Bucky

Release .1

Matt Kinsey, Kate Tallaksen, Freddy Obrecht

Oct 25, 2020

CONTENTS:

CHAPTER

ONE

GETTING STARTED

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

COMPARTMENT MODEL SPEC

3.1 Diagram

3.2 Evolution Equations

Where the compartments E, I and R^H are gamma-distributed with $k = 2$

Force of infection for age group i at location j is

$$\lambda_{ij} = \beta_{ikjl} I^{kl}$$
$$\beta_{ikjl} = \beta \widetilde{C}_{ik} \widetilde{M}_{jl}$$
$$\widetilde{C}_{ij} = \frac{C_{ij}}{\sum_k C_{ik}}$$
$$\widetilde{M}_{ij} = \frac{M_{ij}}{\sum_k M_{ik}}$$

TODO mention calculation of ifr from cfr, case report and asym

TODO chr is being applied to asym? wouldnt that make it IHR?

TODO this is missing normalization related stuff

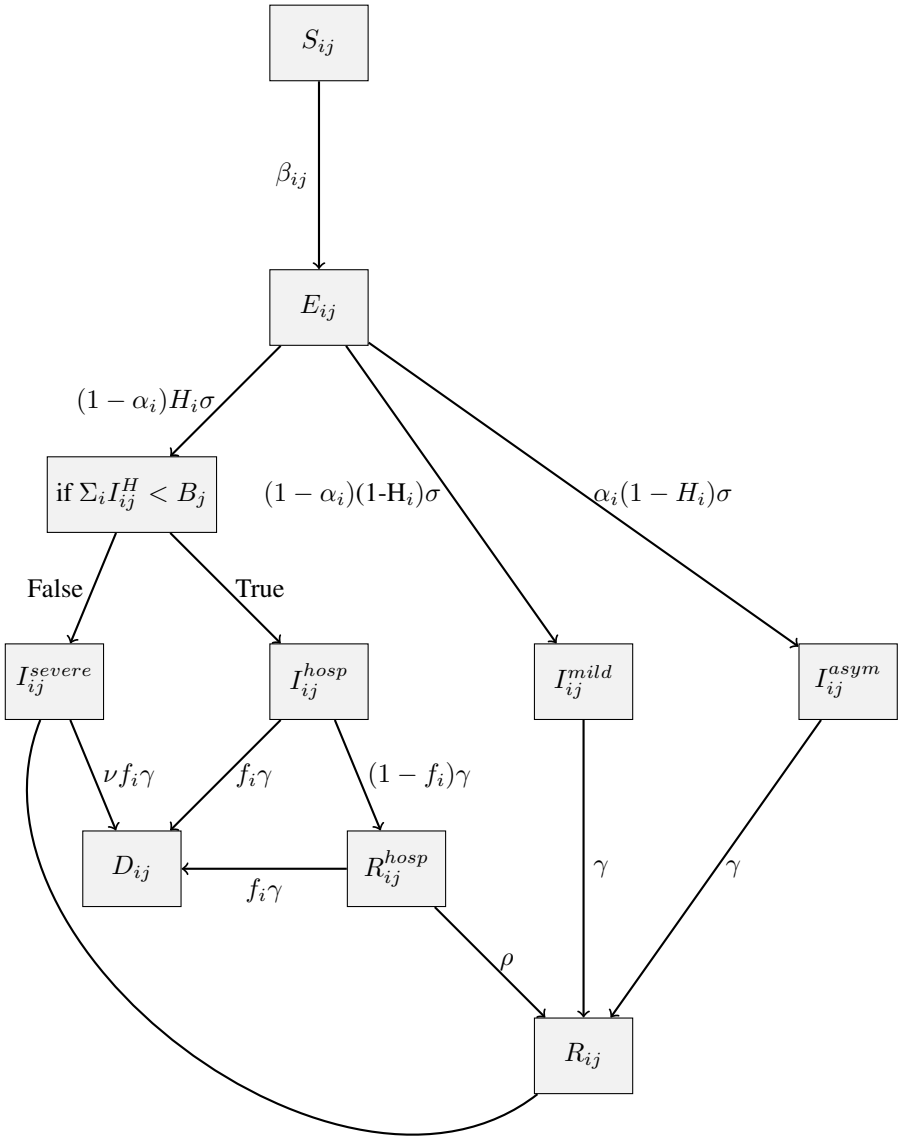
S_{ij}
β_{ij}
E_{ij}
$(1-\alpha_i)H_i\sigma$
$(1-\alpha_i)$(1-H$_i$)σ
$\alpha_i(1-H_i)\sigma$
if $\Sigma_i I_{ij}^H < B_j$
False
True
I_{ij}^{severe}
I_{ij}^{hosp}
I_{ij}^{mild}
I_{ij}^{asym}
$\nu f_i\gamma$
$f_i\gamma$
$(1-f_i)\gamma$
D_{ij}
R_{ij}^{hosp}
$f_i\gamma$
γ
γ
ρ
R_{ij}

Fig. 1: Model

$$\frac{dS_{ij}}{dt} = -\frac{\lambda_{ij} S_{ij}}{N_{ij}}$$
$$\frac{dE_{ij}}{dt} = \frac{\lambda_{ij} S_{ij}}{N_{ij}} - \sigma E_{ij}$$
$$\frac{dI_{ij}^{\text{asym}}}{dt} = \alpha_i (1 - \eta_i) \sigma E_{ij} - \gamma I_{ij}^{\text{asym}}$$
$$\frac{dI_{ij}^{\text{mild}}}{dt} = (1 - \alpha_i)(1 - \eta_i) \sigma E_{ij} - \gamma I_{ij}^{\text{mild}}$$

$$\frac{dI_{ij}^{\text{hosp}}}{dt} = \eta_i \sigma E_{ij} - \gamma I_{ij}^{\text{hosp}}$$
$$\frac{dR_{ij}^{\text{hosp}}}{dt} = (1 - \phi_i) \gamma I_{ij}^{\text{hosp}} - \rho R_{ij}^{\text{hosp}}$$
$$\frac{dD_{ij}}{dt} = \phi_i \gamma I_{ij}^{\text{hosp}}$$

$$\frac{dR_{ij}}{dt} = \rho R_{ij}^{\text{hosp}} + \gamma (I_{ij}^{\text{asym}} + I_{ij}^{\text{mild}})$$

CHAPTER

FOUR

MODEL INPUT AND OUTPUT

4.1 Input

The Bucky model uses two main sources of input: the input graph and CDC-recommended parameters. The input graph contains data regarding demographics, geographic information, and historical case information.

4.1.1 CDC-Recommended Parameters

The Centers for Disease Control and Prevention (CDC) has published pandemic planning scenarios [7] which contain recommended parameters describing biological and epidemiological parameters. Of these five planning scenarios, OCHA-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 Table 2.

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

4.2 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 levevel, with each row indexed by data, admin ID, and quantile. The columns of this output file are described in the tables below.

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

CHAPTER

FIVE

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).

NOTE: This graph should be able to be constructed on the province or state level, but may require model changes. If this is the case, talk to Matt Kinsey.

5.1 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'`.

5.2 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.        ,
         25.        ]),
  'adm2': 1001.0})
```

5.3 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*.

5.4 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.

5.5 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).

5.6 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.

CHAPTER

SIX

USAGE

6.1 Creating Input Graphs

`make_input_graph.py` creates input graphs used by the model and by certain postprocessing steps. This script has one required command line argument: `--date`. This field specifies the **last** date of historical data to use. This is also the **start** date of the simulation.

Arguments and flags:

- `-d`, `--date` (**Required**): Start date of simulaiton. Format: YYYY-MM-DD
- `-o`, `--output` : Output directory for created graph file. *Default*: `data/input_graphs/`
- `--hist_file` : Specify historical case file. *Default:* Uses CSSE data. **Note**: Must be county-level
- `--no_update` : Skips updating data repositories

6.2 Model

`model.py` takes the following arguments (all are optional as they are either flags or have defined defaults):

- `--graph`, `-g`: Graph file to use. *Default*: Most recently created graph.
- par_file (Positional): Parameter file to use. *Default*: `par/scenario_5.yml`
- `--n_mc`, `-n`: Number of Monte Carlo runs to perform. *Default*: 1000
- `--days`, `-d`: Length of simulation in days. *Default*: 20
- `-o`, `--output` : Output directory for simulation data

Flags:

- `-v`, `--verbose`: Sets the verbosity of the console output
- `-q`, `--quiet`: Suppresses all console output
- `-c`, `--cache`: Cache python files, parameter file, and graph pickle file
- `-nmc`, `--no_mc`: Only performs one run.
- `-gpu`, `--gpu` : Use GPU. *Default* : Uses GPU if cupy is installed

Each Monte Carlo run produces its own `.feather` file. These files are placed in a subdirectory corresponding to the Monte Carlo ID (created by timestamp).

6.3 Data Postprocessing

NOTE: Currently, all postprocessing scripts use the graph file that was used during the simulation. This is used to create a lookup table to specify the relation between admin levels (e.g. admin 2 codes belonging to admin 1).

6.3.1 Monte Carlo Aggregation

Before any visualization or analysis can be performed, the raw output needs to be postprocessed. This is done by `postprocess.py`. This script aggregates data at the requested levels (e.g. admin1-level) and produces two types of files: one containing quantiles and one containing mean and standard deviation.

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 mean and standard deviation at the national level: `/output/2020-06-10__14_13_04/adm0_mean_std.csv`

Arguments:

- file (Positional): Directory location of model-created `.feather` files. *Default*: Most recently created subdirectory in `raw_output/`
- `--graph_file`, `-g`: Graph file used during model. *Default*: Most recently created graph in default graph directoy, `data/input_graphs/`
- `--levels`, `-l`: Aggregation levels. *Default*: adm0, adm1, adm2
- `--quantiles`, `-q`: Quantiles to calculate. *Default*: 0.05, 0.25, 0.5, 0.75, 0.95
- `--output`, `-o`: Directory to place the subfolder containing aggregated files. *Default*: `output/`
- `--prefix`: Prefix for the subfolder name. *Default*: None
- `--end_date`: If a user passes in an end date, aggregated data will not include data past this point. *Default*: None
- `--verbose`, `-v`: Prints extra information during postprocessing.
- `--no_quantiles`: Skips computing quantiles and computes only mean and standard deviation.
- `--lookup`: Pass in an explicit lookup table for geographic mapping info. See below caveat.

Lookup Tables

By default, postprocessing uses geographic information on the graph to aggregate geographic areas. For special cases, a lookup table may be passed in via the `--lookup` command. This is intended to be used for splitting states into non-FIPS divisions. When a lookup table is passed in, the output directory will be prepended with a string to distinguish it from output created from the same simulation using the graph file.

6.3.2 Visualization

The Bucky model has the capability to create two types of visualization: plots and maps. Both are contained within the `viz/` directory.

All visualizations are placed in subfolders in the same directory as the aggregated directory. Plots are placed in `plots/` and maps are placed in `maps/`. These folders can be renamed with command-line arguments, but will still be placed within the aggregated data folder.

Example:

```
2020-07-28__15_21_52/
├── adm0_mean_std.csv
├── adm0_quantiles.csv
├── adm1_mean_std.csv
├── adm1_quantiles.csv
├── adm2_mean_std.csv
├── adm2_quantiles.csv
├── maps
│   └── ADM1
│       ├── adm1_AlabamaDailyReportedCases2020-07-26.png
│       ├── adm1_AlabamaDailyReportedCases2020-08-02.png
│       ├── ...
└── plots
    ├── ADM1
    │   ├── Alabama.png
    │   ├── ...
    ├── US.csv
    └── US.png
```

Plots

Plots can be created at any of the three admin levels. Each plot contains two subplots. These plots can optionally include historical data. Example usage:

```
python3 viz/plot.py -i output/2020-06-10__14_13_04/ --hist
```

NOTE: By default, `plot.py` makes admin0 and admin1 plots. In order to create admin2-level plots, a user must also pass in an admin1 name. For example, to create county-level plots for Arizona, `--adm1_name Arizona` must be passed in as an argument.

Arguments and flags:

- `--input_dir`, `-i`: Directory location of aggregated data. *Default*: Most recently created subdirectory in the default directory for processed data (`output/`)
- `--output`, `-o`: Output directory for plots. *Default*: `$INPUT_DIR/plots/`
- `--graph_file`, `-g`: Graph file used during model. *Default*: Most recently created graph in default graph directoy, `data/input_graphs/`
- `--levels`, `-l`: Requested plot levels. *Default*: adm0, adm1
- `--hist`, `-hist`: Plot historical data in addition to simulation data.
- `--window_size`, `-w`: Size of window (in days) to apply to historical data. *Default*: 7
- `--plot_columns`: Columns to plot. *Default*: daily_cases_reported, daily_deaths

- `--hist_columns`: Historical columns to plot. Note: If only one historical column is passed in, historical data will only be present on the top plot. *Default*: Confirmed, Deaths
- `--hist_start`: Start date of historical data. If not passed in, will align with start date of simulation
- `--hist_file`: History file to use. *Default*: Will use US CSSE historical data
- `--end_date`: If a user passes in an end date, data will not be plotted past this point.
- `--adm1_name`: Name of adm1 to create adm2 plots for. *Default*: None
- `--lookup`: Pass in an explicit lookup table for geographic mapping info
- `-v`, `--verbose` : Prints extra information during plotting

By default, confidence intervals are plotted using quantiles. Optionally, the standard deviation can be used by passing in the following arguments:

- `--n_mc`, `-n`: Number of Monte Carlo runs from simulation. *Default*: 1000
- `--use_std`: Flag to indicate standard deviation should be used instead of quantiles.

Historical Data

The plotting utility expects historical data to be at the adm2-level.

Maps

Maps are created at the adm0 or adm1 level. In order to create maps, shapefiles must be provided one level down from the requested map (e.g. adm2-level shapefile must be provided for adm1-level maps). Maps can be created for specific dates or distributed throughout the length of the simulation with a requested frequency.

Example usage:

```
python3 viz/map.py -i output/2020-06-10__14_13_04/  --all_adm1--adm2_shape data/
↪shapefiles/tl_2019_us_county.shp --dates 2020-06-01
```

Arguments and flags:

- `--input_dir`, `-i`: Directory location of processed simulation data, *Default*: Most recently created subdirectory in default output directory (`output/`)
- `--output`, `-o`: Output directory for maps. *Default*: `$INPUT_DIR/maps/`
- `--graph_file`, `-g`: Graph file used during model. *Default*: Most recently created graph in default graph directoy, `data/input_graphs/`
- `--adm0`: Create adm0-level plot
- `--adm1`: Create adm1-level plot for the requested adm1 name
- `--all_adm1`: Create adm1-level plot for every available adm1-level area.
- `--adm1_shape`: Location of adm1 shapefile. *Default*: `data/shapefiles/tl_2019_us_state.shp`
- `--adm2_shape`: Location of adm2 shapefile. *Default* : `data/shapefiles/tl_2019_us_county.shp`
- `--columns`: Data columns to plot. Maps are created separately for each requested column. *Default*: daily_cases_reported, daily_deaths
- `--dates`, `-d`: Dates to create maps. If passed in, takes priority over `frequency`. *Default*: None

- `--freq`, `-f`: Frequency at which to create maps. Allowed values: daily, weekly, monthly. *Default*: weekly
- `--mean`: Use mean value instead of median value for map
- `--linear`: Use linear scaling for values instead of log
- `--cmap`, `-c`: Colormap to use. Must be a valid matplotlib colormap. *Default*: Reds
- `--lookup`: Pass in an explicit lookup table for geographic mapping info

CHAPTER

SEVEN

CLI INTERFACE

7.1 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]
```

7.1.1 Named Arguments

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

Default: "2020-10-11"

-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

7.2 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]
```

7.2.1 Positional Arguments

par_file File containing paramters

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

7.2.2 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

7.3 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]
```

7.3.1 Positional Arguments

file File to proess

Default: "Most recently created folder in raw_output_dir"

7.3.2 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

7.4 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]
```

7.4.1 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

7.5 bucky.viz.map

CHAPTER

EIGHT

BUCKY

8.1 bucky package

8.1.1 Subpackages

bucky.util package

Submodules

bucky.util.read_config module

bucky.util.readable_col_names module

bucky.util.update_data_repos module

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-level population data as as fraction of total population across requested 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*)

Takes a historical data file and reduces it 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 Atlantic's COVID Tracking project to match the format of other pre-processed data sources.

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.**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.plot module

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

8.1.2 Submodules

8.1.3 bucky.arg_parser_model module

arg parser for bucky.model

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

8.1.4 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

[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

8.1.5 bucky.make_state_csv module

8.1.6 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_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*)

8.1.7 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*)

8.1.8 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

8.1.9 Module contents

CHAPTER

NINE

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

TEN

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

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

PYTHON MODULE INDEX

b

INDEX

I

M

P

R

S

T

U