net [CDC] and the CDC's Data Collation and Integration for Public Health Event Response (DCIPHER). All data is taken from the period between March 1, 2020 to July 15, 2020 unless otherwise noted. The time it takes from symptom onset to hospitalization in days is denoted by `I_to_H_time`. The number of days an individual will be hospitalized is `H_TIME`. Finally, the number of days between death and reporting is `D_REPORT_TIME`.

3.2.2 Output

The Bucky model generates one file per Monte Carlo run. This data is post-processed to combine data across all dates and simulations. It can then be aggregated at desired geographic levels. A separate file is created for each requested administrative level, with each row indexed by data, admin ID, and quantile. The columns of this output file are described in the tables below.

Aggregated files are placed in subfolder named using the Monte Carlo ID within the specified output directory. Filenames are constructed by appending the aggregation level with the aggregation type (quantiles vs mean). For example, the following file contains quantiles at the national level:

`/output/2020-06-10__14_13_04/adm0_quantiles.csv`

An example output directory structure is shown below:

```
2020-07-28__15_21_52/
├── adm0_quantiles.csv
├── adm1_quantiles.csv
├── adm2_quantiles.csv
├── maps
│   └── ADM1
│       ├── adm1_AlabamaDailyReportedCases2020-07-26.png
│       ├── adm1_AlabamaDailyReportedCases2020-08-02.png
│       ├── ...
└── plots
    ├── ADM1
    │   ├── DailyReportedCases_Alabama.png
    │   ├── ...
    ├── US.csv
    └── US.png
```

Column and Index Names

Index name	Description
adm*	The adm ID corresponding to the geographic level (i.e. adm2 ID)
date	The date
quantile	Quantile value

Column name	Description
case_reporting_rate	Case reporting rate
active_asymptomatic_cases	Current number of actively infectious but asymptomatic cases
cumulative_cases	Cumulative number of cumulative cases (including unreported)
cumulative_deaths	Cumulative number of deaths
cumulative_deaths_per_100k	Cumulative number of deaths per 100,000 people
cumulative_reported_cases	Cumulative number of reported cases
cumulative_reported_cases_per_100k	Number of reported cumulative cases per 100,000 people
current_hospitalizations	Current number of hospitalizations
current_hospitalizations_per_100k	Number of current hospitalizations per 100,000 people
current_icu_usage	ICU bed usage
current_vent_usage	Current ventilator usage
total_population	Population
daily_cases	Number of daily new cases (including unreported)
daily_deaths	Number of daily new deaths
daily_hospitalizations	Number of daily new hospitalizations
daily_reported_cases	Number of reported daily new cases
doubling_t	Local doubling time as estimated from the historical data
R_eff	Local effective reproductive number

3.3 Graph Information

The Bucky model does not do any data manipulation, smoothing, or correcting to the data it receives from the graph (by design). If data needs to be manipulated or corrected, it should be done before it is placed on the graph.

The graph is created using admin2-level data. If data can not be found at the admin2-level, admin2-level information can be extrapolated using admin2 population and national or state level data (this is expanded upon in the *Population Data* section).

The following data sources are used to create the graph:

- admin2-level shapefile
- admin2-level population data stratified by age
- Historical admin2-level case and death data
- Contact matrix information for the country
- Mobility data (or a proxy)

All data is placed into a single dataframe, joined by the admin2-level key (e.g., FIPS for United States), with the exception of mobility data (which is used to create edges, not nodes).

3.3.1 Graph-Level Attributes

Administrative information is placed on the top graph-level. For example:

```
'adm1_key' : 'adm1',
'adm2_key' : 'adm2',
'adm1_to_str' : {1 : 'Alabama'}, ...,
'adm0_name': 'US',
'start_date' : '2020-09-25'
```

Note: *adm1_to_str* is a dict with key-value pairs indicating the adm1 names for each adm1 value appearing in the graph.

Contact matrices are also on this level under the key *'contact_mats'*.

3.3.2 Sample Node

The following is an example node on the graph for the United States. The rest of the documentation will describe what data is necessary to construct this node.

```
(0,
 {'adm1': 1,
  'Confirmed': 1757.0,
  'Deaths': 25.0,
  'adm2_name': 'Autauga County',
  'N_age_init': array([3364., 3423., 3882., 3755., 3173., 3705., 3461., 3628., 3616.,
         3966., 3811., 3927., 3237., 2589., 2311., 3753.]),
  'Population': 55601.0,
  'IFR': array([6.75414158e-06, 1.24643105e-05, 2.26550214e-05, 4.05345945e-05,
         7.68277004e-05, 1.38382882e-04, 2.54273120e-04, 4.63844627e-04,
         8.51898589e-04, 1.55448599e-03, 2.87077658e-03, 5.20528393e-03,
         9.47735996e-03, 1.73603179e-02, 3.14646839e-02, 9.38331984e-02]),
  'case_hist': array([1207.64227528, 1234.9656055 , 1243.85366911, 1244.13444753,
         1255.27521116, 1268.95333353, 1270.38458817, 1288.05778954,
         1295.55174933, 1297.29129258, 1312.35308192, 1321.2898892 ,
         1323.10534634, 1354.97350342, 1358.88036484, 1362.43488575,
         1377.67551466, 1392.4338964 , 1406.70605635, 1446.3924143 ,
         1450.47616771, 1462.67851762, 1458.98710032, 1470.52271903,
         1481.12998684, 1501.75698721, 1508.06090303, 1513.9178672 ,
         1518.26245703, 1532.99858052, 1553.97101414, 1564.24619451,
         1579.10859377, 1590.56170754, 1597.77332362, 1616.97996262,
         1619.        , 1624.        , 1664.        , 1673.        ,
         1690.        , 1691.        , 1714.        , 1715.        ,
         1738.        , 1757.        ]),
  'death_hist': array([22.76748794, 22.80142062, 22.81307638, 22.79580414, 22.
↪79344408,
         22.79578013, 22.81581338, 22.80532061, 22.7902682 , 22.79603286,
         22.79689139, 22.79601336, 22.79344923, 22.85912123, 22.90405033,
         22.91397178, 22.97898824, 23.02565004, 23.05597481, 23.09719551,
         23.13913548, 24.12323294, 24.17184064, 24.2852927 , 24.38579416,
         24.41284998, 24.41330133, 24.41175889, 24.40910247, 24.41419481,
         24.43286524, 24.47610337, 24.52580854, 24.5245916 , 24.53522989,
         24.54591406, 24.        , 24.        , 24.        , 24.        ,
         24.        , 24.        , 25.        , 25.        , 25.        ,
```

(continues on next page)

(continued from previous page)

```
        25.        ]),
 'adm2': 1001.0})
```

3.3.3 Population Data

Population data should be at a admin2 level and stratified in 16 5-year age bins (if using Prem et al contact matrices):

- 0-4 years
- 5-9 years
- 10-14 years
- 15-19 years
- 20-24 years
- 25-29 years
- 30-34 years
- 35-39 years
- 40-44 years
- 45-49 years
- 50-54 years
- 55-59 years
- 60-64 years
- 65-69 years
- 70-74 years
- 75+ years

If population data for an admin2 area is known (i.e. number of total people per admin2), but it is not age-stratified, this data can be extrapolated assuming age-stratified population data exists at some level. For example, assume a country has age-stratified data provided at the national-level. To get the admin2-level age data, the data is separated into the 16 bins (as a 1-dimensional array of length 16). These bins are then normalized by dividing by the sum. Then, the fraction of people living in the admin2 is calculated by dividing admin2 population by the total national population. For each district, this fraction is multiplied by the age vector to produce a admin2-level age vector. This vector is placed on the node under the key *N_age_init*.

The total population for an admin2 is placed on the node under the key *Population*.

3.3.4 Case Data

Case data should be at the admin2-level and include cumulative data as of the start date of the simulation and historical data for the 45-day period preceding the start date:

- case_hist: **Cumulative** historical case data
- death_hist : **Cumulative** historical death data

Historical data is structured as numerical vectors on the node with the keys *case_hist*, *death_hist*. Historical data for every node must have data points for the 45 days preceding the simulation. If there are known errors in the historical data, they must be corrected before being placed on the graph.

3.3.5 Contact Matrices

Currently, contact matrix data is downloaded from here, which has contact matrices for 152 countries. If a country does not appear in this dataset, a country culturally close can be substituted (for example, Pakistan's contact rates were used for Afghanistan), or another dataset can be used. If another dataset is used, the contact matrix must be formatted such that it has the same shape as the number of age demographic bins (i.e. if there are 16 bins, the matrix must be of size 16 x 16).

3.3.6 Mobility Data

Mobility data is used to construct the edges of the graph. Mobility data, or a proxy for it, is used to describe the contact rates between counties.

The baseline mobility data shows up as an edge attributed called *weight*. *R0_frac* is a factor that is multiplied with the baseline mobility value to model the effect of NPIs, etc., on mobility. For example, given baseline mobility data from February 2020, *R0_frac* would be computed by dividing recent mobility data values with the February 2020 baseline. *R0_frac* exists to provide a knob to tune during the simulation to model NPIs.

3.4 Non-pharmaceutical Interventions

Non-pharmaceutical interventions (NPIs) are mitigations, apart from getting vaccinated and taking medicine, that people and communities can take to help slow the spread of communicable diseases. As a vaccine for COVID-19 has yet to be deployed, NPIs are among the best strategies for controlling the spread of the current COVID-19 virus. The structure of the Bucky model allows for the incorporation of NPIs via the modification of a combination of the following : the basic reproduction number, local contact matrices, and inter-regional mobility matrices.

For each country an initial list of NPIs was obtained from the ACAPS COVID-19 Government Measures Dataset . This dataset is complemented with additional qualitative information from in-country stakeholders. The estimated compliance level are tailored to specific countries.

3.4.1 Implementation

NPIs are categorized and implemented in Bucky based on their classification into three categories:

Contact-Matrix Based NPIs

These NPIs are those that effect only certain age groups within the total population. These NPI effect the ratios relating the components of the contact matrices. The NPI that fall under this category are:

- School Closure
- Shielding Elderly

Mobility Based NPI

This classification is for those NPI that lead to changes in mobility/movement between administrative districts (as opposed to movement within an administrative district). The NPI that fall under this category are :

- Closing of borders, ports, and/or international flights
- Restricting inter-regional movement

Reproduction Number Based NPI

This classification is for those NPI that have an effect on the overall scaling of transmissibility. It encompasses both intra-regional measures to reduce transmission as well as national level initiatives designed to reduce transmission throughout the country. The NPI that fall under this category are :

- Social distancing
- Face mask wearing
- Installation of hand washing stations
- Reduction of size of public gatherings
- Closing businesses
- Partial Lockdown
- Awareness campaigns (e.g., vaccination programs)

A summary of the NPIs that are currently implemented in Bucky are given in the table below. This table includes the classification, effects, and sources that are currently being used to approximate the effects of various NPI.

With the current implementation, we have the ability to distinguish between the effects of NPI within the categories mentioned above. For the case in which multiple NPI within category III are implemented, we have implemented a value-added approach to calculating their effectiveness in reducing the basic reproduction number. In this case, we calculate the reduction in R_0 based on the number of NPIs in place. If one NPI is in place, R_0 is reduced by 40%. If two NPI are in place, R_0 is reduced by 60%. If three or more NPI are in place, then R_0 is reduced by 70%.

NPI Classification	Effect in Model	Mean Reduction (SD)	Source
Contact-based: School closure	Reduce contact between school aged groups and increase the contacts in the home environment	~44% reduction in overall community transmission	[WSM+20]
Mobility-based	Reduction in mobility between regions	60% (10)	[CAN+20] [WSM+20]
Reproduction number-based	60-8% reduction in overall community transmission	72.5% (6.25)	[JVZG+20] [JJA+20]

CHAPTER

FOUR

CLI INTERFACE

4.1 bucky.make_input_graph

Bucky Model input graph creation

```
usage: make_input_graph [-h] [-d DATE] [-o OUTPUT] [--hist_file HIST_FILE]
                        [-v] [--no_update]
```

4.1.1 Named Arguments

-d, --date Start date of simulation, last date for historical data.

Default: "2020-10-13"

-o, --output Directory for graph file. Defaults to data/input_graphs/

Default: "config.yml: <data_dir>/input_graphs/"

--hist_file File to use for historical data

Default: "config.yml: <data_dir>/cases/csse_hist_timeseries.csv"

-v, --verbose verbose output (repeat for increased verbosity; defaults to WARN, -v is INFO, -vv is DEBUG)

Default: 0

--no_update Skip updating data

Default: True

4.2 bucky.model

Bucky Model

```
usage: model [-h] [--graph GRAPH_FILE] [--n_mc N_MC] [--days DAYS] [-v] [-q]
             [-c] [-nmc] [-gpu] [-opt] [-r] [-o OUTPUT_DIR]
             [--npi_file [NPI_FILE]] [--disable-npi]
             [par_file]
```

4.2.1 Positional Arguments

par_file File containing paramters

Default: "config.yml: <base_dir>/par/scenario_5.yml"

4.2.2 Named Arguments

--graph, -g Pickle file containing the graph to run

Default: "Most recently created graph in <data_dir>/input_graphs"

--n_mc, -n Number of runs to do for Monte Carlo

Default: 100

--days, -d Length of the runs in days

Default: 40

-v, --verbose verbose output (repeat for increased verbosity; defaults to WARN, -v is INFO, -vv is DEBUG)

Default: 0

-q, --quiet quiet output (only show ERROR and higher)

Default: 0

-c, --cache Cache python files/par file/graph pickle for the run

Default: False

-nmc, --no_mc Just do one run with the mean param values

Default: False

-gpu, --gpu Use cupy instead of numpy

Default: False

-opt, --opt Enable cupy kernel optimizations. Do this for large runs using the gpu (n > 100).

Default: False

-r, --reject_runs Reject Monte Carlo runs with incidence rates that don't align with historical data

Default: False

-o, --output_dir Dir to put the output files

Default: "config.yml: <raw_output_dir>"

--npi_file File containing NPIs

--disable-npi Disable all active NPI from the npi_file at the start of the run

Default: False

4.3 bucky.postprocess

Bucky Model postprocessing

```
usage: postprocess [-h] [-g GRAPH_FILE] [-l LEVELS [LEVELS ...]]
                   [-q QUANTILES [QUANTILES ...]] [-o OUTPUT]
                   [--prefix PREFIX] [--end_date END_DATE] [--lookup LOOKUP]
                   [-n NPROCS] [-cpu] [--verify] [--no-sort] [-v]
                   [file]
```

4.3.1 Positional Arguments

file File to proess

Default: "Most recently created folder in raw_output_dir"

4.3.2 Named Arguments

-g, --graph_file Graph file used for simulation

-l, --levels Levels on which to aggregate

Default: ['adm0', 'adm1', 'adm2']

-q, --quantiles Quantiles to process

Default: [0.01, 0.025, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.975, 0.99]

-o, --output Directory for output files

Default: "config.yml: <output_dir>"

--prefix Prefix for output folder (default is UUID)

--end_date

--lookup Lookup table defining geoid relationships

-n, --nprocs Number of threads doing aggregations, more is better till you go OOM...

Default: 1

-cpu, --cpu Do not use cupy

Default: False

--verify Verify the quality of the data

Default: False

--no-sort, --no_sort Skip sorting the aggregated files

Default: False

-v, --verbose Print extra information

Default: False

4.4 bucky.viz.plot

Bucky model plotting tools

```
usage: viz.plot [-h] [-i INPUT_DIR] [-o OUTPUT] [-g GRAPH_FILE]
                [-l LEVELS [LEVELS ...]]
                [--plot_columns PLOT_COLUMNS [PLOT_COLUMNS ...]]
                [--lookup LOOKUP] [--min_hist MIN_HIST]
                [--hist_start HIST_START] [--adm1_name ADM1_NAME]
                [--end_date END_DATE] [-v] [-hist] [--hist_file HIST_FILE]
                [-q QUANTILES [QUANTILES ...]] [-w WINDOW_SIZE]
```

4.4.1 Named Arguments

-i, --input_dir	Directory location of aggregated data
	Default: "Most recently created folder in output_dir"
-o, --output	Output directory for plots. Defaults to input_dir/plots/
-g, --graph_file	Graph file used during model. Defaults to most recently created graph
-l, --levels	Requested plot levels
	Default: ['adm0', 'adm1']
--plot_columns	Columns to plot
	Default: ['daily_reported_cases', 'daily_deaths']
--lookup	Lookup table for geographic mapping info
--min_hist	Minimum number of historical data points to plot.
	Default: 0
--hist_start	Start date of historical data. If not passed in, will align with start date of simulation
--adm1_name	Admin1 to make admin2-level plots for
--end_date	Data will not be plotted past this point
-v, --verbose	Print extra information
	Default: False
-hist, --hist	Plot historical data in addition to simulation data
	Default: False
--hist_file	Path to historical data file. If None, uses either CSSE or Covid Tracking data depending on columns requested.
-q, --quantiles	Specify the quantiles to plot. Defaults to all quantiles present in data.
-w, --window_size	Size of window (in days) to apply to historical data
	Default: 7

4.5 bucky.viz.map

Bucky model mapping tools

```
usage: viz.map [-h] [-i INPUT_DIR] [-o OUTPUT] [-g GRAPH_FILE]
               [--columns COLUMNS [COLUMNS ...]] [--mean] [--linear]
               [-f {daily,weekly,monthly}] [-d DATES [DATES ...]] [--adm0]
               [--all_adm1] [--adm1 ADM1 [ADM1 ...]] [--adm1_shape ADM1_SHAPE]
               [--adm2_shape ADM2_SHAPE] [--adm1_col ADM1_COL]
               [--adm2_col ADM2_COL] [--lookup LOOKUP] [-c CMAP]
```

4.5.1 Named Arguments

-i, --input_dir Directory location of processed simulation data

Default: "Most recently created folder in output_dir"

-o, --output Output directory for maps. Defaults to input_dir/maps/

-g, --graph_file Graph file used during model

--columns Data columns to plot. Maps are created separately for each requested column

Default: ['daily_cases_reported', 'daily_deaths']

--mean Use mean value instead of median value for map

Default: False

--linear Use linear scaling for values instead of log

Default: False

-f, --freq Possible choices: daily, weekly, monthly

Frequency at which to create maps

Default: "weekly"

-d, --dates Specific dates to map

--adm0 Create adm0-level plot

Default: False

--all_adm1 Create adm1-level plot for every available adm1-level area

Default: False

--adm1 Create adm1-level plot for the requested adm1 name

--adm1_shape Location of admin1 shapefile

Default: "config.yml: <data_dir>/shapefiles/tl_2019_us_state.shp"

--adm2_shape Location of admin2 shapefile

Default: "config.yml: <data_dir>/shapefiles/tl_2019_us_county.shp"

--adm1_col Shapefile adm1 column name

Default: "STATEFP"

--adm2_col Shapefile adm2 column name

Default: "GEOID"

--lookup	Lookup table for geographic mapping info
-c, --cmap	Colormap to use. Must be a valid matplotlib colormap. Default: “Reds”

CHAPTER

FIVE

BUCKY MODULES

Note: Docstrings are still being added, expect some to be missing or incomplete.

5.1 bucky package

5.1.1 Subpackages

bucky.util package

Submodules

bucky.util.distributions module

Provides any probability distributions used by the model that aren't in numpy/cupy.

bucky.util.distributions.**mPERT_sample**(*mu*, *a=0.0*, *b=1.0*, *gamma=4.0*, *var=None*)

Provides a vectorized Modified PERT distribution.

Parameters

- **mu** (*float, array_like*) – Mean value for the PERT distribution.
- **a** (*float, array_like*) – Lower bound for the distribution.
- **b** (*float, array_like*) – Upper bound for the distribution.
- **gamma** (*float, array_like*) – Shape paramter.
- **var** (*float, array_like, None*) – Variance of the distribution. If var != None, gamma will be calcuated to meet the desired variance.

Returns **out** – Samples drawn from the specified mPERT distribution. Shape is the broadcasted shape of the the input parameters.

Return type float, array_like

bucky.util.distributions.**truncnorm**(*xp*, *loc=0.0*, *scale=1.0*, *size=1*, *a_min=None*, *a_max=None*)

Provides a vectorized truncnorm implementation that is compatible with cupy.

The output is calculated by using the numpy/cupy random.normal() and truncted via rejection sampling. The interface is intended to mirror the scipy implementation of truncnorm.

Parameters **xp** (*module*) –

bucky.util.get_historical_data module

bucky.util.get_historical_data.**add_daily_history**(*history_data*, *window_size=None*)
Applies a window to cumulative historical data to get daily data.

Parameters

- **history_data** (*Pandas DataFrame*) – Cumulative case and death data
- **window_size** (*int or None*) – Size of window in days

Returns **history_data** – Historical data with added columns for daily case and death data

Return type Pandas DataFrame

bucky.util.get_historical_data.**get_historical_data**(*columns*, *level*, *lookup_df*, *window_size*, *hist_file*)
Gets historical data for the columns requested.

Parameters

- **columns** (*list of str*) – Column names for historical data
- **level** (*str*) – Geographic level to get historical data for, e.g. adm1
- **lookup_df** (*Pandas DataFrame*) – Dataframe with names and values for admin0, admin1, and admin2 levels
- **window_size** (*int*) – Size of window in days
- **hist_file** (*string or None*) – Historical data file to use if not using defaults.

Returns **history_data** – Historical data indexed by data and geographic level containing only requested columns

Return type Pandas DataFrame

bucky.util.graph2histcsv module

bucky.util.read_config module

bucky.util.readable_col_names module

bucky.util.scoring module

bucky.util.scoring.**IS**(*x*, *lower*, *upper*, *alp*)
:param : TODO :param : TODO

Returns TODO

bucky.util.scoring.**WIS**(*x*, *q*, *x_q*, *norm=False*, *log=False*, *smooth=False*)
:param : TODO :param : TODO

Returns TODO

bucky.util.scoring.**logistic**(*x*, *x0=0.0*, *k=1.0*, *L=1.0*)
:param : TODO :param : TODO

Returns TODO

bucky.util.scoring.**smooth_IS**(*x*, *lower*, *upper*, *alp*)
:param : TODO :param : TODO

Returns TODO

bucky.util.update_data_repos module

Data Updating Utility (bucky.util.update_data_repos)

A utility for fetching updated data for mobility and case data from public repositories.

This module pulls from public git repositories and preprocessed the data if necessary. For case data, unallocated or unassigned cases are distributed as necessary.

bucky.util.update_data_repos.**distribute_data_by_population**(*total_df*, *dist_vect*, *data_to_dist*, *replace*)
Distributes data by population across a state or territory.

Parameters

- **total_df** (*Pandas DataFrame*) – DataFrame containing confirmed and death data indexed by date and FIPS code
- **dist_vect** (*Pandas DataFrame*) – Population data for each county as proportion of total state population, indexed by FIPS code
- **data_to_dist** (*Pandas DataFrame*) – Data to distribute, indexed by data
- **replace** (*boolean*) – If true, distributed values overwrite current historical data in DataFrame. If false, distributed values are added to current data

Returns **total_df** – Modified input dataframe with distributed data

Return type Pandas DataFrame

bucky.util.update_data_repos.**distribute_mdoc**(*df*, *csse_deaths_file*)
Distributes Michigan Department of Corrections data across Michigan counties by population.

Parameters

- **df** (*Pandas DataFrame*) – Current historical DataFrame indexed by FIPS and date, which includes MDOC and FCI data
- **csse_deaths_file** (*string*) – File location of CSSE deaths file (contains population data)

Returns **df** – Modified historical dataframe with Michigan prison data distributed and added to Michigan data

Return type Pandas DataFrame

bucky.util.update_data_repos.**distribute_nyc_data**(*df*)
Distributes NYC case data across the six NYC counties.

Parameters

- **df** (*Pandas DataFrame*) – DataFrame containing historical data indexed by FIPS and date
- **add deprecation warning b/c csse has fixed this** (*TODO*) –

Returns **df** – Modified DataFrame containing corrected NYC historical data indexed by FIPS and date

Return type Pandas DataFrame

bucky.util.update_data_repos.**distribute_territory_data**(*df*, *add_american_samoa*)

Distributes territory-wide case and death data for territories.

Uses county population to distribute cases for US Virgin Islands, Guam, and CNMI. Optionally adds a single case to the most populous American Samoan county.

Parameters

- **df** (*Pandas DataFrame*) – Current historical DataFrame indexed by FIPS and date, which includes territory-wide case and death data
- **add_american_samoa** (*boolean*) – If true, adds 1 case to American Samoa

Returns **df** – Modified historical dataframe with territory-wide data distributed to counties

Return type Pandas DataFrame

bucky.util.update_data_repos.**distribute_unallocated_csse**(*confirmed_file*, *deaths_file*, *hist_df*)

Distributes unallocated historical case and deaths data from CSSE.

JHU CSSE data contains state-level unallocated data, indicated with "Unassigned" or "Out of" for each state. This function distributes these unallocated cases based on the proportion of cases in each county relative to the state.

Parameters

- **confirmed_file** (*string*) – filename of CSSE confirmed data
- **deaths_file** (*string*) – filename of CSSE death data
- **hist_df** (*Pandas DataFrame*) – current historical DataFrame containing confirmed and death data indexed by date and FIPS code

Returns **hist_df** – modified historical DataFrame with cases and deaths distributed

Return type Pandas DataFrame

bucky.util.update_data_repos.**get_county_population_data**(*csse_deaths_file*, *county_fips*)

Uses JHU CSSE deaths file to get county population data as as fraction of population across list of counties.

Parameters

- **csse_deaths_file** (*string*) – filename of CSSE deaths file
- **county_fips** (*array-like*) – list of FIPS to return population data for

Returns **population_df** – DataFrame with population fraction data indexed by FIPS

Return type Pandas DataFrame

bucky.util.update_data_repos.**get_timeseries_data**(*col_name*, *filename*, *fips_key='FIPS'*, *is_csse=True*)

Transforms a historical data file to a dataframe with FIPs, date, and case or death data.

Parameters

- **col_name** (*string*) – Column name to extract from data.
- **filename** (*string*) – Location of filename to read.
- **fips_key** (*string, optional*) – Key used in file for indicating county-level field.
- **is_csse** (*boolean*) – Indicates whether the file is CSSE data. If True, certain areas without FIPS are included.

Returns **df** – Dataframe with the historical data indexed by FIPS, date

Return type Pandas DataFrame

bucky.util.update_data_repos.**git_pull**(*abs_path*)
Updates a git repository given its path.

Parameters **abs_path** (*string*) – Abs path location of repository to update

bucky.util.update_data_repos.**process_csse_data**()
Performs pre-processing on CSSE data.

CSSE data is separated into two different files: confirmed cases and deaths. These two files are combined into one dataframe, indexed by FIPS and date with two columns, Confirmed and Deaths. This function distributes CSSE that is either unallocated or territory-wide instead of county-wide. Michigan data from the state Department of Corrections and Federal Correctional Institution is distributed to Michigan counties. New York City data which is currently all placed in one county (New York County) is distributed to the other NYC counties. Territory data for Guam, CNMI, and US Virgin Islands is also distributed. This data is written to a CSV.

bucky.util.update_data_repos.**process_usafacts**(*case_file*, *deaths_file*)
Performs preprocessing on USA Facts data.

USAFacts contains unallocated cases and deaths for each state. These are allocated across states based on case distribution in the state.

Parameters

- **case_file** (*string*) – Location of USAFacts case file
- **deaths_file** (*string*) – Location of USAFacts death file

Returns **combined_df** – USAFacts data containing cases and deaths indexed by FIPS and date.

Return type Pandas DataFrame

bucky.util.update_data_repos.**update_covid_tracking_data**()
Downloads and processes data from the COVID Tracking project to match the format of other preprocessed data.

The COVID Tracking project contains data at a state-level. Each state is given a random FIPS selected from all FIPS in that state. This is done to make aggregation easier for plotting later. Processed data is written to a CSV.

bucky.util.update_data_repos.**update_repos**()
Uses git to update public data repos.

bucky.util.update_data_repos.**update_usafacts_data**()
Retrieves updated historical data from USA Facts, preprocesses it, and writes to CSV.

bucky.util.util module

class bucky.util.util.**TqdmLoggingHandler**(*level=0*)
Bases: logging.Handler

emit(*record*)
Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

bucky.util.util.**bin_age_csv**(*filename*, *out_filename*)

bucky.util.util.**cache_files**(**argv*)

bucky.util.util.**date_to_t_int**(*dates*, *start_date*)

class bucky.util.util.**dotdict**
Bases: dict

dot.notation access to dictionary attributes.

bucky.util.util.**estimate_IFR**(*age*)

bucky.util.util.**map_np_array**(*a*, *d*)

bucky.util.util.**remove_chars**(*seq*)

bucky.util.util.**unpack_cache**(*cache_file*)

Module contents

bucky.viz package

Submodules

bucky.viz.geoid module

bucky.viz.geoid.**read_geoid_from_graph**(*graph_file=None*)
Creates a dataframe relating geographic administration levels, e.g. admin2 values in a given admin1.

Parameters **graph_file** (*string or None*) – Location of graph file. If None, uses most recently created graph in data/input_graphs/

Returns **df** – Dataframe with names and values for admin0, admin1, and admin2 levels

Return type Pandas DataFrame

bucky.viz.geoid.**read_lookup**(*geofile*, *country='US'*)
Creates a dataframe relating geographic administration levels e.g. admin2 values in a given admin1 based on a lookup table.

Parameters

- **geofile** (*string*) – Location of lookup table
- **country** (*string (default: 'US')*) – Country name

Returns **df** – Dataframe with names and values for admin0, admin1, and admin2

Return type Pandas DataFrame

bucky.viz.map module

bucky.viz.map.**get_dates**(*df*, *frequency='weekly'*)
Given a DataFrame of simulation data, this method returns dates based on the requested frequency.

Parameters

- **df** (*Pandas DataFrame*) – Dataframe of simulation data
- **frequency** (*{'daily', 'monthly', 'weekly' (default)}*) – Frequency of selected dates

Returns **date_list** – List of dates

Return type list of strings

bucky.viz.map.**get_map_data**(*data_dir*, *adm_level*, *use_mean=False*)

Reads requested simulation data.

Maps are created using one level down from the requested map level. For example, a national map is created using state-level data.

Parameters

- **data_dir** (*string*) – Location of preprocessed simulation data
- **adm_level** (*{'adm0', adm1'}*) – Admin level of requested map
- **use_mean** (*boolean*) – If true, uses mean data. Otherwise, uses median quantile

Returns **df** – Requested preprocessed simulation data

Return type Pandas DataFrame

bucky.viz.map.**get_state_outline**(*adm2_data*, *adm1_data*)

Given admin2 shape data, finds matching admin1 shape data in order to get the admin1 outline.

Parameters

- **adm2_data** (*Geopandas GeoDataFrame*) – Admin2-level shape data
- **adm1_data** (*Geopandas GeoDataFrame*) – Admin1-level shape data

Returns **outline_df** – Admin1-level shape data that match values in admin2

Return type Geopandas GeoDataFrame

bucky.viz.map.**make_adm1_maps**(*adm2_shape_df*, *adm1_shape_df*, *df*, *lookup_df*, *dates*, *cols*, *adm1_list*, *output_dir*, *log_scale=True*, *colormap='Reds'*, *add_outline=False*)

Creates adm1 maps.

Parameters

- **adm2_shape_df** (*Geopandas GeoDataFrame*) – Shapefile information at the admin2 level
- **adm1_shape_df** (*Geopandas GeoDataFrame*) – Shapefile information at the admin1 level
- **df** (*Pandas DataFrame*) – Simulation data to plot
- **lookup_df** (*Pandas DataFrame*) – Dataframe containing mapping between admin levels
- **dates** (*list of strings*) – List of dates to make maps for
- **cols** (*list of strings*) – List of columns to make maps for
- **adm1_list** (*list of strings or None*) – List of explicit admin1 names to create names for. If None, a map is made for each unique admin1 in the lookup table
- **output_dir** (*string*) – Directory to place created maps
- **log_scale** (*boolean (default: True)*) – If true, uses log scaling
- **colormap** (*string, (default: 'Reds')*) – Colormap to use; must be a valid Matplotlib colormap
- **add_outline** (*boolean (default: False)*) – Add a thicker outline to the map

bucky.viz.map.**make_map**(*shape_df*, *df*, *dates*, *adm_key*, *cols*, *output_dir*, *title_prefix=None*, *log_scale=True*, *colormap='Reds'*, *outline_df=None*)

Creates a map for each date and column.

Parameters

- **shape_df** (*Geopandas GeoDataFrame*) – Shapefile information at the required admin level
- **df** (*Pandas DataFrame*) – Simulation data to plot
- **dates** (*list of strings*) – List of dates to make maps for
- **cols** (*list of strings*) – List of columns to make maps for
- **output_dir** (*string*) – Directory to place created maps
- **title_prefix** (*string or None*) – String to add to map prefix
- **log_scale** (*boolean*) – If true, uses log scaling
- **colormap** (*string (default: 'Reds')*) – Colormap to use; must be a valid Matplotlib colormap
- **outline_df** (*Geopandas GeoDataFrame or None*) – Shapefile for outline

bucky.viz.plot module

Bucky Plotting Tools (bucky.viz.plot)

TODO

bucky.viz.plot.**interval**(*mean*, *sem*, *conf*, *N*)

bucky.viz.plot.**make_plots**(*adm_levels*, *input_dir*, *output_dir*, *lookup*, *plot_hist*, *plot_columns*, *quantiles*, *window_size*, *end_date*, *hist_file*, *min_hist_points*, *admin1=None*, *hist_start=None*)

Wrapper function around plot. Creates plots, aggregating data if necessary.

Parameters

- **adm_levels** (*list of strings*) – List of ADM levels to make plots for
- **input_dir** (*string*) – Location of simulation data
- **output_dir** (*string*) – Parent directory to place created plots.
- **lookup** (*Pandas DataFrame*) – Lookup table for geographic mapping information
- **plot_hist** (*boolean*) – If true, will plot historical data
- **plot_columns** (*list of strings*) – List of columns to plot from data
- **quantiles** (*list of floats (or None)*) – List of quantiles to plot. If None, will plot all available quantiles in data.
- **window_size** (*int*) – Size of window (in days) to apply to historical data
- **end_date** (*string, formatted as YYYY-MM-DD*) – Plot data until this date. If None, uses last date in simulation
- **hist_file** (*string*) – Path to historical data file. If None, uses either CSSE or Covid Tracking data depending on columns requested.
- **min_hist_points** (*int*) – Minimum number of historical data points to plot.
- **admin1** (*list of strings, or None*) – List of admin1 values to make plots for. If None, a plot will be created for every unique admin1 values. Otherwise, plots are only made for those requested.

- **hist_start** (*string, formatted as YYYY-MM-DD*) – Plot historical data from this point. If None, aligns with simulation start date

bucky.viz.plot.**plot**(*output_dir*, *lookup_df*, *key*, *sim_data*, *hist_data*, *plot_columns*, *quantiles*)

Given a dataframe and a key, creates plots with requested columns.

For example, a DataFrame with state-level data would create a plot for each unique state. Simulation data is plotted as a line with shaded confidence intervals. Historical data is added as scatter points if requested.

Parameters

- **output_dir** (*string*) – Location to place created plots.
- **lookup_df** (*Pandas DataFrame*) – Dataframe containing information relating different geographic areas
- **key** (*string*) – Key to use to relate simulation data and geographic areas. Must appear in lookup and simulation data (and historical data if applicable)
- **sim_data** (*Pandas DataFrame*) – Simulation data to plot
- **hist_data** (*Pandas DataFrame*) – Historical data to add to plot
- **plot_columns** (*list of strings*) – Columns to plot
- **quantiles** (*list of floats (or None)*) – List of quantiles to plot. If None, will plot all available quantiles in data.

Module contents

5.1.2 Submodules

5.1.3 bucky.arg_parser_model module

arg parser for bucky.model

This module handles all the CLI argument parsing for bucky.model and autodetects CuPy.

5.1.4 bucky.make_input_graph module

bucky.make_input_graph.**compute_population_density**(*age_df*, *shape_df*)

Computes normalized population density.

Parameters

- **age_df** (*Pandas DataFrame*) – age-stratified population data
- **shape_df** (*Geopandas GeoDataFrame*) – GeoDataFrame with shape information indexed by FIPS

Returns **popdens** – DataFrame with population density by FIPS

Return type Pandas DataFrame

bucky.make_input_graph.**get_case_history**(*historical_data*, *end_date*, *num_days=45*)

Gets case and death history for the requested number of days for each FIPS.

If data is missing for a date, it is replaced with the data from the last valid date.

Parameters

- **historical_data** (*Pandas DataFrame*) – Dataframe with case, death data indexed by date, FIPS
- **end_date** (*date string*) – Last date to get data for
- **num_days** (*int*) – Number of days of history requested

Returns **hist** – Dictionary of case data, keyed by FIPS

Return type dict

bucky.make_input_graph.**get_lex**(*last_date*, *window_size=7*)
Reads county-level location exposure indices from PlaceIQ location data and applies a window.

Parameters

- **last_date** (*last_date*) – Fetches data for requested date
- **window_size** (*int (default: 7)*) – Size of window, in days, to apply to data

Returns **frac_df** – TODO

Return type Pandas DataFrame

bucky.make_input_graph.**get_mobility_data**(*popdens*, *end_date*, *age_data*, *add_territories=True*)
Fetches mobility data.

Parameters

- **popdens** (*Pandas DataFrame*) – Population density indexed by FIPS
- **end_date** (*string*) – Last date of historical data
- **age_data** (*Pandas DataFrame*) – County-level age-stratified population data
- **add_territories** (*boolean*) – Adds territory data if True

Returns

- **mean_edge_weights** (*Pandas DataFrame*) – TODO
- **move_dict** (*dict*) – TODO

bucky.make_input_graph.**get_safegraph**(*last_date*, *window_size=7*)
Reads SafeGraph mobility data and applies a window.

Parameters

- **last_date** (*last_date*) – Fetches data for requested date
- **window_size** (*int (default: 7)*) – Size of window, in days, to apply to data

Returns **frac_df** – TODO

Return type Pandas DataFrame

bucky.make_input_graph.**read_descartes_data**(*end_date*)
Reads Descartes mobility data [WS20].

Parameters **end_date** (*string*) – Last date to get Descartes data

Returns

- **nat_frac_move** (*Pandas DataFrame*) – TODO
- **dl_state** (*Pandas DataFrame*) – TODO
- **dl_county** (*Pandas DataFrame*) – TODO

Notes

Data provided by Descartes Labs (https://descarteslabs.com/mobility/)[1]

bucky.make_input_graph.**read_lex_data**(*date*)

Reads county-level location exposure indices for a given date from PlaceIQ location data.

In order to improve performance, preprocessed data is saved. If the user requests data for a date that has already been preprocessed, it will read the data from disk instead of repeating the processing.

Parameters **date** (*string*) – Fetches data for requested date

Returns **df_long** – Preprocessed LEX data

Return type Pandas DataFrame

5.1.5 bucky.model module

class bucky.model.**SEIR_covid**(*seed=None*, *randomize_params_on_reset=True*)

Bases: object

static RHS_func(*t*, *y*, *Nij*, *contact_mats*, *Aij*, *par*, *npi*)

estimate_doubling_time(*days_back=7*, *doubling_time_window=7*, *mean_time_window=None*, *min_doubling_t=1.0*)

estimate_doubling_time_WHO(*days_back=14*, *doubling_time_window=7*, *mean_time_window=None*, *min_doubling_t=1.0*)

estimate_reporting(*cfr*, *days_back=14*, *case_lag=None*, *min_deaths=100.0*)

get_state_indices()

reset(*seed=None*, *params=None*)

reset_A(*var*)

run_once(*seed=None*, *outdir='raw_output/'*, *output=True*, *output_queue=None*)

exception bucky.model.**SimulationException**

Bases: Exception

bucky.model.**get_runid**(*pid=0*)

5.1.6 bucky.npi module

bucky.npi.**read_npi_file**(*fname*, *start_date*, *end_t*, *adm2_map*, *disable_npi=False*)

TODO Description.

Parameters

- **fname** (*string*) – Filename of NPI file
- **start_date** (*string*) – Start date to use
- **end_t** (*int*) – Number of days after start date
- **adm2_map** (*NumPy array*) – Array of adm2 IDs
- **disable_npi** (*bool (default: False)*) – Bool indicating whether NPIs should be disabled

[1] Warren, Michael S. & Skillman, Samuel W. "Mobility Changes in Response to COVID-19". arXiv:2003.14228 [cs.SI], Mar. 2020. arxiv.org/abs/2003.14228

Returns **npi_params** – TODO

Return type dict

5.1.7 bucky.numerical_libs module

Provides an interface to import numerical libraries using the GPU (if available).

bucky.numerical_libs.**use_cupy**(*optimize=False*)

Perform imports for libraries with APIs matching numpy, scipy.integrate.ivp, scipy.sparse.

These imports will use a monkey-patched version of these modules that has had all it's numpy references replaced with CuPy.

if optimize is True, place the kernel optimization context in xp.optimize_kernels, otherwise make it a nullcontext (noop)

returns nothing but imports a version of 'xp', 'ivp', and 'sparse' to the global scope of this module

Parameters **optimize** (*bool*) – Enable kernel optimization in cupy >=v8.0.0. This will slow down initial function call (mostly reduction operations) but will offer better performance for repeated calls (e.g. in the RHS call of an integrator).

Returns **exit_code** – Non-zero value indicates error code, or zero on success.

Return type int

Raises **NotImplementedError** – If the user calls a monkeypatched function of the libs that isn't fully implemented.

5.1.8 bucky.parameters module

bucky.parameters.**CI_to_std**(*CI*)

class bucky.parameters.**buckyParams**(*par_file=None*, *gpu=False*)

Bases: object

static age_interp(*x_bins_new*, *x_bins*, *y*)

static calc_derived_params(*params*)

generate_params(*var=0.2*)

static read_yml(*par_file*)

reroll_params(*base_params*, *var*)

static rescale_doubling_rate(*D*, *params*, *xp*, *A=None*)

bucky.parameters.**calc_Reff**(*m*, *n*, *Tg*, *Te*, *r*)

bucky.parameters.**calc_Te**(*Tg*, *Ts*, *n*, *f*)

bucky.parameters.**calc_Ti**(*Te*, *Tg*, *n*)

bucky.parameters.**calc_beta**(*Te*)

bucky.parameters.**calc_gamma**(*Ti*)

5.1.9 bucky.postprocess module

bucky.postprocess.**divide_by_pop**(*dataframe*, *cols*)

Given a dataframe and list of columns, divides the columns by the population column ('N').

Parameters

- **dataframe** (*Pandas DataFrame*) – Simulation data
- **cols** (*list of strings*) – Column names to scale by population

Returns **dataframe** – Original dataframe with the requested columns scaled

Return type Pandas DataFrame

5.1.10 Module contents

CHAPTER

SIX

REFERENCES

- "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University" <https://github.com/CSSEGISandData/COVID-19>
- Dong E, Du H, Gardner L. "An interactive web-based dashboard to track COVID-19 in real time". Lancet Inf Dis. 20(5):533-534. <https://doi.org/10.1016/S1473-3099(20)30120-1>
- Warren, Michael S. & Skillman, Samuel W. "Mobility Changes in Response to COVID-19". arXiv:2003.14228 [cs.SI], Mar. 2020. <https://arxiv.org/abs/2003.14228>
- "Measuring movement and social contact with smartphone data: a real-time application to COVID-19" by Couture, Dingel, Green, Handbury, and Williams <https://github.com/COVIDExposureIndices/COVIDExposureIndices/blob/master/CDGHW.pdf>
- Prem K, Cook AR, Jit M (2017) Projecting social contact matrices in 152 countries using contact surveys and demographic data. PLoS Comput Biol 13(9): e1005697. <https://doi.org/10.1371/journal.pcbi.1005697>

#.. image:: ../../logo.png

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[BCB+20] Oyungerel Byambasuren, Magnolia Cardona, Katy Bell, Justin Clark, Mary-Louise McLaws, and Paul Glasziou. Estimating the extent of true asymptomatic covid-19 and its potential for community transmission: systematic review and meta-analysis. *Available at SSRN 3586675*, 2020.

[CGM+20] Miriam Casey, John Griffin, Conor G McAloon, Andrew W Byrne, Jamie M Madden, David McEvoy, Aine B Collins, Kevin Hunt, Ann Barber, Francis Butler, and others. Estimating pre-symptomatic transmission of covid-19: a secondary analysis using published data. *medRxiv*, 2020.

[CDC] CDC. The coronavirus disease 2019 (covid-19)-associated hospitalization surveillance network (covid-net).

[CAN+20] Benjamin J. Cowling, Sheikh Taslim Ali, Tiffany W. Y. Ng, Tim K. Tsang, Julian C. M. Li, Min Whui Fong, Qiuyan Liao, Mike YW Kwan, So Lun Lee, Susan S. Chiu, Joseph T. Wu, Peng Wu, and Gabriel M. Leung. Impact assessment of non-pharmaceutical interventions against coronavirus disease 2019 and influenza in hong kong: an observational study. *The Lancet Public Health*, 5(5):e279–e288, May 2020. URL: https://doi.org/10.1016/S2468-2667(20)30090-6, doi:10.1016/S2468-2667(20)30090-6.

[fDCP+20] Centers for Disease Control, Prevention, and others. Covid-19 pandemic planning scenarios. *URL: https://www. cdc. gov/coronavirus/2019-ncov/hcp/planning-scenarios. html Accessed May*, 2020.

[HLW+20] Xi He, Eric HY Lau, Peng Wu, Xilong Deng, Jian Wang, Xinxin Hao, Yiu Chung Lau, Jessica Y Wong, Yujuan Guan, Xinghua Tan, and others. Temporal dynamics in viral shedding and transmissibility of covid-19. *Nature medicine*, 26(5):672–675, 2020.

[Het89] Herbert W. Hethcote. *Three Basic Epidemiological Models*, pages 119–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. URL: https://doi.org/10.1007/978-3-642-61317-3_5, doi:10.1007/978-3-642-61317-3_5.

[JVZG+20] Christopher I Jarvis, Kevin Van Zandvoort, Amy Gimma, Kiesha Prem, Petra Klepac, G James Rubin, and W John Edmunds. Quantifying the impact of physical distance measures on the transmission of covid-19 in the uk. *BMC medicine*, 18:1–10, 2020.

[JJA+20] Lemaitre C Joseph, Perez-Saez Javier, Azman S Andrew, Rinaldo Andrea, and Fellay Jacques. Assessing the impact of non-pharmaceutical interventions on sars-cov-2 transmission in switzerland. *Swiss Medical Weekly*, 150(ARTICLE):w20295, 2020.

[LKL+20] Seungjae Lee, Tark Kim, Eunjung Lee, Cheolgu Lee, Hojung Kim, Heejeong Rhee, Se Yoon Park, Hyo-Ju Son, Shinae Yu, Jung Wan Park, and others. Clinical course and molecular viral shedding among asymptomatic and symptomatic patients with sars-cov-2 infection in a community treatment center in the republic of korea. *JAMA internal medicine*, 2020.

[LYW+20] Yang Liu, Li-Meng Yan, Lagen Wan, Tian-Xin Xiang, Aiping Le, Jia-Ming Liu, Malik Peiris, Leo LM Poon, and Wei Zhang. Viral dynamics in mild and severe cases of covid-19. *The Lancet Infectious Diseases*, 2020.

[MCH+20] Conor McAloon, Áine Collins, Kevin Hunt, Ann Barber, Andrew W Byrne, Francis Butler, Miriam Casey, John Griffin, Elizabeth Lane, David McEvoy, and others. Incubation period of covid-19: a rapid systematic review and meta-analysis of observational research. *BMJ open*, 10(8):e039652, 2020.

[NYS+20] Ji Yun Noh, Jin Gu Yoon, Hye Seong, Won Suk Choi, Jang Wook Sohn, Hee Jin Cheong, Woo Joo Kim, and Joon Young Song. Asymptomatic infection and atypical manifestations of covid-19: comparison of viral shedding duration. *The Journal of Infection*, 2020.

[OT20] Daniel P Oran and Eric J Topol. Prevalence of asymptomatic sars-cov-2 infection: a narrative review. *Annals of Internal Medicine*, 2020.

[PTC+20] Piero Poletti, Marcello Tirani, Danilo Cereda, Filippo Trentini, Giorgio Guzzetta, Giuliana Sabatino, Valentina Marziano, Ambra Castrofino, Francesca Grosso, Gabriele Del Castillo, and others. Probability of symptoms and critical disease after sars-cov-2 infection. *arXiv preprint arXiv:2006.08471*, 2020.

[WS20] Michael S Warren and Samuel W Skillman. Mobility changes in response to covid-19. *arXiv preprint arXiv:2003.14228*, 2020.

[WSM+20] Chad R. Wells, Pratha Sah, Seyed M. Moghadas, Abhishek Pandey, Affan Shoukat, Yaning Wang, Zheng Wang, Lauren A. Meyers, Burton H. Singer, and Alison P. Galvani. Impact of international travel and border control measures on the global spread of the novel 2019 coronavirus outbreak. *Proceedings of the National Academy of Sciences*, 117(13):7504–7509, 2020. URL: https://www.pnas.org/content/117/13/7504, arXiv:https://www.pnas.org/content/117/13/7504.full.pdf, doi:10.1073/pnas.2002616117.

PYTHON MODULE INDEX

b

INDEX

G

I

L

M

P

R

S

T

U

W