src Documentation

Release 0.1.13

Author

May 16, 2020

Contents:

1 EpiStoch **3**
- 1.1 Epidemics Models with Random Infectious Period . . . 3
- 1.2 Models . . . 4
- 1.3 Notes . . . 4

2 Installation **5**
- 2.1 Stable release . . . 5
- 2.2 From sources . . . 5

3 Usage **7**

4 Reference Manual **9**
- 4.1 epistoch package . . . 9
  - 4.1.1 Subpackages . . . 9
    - 4.1.1.1 epistoch.utils package . . . 9
  - 4.1.2 Submodules . . . 11
  - 4.1.3 epistoch.experiments module . . . 11
  - 4.1.4 epistoch.seird_ph module . . . 12
  - 4.1.5 epistoch.sir_g module . . . 12
  - 4.1.6 epistoch.sir_phg module . . . 14
- 4.2 pyphase package . . . 15
  - 4.2.1 Submodules . . . 15
  - 4.2.2 pyphase.phase module . . . 15

5 Contributing **19**
- 5.1 Types of Contributions . . . 19
  - 5.1.1 Report Bugs . . . 19
  - 5.1.2 Fix Bugs . . . 19
  - 5.1.3 Implement Features . . . 19
  - 5.1.4 Write Documentation . . . 20
  - 5.1.5 Submit Feedback . . . 20
- 5.2 Get Started! . . . 20
- 5.3 Pull Request Guidelines . . . 21
- 5.4 Tips . . . 21
- 5.5 Deploying . . . 21

6 Credits **23**

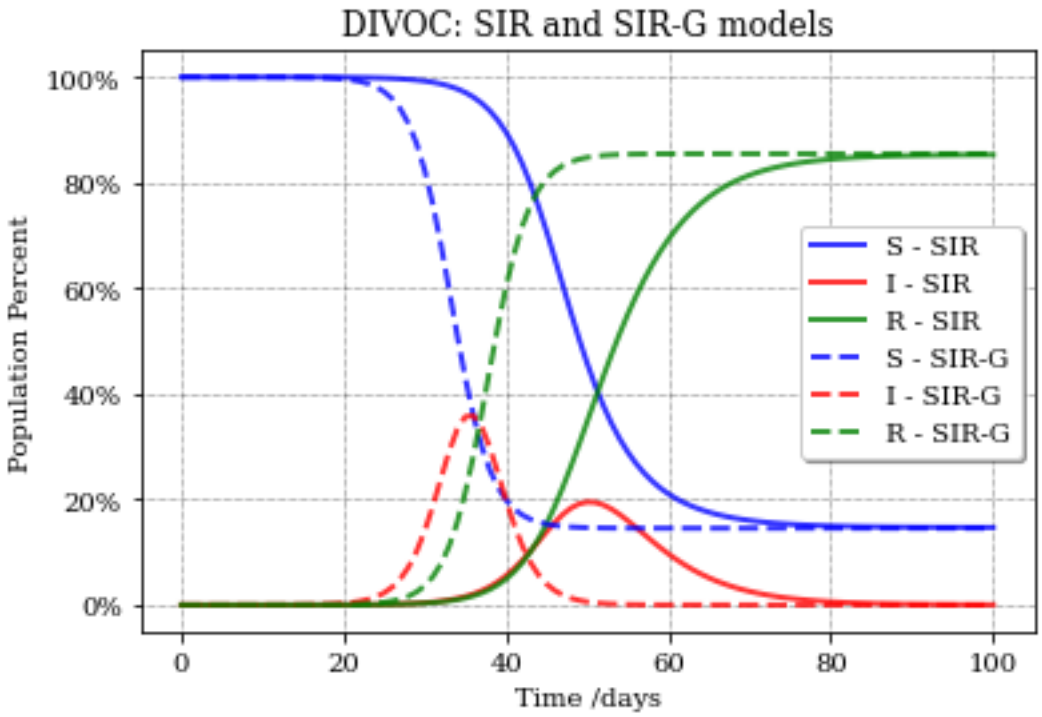
6.1 Development Lead 23
6.2 Contributors 23

7 History **25**
7.1 1.0 (2020-05-01) 25

8 Indices and tables **27**

Python Module Index **29**

Index **31**


DIVOC: SIR and SIR-G models
Population Percent
100%
80%
60%
40%
20%
0%
0
20
40
60
80
100
Time /days
S - SIR
I - SIR
R - SIR
S - SIR-G
I - SIR-G
R - SIR-G

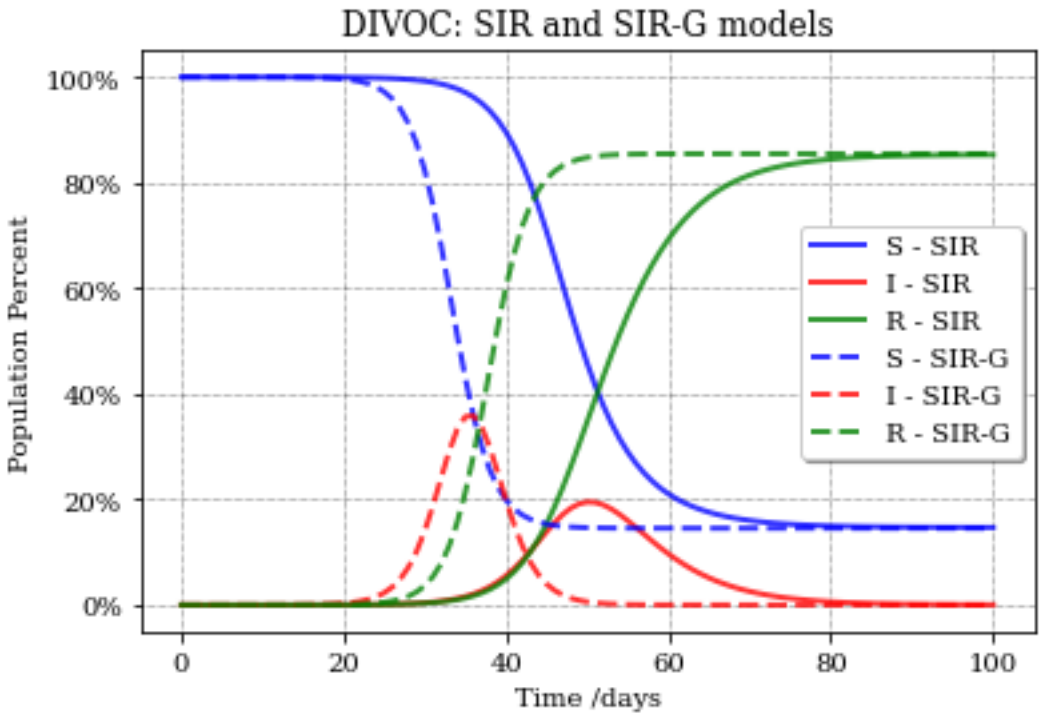
CHAPTER 1

EpiStoch

1.1 Epidemics Models with Random Infectious Period

This software allows you to model epidemics with general random distribution for the infectious period.

Traditional epidemiology models, like SIR, do not take into account the distribution for the length of the infectious period. In this software, we include three functions that compute these type of models using other distributions.



In this graph you can see how different the predictions are for the regular SIR model with respect to SIR-G that actually uses a more realistic distribution for the infectious period. In SIR-G case the peak of infection occurs before, and has a bigger intensity. The number of individuals that eventually get infected, however, remains the same for both models

1.2 Models

- SIR: Classical SIR model, with (implied) exponential infectious period.
- SIR_G: Like the classical SIR model, but with an arbitrary distribution.
- SIR-PHG: A SIR model with Phase-Type distributions for the infectious period.
- SEIRD: A SEIRD Model with hase-Type distributions for each stage.

1.3 Notes

- The theoretical foundation of the method is explained in this paper.
- Documentation: https://epistoch.readthedocs.io.
- Source Code: https://github.com/griano/epistoch.
- Free software: MIT license. THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

CHAPTER 2

Installation

2.1 Stable release

To install EpiStoch, run this command in your terminal:

```
$ pip install epistoch
```

This is the preferred method to install EpiStoch, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for EpiStoch can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/griano/epistoch
```

Or download the tarball:

```
$ curl -OJL https://github.com/griano/epistoch/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

In this example we compute a SIR-G model with Gamma distribution:

```
from epistoch import *
from epistoch.utils.plotting import plot_sir
from scipy import stats
import matplotlib.pyplot as plt


# Let's build a SIR-G model

dist = stats.gamma(a=2, scale=10)
# The expected time is 20 days

SIR_general = sir_g(
    name="SIR-G-Example",
    population=1000,
    num_days=230,
    reproductive_factor=2.2,
    infectious_time_distribution=dist,
    method="loss",
)

# Report a summary
report_summary(SIR_general)

# Now plot the result
plot_sir(SIR_general)
plt.show()

# The data can be seen in
print(SIR_general["data"])
```

CHAPTER 4

Reference Manual

4.1 epistoch package

Top-level package for EpiStoch.

4.1.1 Subpackages

4.1.1.1 epistoch.utils package

Submodules

epistoch.utils.plotting module

Created on Tue Apr 21 17:14:05 2020

@author: Germán Riaño

epistoch.utils.plotting.**plot_IR**(*model*, *fig=None*, *linestyle='-'*, *title=None*, *legend_fmt={'loc': 'best', 'shadow': True}*, *use_latex=False*)

epistoch.utils.plotting.**plot_sir**(*model*, *fig=None*, *title=None*, *formats={'I': 'r-', 'R': 'g-', 'S': 'b-'}*, *legend_fmt={'loc': 'best', 'shadow': True}*, *use_latex=False*, ***kwd*)

epistoch.utils.plotting.**prepare_plot**(*title*, *use_latex=False*)

epistoch.utils.plotting.**set_plt_for_latex**()

epistoch.utils.stats module

Created on Wed Apr 15 15:14:55 2020

@author: Germán Riaño

class epistoch.utils.stats.**ConstantDist**(*momtype=1*, *a=None*, *b=None*, *xtol=1e-14*, *badvalue=None*, *name=None*, *longname=None*, *shapes=None*, *extradoc=None*, *seed=None*)

Bases: scipy.stats._distn_infrastructure.rv_continuous

epistoch.utils.stats.**avg_recovery_rate**(*dist*)

epistoch.utils.stats.**get_gamma**(*mu*, *sigma*)

Builds a gamma distribution from mean and standard deviation for this variable.

Parameters

- **mu** (*double*) – The expected value
- **sigma** (*double*) – standard deviation

Returns

Return type A frozen ditribution object

epistoch.utils.stats.**get_lognorm**(*mu*, *sigma*)

Builds a lognormal distribution from mean and standard deviation for this variable. (Not the mean and sd of the corresponding normal)

Parameters

- **mu** (*double*) – The expected value
- **sigma** (*double*) – standard deviation

Returns

Return type A frozen ditribution object

epistoch.utils.stats.**loss_function**(*dist*, *force=False*)

Creates a loss function of order 1 for a distribution from scipy

Parameters

- **dist** (*scipy.stats._distn_infrastructure.rv_froze*) – a distribution object form scipy.stats
- **force** (*boolean*) – whether force an integral computation instead of known formula

Returns

Return type Callable that represent this loss function

epistoch.utils.utils module

epistoch.utils.utils.**compute_integral**(*n*, *delta*, *S*, *I*, *times*, *survival*, *pdfs*, *loss1*, *dist*, *method='loss'*)

Compute the integral needed for the integro-differential model.

In other words, computes

$$\int_0^t g(t-x)I(x)S(x)dx \quad \text{for } t = n * \delta$$

Parameters

- **n** (*integer*) – upper limit for integral.
- **delta** (*float*) – interval size

- **S** (*array*) – Susceptible
- **I** (*array*) – Infected
- **times** (*array*) – times at which the arraya are evaluated
- **survival** (*array of float*) – array $G_k \equiv P\{T > k\delta\}$.
- **pdfs** (*array of float*) – array $g(k\delta)$.
- **array float** (*loss1*) – $L_k \equiv L(k\delta)$
- **dist** (*rv_continuous*) – Object that represents the distribution
- **method** (*string*) – One from "loss", "simpson" or "interpolate"

Returns

Return type Value of the integral

epistoch.utils.utils.**get_total_infected**(*reproductive_factor*, *normalized_s0=1*)

Estimate the total number of infected for a given reproductive factor.

Find the value z that solves

$$1 - z = S_0 * e^{\mathcal{R}_0 z},$$

where $\mathcal{R}_0$ is the `reproductive_factor`, and S_0 is the (normalized) initial population `normalized_s0`.

Parameters

- **reproductive_factor** (*float*) – Basic reproductive factor.
- **normalized_s0** (*float, optional*) – Initial fraction of population that is infected. The default is 1.

Returns The stimated fraction of total infected.

Return type float

epistoch.utils.utils.**print_error**(*model1*, *model2*)

Print error difference between two models

Parameters

- **model1** – Result of a SIR, SIR_G model
- **model2** – Result of a SIR, SIR_G model

epistoch.utils.utils.**report_summary**(*result*)

Report a summary for a model.

Parameters **result** – Result from SIR like model

4.1.2 Submodules

4.1.3 epistoch.experiments module

epistoch.experiments.**compare_models**(*name*, *dist*, *N*, *I0=1*, *num_days=100*, *R0=2.25*, *do_plots=True*, *use_latex=False*, *plot_path='.'*, *plot_ext='pdf'*)

epistoch.experiments.**variance_analysis**(*N=1000000, I0=1, num_days=100, R0=2.25, infectious_period_mean=4.4, cvs=[0.5, 1.0, 2.0], use_latex=False, plot_path='.', plot_ext='pdf'*)

4.1.4 epistoch.seird_ph module

Created on Sun Apr 26 21:50:11 2020

@author: Germán Riaño, PhD

epistoch.seird_ph.**seird_ph**(*name*, *population*, *beta*, *exposed_time*, *time_to_die*, *time_to_recover*, *fatality_rate*, *num_days*, *I0=1*, *logger=None*)

Compute a SEIRD model with Phase-Type distribution for the different stages.

Parameters

- **name** (*string*) – Model name.
- **population** (*int or float*) – Population size.
- **beta** (*float*) – Contagion rate.
- **exposed_time** (*PH distribution*) – Distribution of time exposed, not yet infectious.
- **time_to_die** (*PH distribution*) – Time to die after becoming infectious.
- **time_to_recover** (*PH distribution*) – Time to recover after becoming infectious.
- **fatality_rate** (*float*) – Percentage of individuals that die.
- **num_days** (*int*) – Number of days ot analyze.
- **I0** (*int, optional*) – Initial infected population. The default is 1.
- **logger** (*Logger object, optional*) – Logger object. If not given default logging is used.

Returns

result –

Dictionary with fields:

- name: model name
- population: Total population
- **data: data Frame with columns**
  - S : Susceptible,
  - E : Exposed,
  - I : Infectious (Dying or recovering),
  - Rc : Total Recovered,
  - D : Total deaths
  - R : Removed (R+D),

Return type dict

4.1.5 epistoch.sir_g module

Created on Sat Apr 11 12:20:32 2020

@author: Germán Riaño (griano@germanriano.com)

epistoch.sir_g.**sir_classical**(*name*, *population*, *reproductive_factor*, *infectious_period_mean*, *num_days*, *I0=1.0*, *S0=None*, *times=None*)

Solves a classical SIR model.

Parameters

- **name** (*string*) – Model name
- **population** (*float*) – total population size
- **reproductive_factor** (*float*) – basic reproductive factor(R0)
- **infectious_period_mean** (*float*) – expected value of Infectious Period Time
- **I0** (*float*) – Initial infectious population
- **S0** (*float*) – Initial susceptible population (optinal, defaults to all but I0)
- **num_days** (*int*) – number of days to run
- **times** (*array of float*) – times where the functions should be reported. Defaults to `np.linspace(0, num_days, num_days + 1)`

Returns

Dictionary with fields:

- name: model name
- population: Total population
- total_infected
- **data: data Frame with columns**
  - S : Susceptible,
  - I : Infectious (Dying or recovering),
  - R : Removed (recovered + deaths),

Return type dict

epistoch.sir_g.**sir_g**(*name*, *population*, *reproductive_factor*, *infectious_time_distribution*, *num_days*, *I0=1.0*, *S0=None*, *num_periods=None*, *method='loss'*, *logger=None*)

Solves a SIR-G model.

Parameters

- **name** (*string*) – Model name
- **population** (*float*) – total population size
- **reproductive_factor** (*float*) – basic reproductive factor(R0)
- **infectious_time_distribution** (*scipy.stats.rv_continuous*) – expected value of Infectious Period Time
- **num_days** (*int*) – number of days to run
- **I0** (*float*) – Initial infectious population
- **S0** (*float*) – Initial susceptible population (optional, defaults to all but I0)
- **num_periods** (*int*) – Number of periods to use for computations. Higher number will lead ot more precise computation.
- **method** (*string*) – Method used for the internal integral
- **logger** – Logger object

Returns

Dictionary with fields:

- name: model name
- population: Total population
- total_infected
- **data: data Frame with columns**
  - S : Susceptible,
  - I : Infectious (Dying or recovering),
  - R : Removed (recovered + deaths),

Return type dict

4.1.6 epistoch.sir_php module

Created on Sun Apr 26 11:43:11 2020

@author: Germán Riaño, PhD

epistoch.sir_php.**sir_php**(*name*, *population*, *beta*, *infectious_time_distribution*, *num_days=160*, *I0=1.0*, *S0=None*, *logger=None*, *report_phases=False*)

Compute a SIR-PH model

Parameters

- **name** (*string*) – Model name
- **population** (*float*) – Total population.
- **beta** (*float, optional*) – Contagion rate. The default is 0.2.
- **infectious_time_distribution** (*phase*) – Must be a PH distribution.
- **num_days** (*int*) – Number of days to run.
- **I0** (*float, optional*) – initial population. The default is 1.0.
- **S0** (*float or int, optional*) – Initial susceptible. The default is all but I0.
- **logger** (*Logger, optional*) – Logger object. If none is given, default logging used.
- **report_phases** (*boolean, optional*) – Whether to include phase levels and removed from every level in the report. The default is False.

Returns

- *dict*
- *A dictionary with these fields –*
  - name: model name
  - population: total population
  - total_infected: estimation of total infected individuals
  - **data: DataFrame with columns**
    - S : Susceptible

* I : Infected individuals
* R : Removed Individuals
* Optionally: I-Phase0,...,I-Phase(n-1), and R-Phase0,...R-Phase(n-1)

epistoch.sir_phg.**to_array**(**args*)

4.2 pyphase package

4.2.1 Submodules

4.2.2 pyphase.phase module

Module pyphase.phase.

This module allows you to work with Phase-Type distributions.

The CDF for PH variable is given by

$$F(t) = 1 - \boldsymbol{\alpha} e^{\boldsymbol{A}t}\mathbf{1} \qquad x \geq 0,$$

where $\boldsymbol{\alpha}$ is a probability vector and the matrix $\boldsymbol{A}$ has the transition rates between the phases.

To build a PH variable you need the vectoe `alpha` and matrix `A`

```
>>> from pyphase import *
>>> alpha = [0.1 , 0.9]
>>> A = [[-2.0 , 1.0], [1.5, -3.0]]
>>> v = phase(alpha, A)
>>> print(v)
PhaseType:
  alpha = [[0.1 0.9]]
  A     = [[-2.   1. ]
           [ 1.5 -3. ]]
```

After you have the variable you can do all operations that are typical with random variables in scipy.stats:

```
>>> print(f"The mean is {v.mean():.3f}")
The mean is 0.789
>>> print(f"v.cdf(1.0)={v.cdf(1.0):.3f}")
v.cdf(1.0)=0.721
```

class pyphase.phase.**PhaseType**(*alpha*, *A*, *dist*, **args*, ***kwds*)

Bases: scipy.stats._distn_infrastructure.rv_frozen

Represent a Phase-Type distribution.

Users should not call it directly but rather call `phase(alpha, A)`

equilibrium_pi()

Return the equilibrium distribution of the associated PH distribution.

In other word, it finds the vector pi that solves

$$\boldsymbol{\pi} = \boldsymbol{\pi}(\boldsymbol{A} + \boldsymbol{\alpha a}), \qquad \boldsymbol{\pi}\mathbf{1} = 1$$

Returns vector :math"*pmbpi* that solves the equilibrium equations.

Return type array

loss1(*x*)
First order loss function.

For a random variable C teh first order loss function is defined as

$$L(x) = [E(X - x)^+]$$

Parameters **x** (*float*) – The value at which the first order distribution is evaluated

Returns Value of the loss function at x.

Return type float

params()
Returns the parameters of this PH distribution.

Returns $(\boldsymbol{\alpha}, \boldsymbol{A}, \boldsymbol{a} = -\boldsymbol{A}\mathbf{1}$, dimension)

Return type tuple

pyphase.phase.**ph_erlang**(*n*, *lambd=None*, *mean=None*)
Builds an Erlang(n,lmbda).

If mean is provided then lambda = n/mean.

Parameters

- **n** (*int*) – The order of this Erlang
- **lambd** (*float, optional*) – Provide only one between lambd or mean
- **mean** (*float, optional*) – Provide only one between lambd or mean

Returns PH representation for an Erlang

Return type PH

pyphase.phase.**ph_expon**(*lambd=None*, *mean=None*)
Builds an exponential distribution represented as PH.

If mean is provided then lambda = 1/mean.

Parameters

- **lambd** (*float, optional*) – Rate value. One between lambd an mean must be given.
- **mean** (*float, optional*) – Mean value. One between lambd an mean must be given.

Returns PH object.

Return type PH

pyphase.phase.**ph_mix**(*ph1*, *ph2*, *p1*)
Produces a PH variable thet is a mixture of the two given PH variables.

Parameters

- **ph1** (*PH*) – The first variable.
- **ph2** (*PH*) – The second variable.
- **p1** (*float*) – Probability of choosing the first variable. 0 <= p1 <= 1

Returns The resulting PH variable.

Return type PH

pyphase.phase.**ph_sum**(*ph1*, *ph2*)

Produces a new PH that is the sum of the given PH variables.

Parameters

- **ph1** (*PH*) – The first variable.
- **ph2** (*PH*) – The second variable.

Returns The resulting PH variable.

Return type PH

pyphase.phase.**phase**(*alpha*, *A*)

Creates a new PH variable.

This method builds a PhaseType object by given the array $\boldsymbol{\alpha}$ and matrix $\boldsymbol{A}$. The CDF for PH variable is given

$$F(t) = 1 - \boldsymbol{\alpha} e^{\boldsymbol{A}t}\mathbf{1} \qquad x \geq 0$$

Parameters

- **alpha** (*array of float*) – The initial probabilities vector. It must satisfy $\sum_i \alpha_i \leq 1$.
- **A** (*matrix of float*) – The transition matrix between phases.

Returns An object representing this distribution.

Return type PH

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/griano/epistoch/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

EpiStoch could always use more documentation, whether as part of the official EpiStoch docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/griano/epistoch/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *epistoch* for local development.

1. Fork the *epistoch* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/epistoch.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv epistoch
$ cd epistoch/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 epistoch tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/griano/epistoch/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_epistoch
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Germán Riaño <griano@germanriano.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 1.0 (2020-05-01)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

e

epistoch, 9
epistoch.experiments, 11
epistoch.seird_ph, 12
epistoch.sir_g, 12
epistoch.sir_phg, 14
epistoch.utils, 9
epistoch.utils.plotting, 9
epistoch.utils.stats, 9
epistoch.utils.utils, 10

p

pyphase, 15
pyphase.phase, 15

Index

A

avg_recovery_rate() (*in module epistoch.utils.stats*), 10

C

compare_models() (*in module epistoch.experiments*), 11
compute_integral() (*in module epistoch.utils.utils*), 10
ConstantDist (*class in epistoch.utils.stats*), 9

E

epistoch (*module*), 9
epistoch.experiments (*module*), 11
epistoch.seird_ph (*module*), 12
epistoch.sir_g (*module*), 12
epistoch.sir_phg (*module*), 14
epistoch.utils (*module*), 9
epistoch.utils.plotting (*module*), 9
epistoch.utils.stats (*module*), 9
epistoch.utils.utils (*module*), 10
equilibrium_pi() (*pyphase.phase.PhaseType method*), 15

G

get_gamma() (*in module epistoch.utils.stats*), 10
get_lognorm() (*in module epistoch.utils.stats*), 10
get_total_infected() (*in module epistoch.utils.utils*), 11

L

loss1() (*pyphase.phase.PhaseType method*), 16
loss_function() (*in module epistoch.utils.stats*), 10

P

params() (*pyphase.phase.PhaseType method*), 16
ph_erlang() (*in module pyphase.phase*), 16
ph_expon() (*in module pyphase.phase*), 16
ph_mix() (*in module pyphase.phase*), 16
ph_sum() (*in module pyphase.phase*), 17
phase() (*in module pyphase.phase*), 17
PhaseType (*class in pyphase.phase*), 15
plot_IR() (*in module epistoch.utils.plotting*), 9
plot_sir() (*in module epistoch.utils.plotting*), 9
prepare_plot() (*in module epistoch.utils.plotting*), 9
print_error() (*in module epistoch.utils.utils*), 11
pyphase (*module*), 15
pyphase.phase (*module*), 15

R

report_summary() (*in module epistoch.utils.utils*), 11

S

seird_ph() (*in module epistoch.seird_ph*), 12
set_plt_for_latex() (*in module epistoch.utils.plotting*), 9
sir_classical() (*in module epistoch.sir_g*), 12
sir_g() (*in module epistoch.sir_g*), 13
sir_phg() (*in module epistoch.sir_phg*), 14

T

to_array() (*in module epistoch.sir_phg*), 15

V

variance_analysis() (*in module epistoch.experiments*), 11