diffusive$_{d}istinguishability$Documentation

Release 0.1.0

Julie Cass

May 03, 2019

Contents:

CHAPTER 1

diffusive_distinguishability

Simulation of homogeneous diffusion, bayesian estimation of underlying diffusion constant and analysis of distinguishability between diffusivities

1.1 Getting Started

The python package `ndim_homogeneous_distinguishability.py` contains the meat of this project, as a set of functions which can be used to:

1. Simulate diffusive trajectories (pure diffusion with a homogeneous diffusion constant)
2. Use Bayesian inference to estimate the diffusion constant used to generate a trajectory by producing a posterior diffusivity distribution
3. Analyze the dependence of diffusivity estimation error, and the ability to distinguish between trajectories with differing diffusivities, conditional on model parameters

Examples of how to use these functions, as well as some of our own analysis of diffusivity distinguishability, are provided in the Jupyter notebook `ndim_diffusion_analysis_tutorial.ipynb`.

Also included are some stored pre-calculated numpy arrays used in the provided Jupyter notebook example analysis (in the directory `loc_error_saved_files`) and another Jupyter notebook containing a toy model quantifying the relative impact of localization error on diffusion estimates conditional on number of spatial dimensions (`test_overestimation.ipynb`).

- Free software: Allen Institute Software License
- Documentation: https://diffusive-distinguishability.readthedocs.io.

1.2 Support

We are not currently supporting this code, but simply releasing it to the community AS IS but are not able to provide any guarantees of support. The community is welcome to submit issues, but you should not expect an active response.

1.3 Associated text

Below is a working abstract for the 'Bayesian detection of diffusive heterogeneity' project found in this repo:

1.3.1 Abstract

Cells are crowded and spatially heterogeneous, complicating the transport of organelles,proteins and other substrates. The diffusion constant partially characterizes dynamicsin complex cellular environments but, when taken on a per-cell basis, fails to capturespatial dependence of diffusivity. Measuring spatial dependence of diffusivity ischallenging because temporally and spatially finite observations offer limitedinformation about a spatially varying stochastic process. We present a Bayesianframework that estimates diffusion constants from single particle trajectories, andpredicts our ability to distinguish differences in diffusion constants, conditional on howmuch they differ and the amount of data collected.

1.3.2 Introduction

Diffusion is essential for the intracellular transport of organelles, proteins and substrates, and is commonly characterized through analyses of single particle tracking (SPT) in live-cell images. While powerful analyses from SPT have indicated the complexity of transport in live cells, the spatial variation of the diffusion constant remains poorly characterized. This can be attributed to challenges in disentangling effects of biological heterogeneity and limited sampling of a stochastic process. To address these challenges, we developed a Bayesian framework to estimate a posterior distribution of the possible diffusion constants underlying SPT dynamics. This framework can be used to generate a look-up table predicting the detectability of differences in diffusion constants, conditional on the ratio of their values and amount of trajectory data collected.

1.3.3 Materials and methods

We simulate particle diffusion in a range of homogeneous diffusion constants, and digest the resulting trajectories into frame-to-frame displacements. Using an inverse-gamma conjugate prior, we make the conservative guess that any order of magnitude of diffusion constant is equally likely. The set of displacements in a single trajectory are used to generate a posterior inverse-gamma distribution estimating the probability that any given diffusion constant was used to generate the trajectory. This distribution peaks near the true diffusion constant and has a width corresponding to the confidence interval of our estimate, which is largely determined by the trajectory length. Given a pair of posteriors derived for trajectories with differing underlying diffusion constants, we can characterize their similarity by computing the Kullback-Leibler divergence. This metric acts as a single-value estimation of how well we can analytically distinguish that trajectories were generated from different diffusion constants. For longer trajectory lengths, stochastic variations will be less dominant, increasing distinguishability.

1.3.4 Results

To assess the conditional feasibility of computationally detecting differences in diffusivity, we generate a landscape of the KL divergence between posteriors generated from pairs of simulations, with varying trajectory lengths and differences in diffusivity. To further correct for stochastic variations in simulations, the KL divergence reported for each

entry in the landscape is the mean value from thousands of replicates. We find that, using this method, diffusivities differing by a factor of 1.5 or more can be easily distinguished when at least 50 timepoints are reported for each trajectory. This landscape offers a look-up table for estimating the number of frames that must be acquired experimentally to distinguish diffusivities to a desired precision. This framework could therefore play a valuable role in describing the feasibility of and requirements for experiments addressing the spatial heterogeneity of the intracellular diffusive environment. To address the affects of static localization error of punctate objects from microscopy images, we included Gaussian error to the particle location at each point in its trajectory. The standard deviation of this Gaussian determines the amount of localization error applied. Now, error in the ability to detect the underlying diffusion constant is a compound error due to the affects of both localization error and error in Bayesian estimation of the posterior maximum.

1.3.5 Conclusion

The spatial heterogeneity of diffusion may have major impacts in the transport ofessential cellular substrates but remains largely uncharacterized. To shed light on thefeasibility of resolving spatial from stochastic drivers of diffusive heterogeneity intrajectory data, we developed a framework for predicting our ability to detect differences in diffusivity, conditional on the amount of experimental data collected. Our framework can therefore be used to inform the design of experiments aimed to characterize the spatial dependence of diffusivity across cells.

1.4 Credits

This package was created with Cookiecutter.

CHAPTER 2

Installation

2.1 Stable release

To install diffusive_distinguishability, run this command in your terminal:

```
$ pip install diffusive_distinguishability
```

This is the preferred method to install diffusive_distinguishability, 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 diffusive_distinguishability can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/jcass11/diffusive_distinguishability
```

Or download the tarball:

```
$ curl  -OL https://github.com/jcass11/diffusive_distinguishability/tarball/master
```

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

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

CHAPTER 3

Usage

To use diffusive_distinguishability in a project:

```
import diffusive_distinguishability
```

CHAPTER 4

diffusive_distinguishability

4.1 diffusive_distinguishability package

4.1.1 Subpackages

diffusive_distinguishability.bin package

Module contents

Bin scripts package for diffusive_distinguishability.

diffusive_distinguishability.tests package

Submodules

diffusive_distinguishability.tests.test_ndim_homogeneous_distinguishability module

Module contents

Unit test package for diffusive_distinguishability.

4.1.2 Submodules

4.1.3 diffusive_distinguishability.ndim_homogeneous_distinguishability module

diffusive_distinguishability.ndim_homogeneous_distinguishability.**compare2**(*n_dim*, *d_const1*, *mult*, *n_steps*, *dt*, *n_reps*, *loc_std=0*)

For one pair of diffusion constants (d_const, d_const*mult) get KL divergence of their posteriors, where the posteriors are generated from an alpha and beta which are the median values from repeating posterior estimation n_reps times.

Parameters

- **n_dim** – number of spatial dimensions
- **d_const1** – diffusion constant (um2/s)
- **mult** – multiplier to get d_const2 = mult*d_const1
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **n_reps** – number of trajectory replicates
- **loc_std** – standard deviation of localization error (um)

diffusive_distinguishability.ndim_homogeneous_distinguishability.**error_sensitivity**(*d_const*, *n_steps_lis*, *dt*, *n_reps*, *loc_std*)

Look at how the mean and median percent error of the posterior mean relative to the true value depend on the trajectory length used to generate posteriors and number of reps we run.

Parameters

- **d_const** – diffusion constants (um2/s)
- **n_steps_list** – list of trajectory lengths to test
- **dt** – timestep(s) used to generate trajectories (s)
- **n_reps** – number(s) of reps to run to calculate mean and mediate percent error
- **loc_std** – standard deviation for Gaussian localization error (um)

Returns three dataframes (for 1, 2, and 3 dimensions); each contains the mean percent posterior error relative to

true diffusion constant value, for all pairs of trajectory lengths and localization errors includes in these two input lists

diffusive_distinguishability.ndim_homogeneous_distinguishability.**estimate_diffusion**(*n_dim*, *dt*, *dr*, *prior=<s ob- ject>*)

Returns the posterior estimate for the diffusion constant given the displacement data and the prior.

Parameters

- **n_dim** – number of spatial dimensions for simulation (1, 2, or 3)
- **dt** – timestep size (s)
- **dr** – list of normed step sizes from a single trajectory (um)
- **prior** – inverse gamma prior distribution estimate for the diffusion constant

Returns inverse gamma posterior distribution estimate for the diffusion constant

diffusive_distinguishability.ndim_homogeneous_distinguishability.**fill_heatmap_gen**(*n_dim*, *d_const*, *mult_list*, *n_steps*, *dt*, *n_reps*, *loc_std=0*)

Generate a heatmap of KL divergence values for pairwise comparison of diffusion constant posterior distributions. Compared posteriors are generated by scanning through pairings of [d_const, mult*d_const] where mult takes on the range of values provided by mult_list and trajectory lengths. For each pair of diffusion constants, generate a trajectory of length n_steps and find the associated posterior parameter fit, repeating n_reps times to get median parameter values (alpha, beta). Use these median values of alpha and beta to select one posterior diffusion constant distribution for that diffusion constant. Repeat this process for diffusion constant d_const*mult, then calculate the KL divergence of the posteriors for (d_const, d_const*multiplier) and store in dataframe. Repeat for all pairs of (n_steps, multiplier) to fill the dataframe. The results is a heatmap of how distinguishable two diff constants are, conditional upon their relative values and the length of trajectories used.

Parameters

- **n_dim** – number of spatial dimensions
- **d_const** – diffusion constant (um2/s)
- **mult_list** – list of multipliers to get set of d_const2 values, where d_const2 = mult*d_const
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **n_reps** – number of trajectories
- **loc_std** – standard deviation of localization error (um)

Return df dataframe containing the pairwise KL divergences

diffusive_distinguishability.ndim_homogeneous_distinguishability.**generate_posterior**(*n_dim*, *d_const*, *n_steps*, *dt*, *loc_std=0*)

Simulate a single trajectory and find the diffusion constant posterior (inverse gamma) distribution.

Parameters

- **n_dim** – number of spatial dimensions
- **d_const** – diffusion constant (um2/s)
- **n_steps** – trajectory length (number of steps)

- **dt** – timestep size (s)
- **loc_std** – standard deviation for Gaussian localization error (um)

Return alpha, beta scale and shape parameters for inverse gamma posterior for a diffusive trajectory

diffusive_distinguishability.ndim_homogeneous_distinguishability.**get_dim_error**(*n_dim*, *d_const*, *n_steps*, *dt*, *n_reps*, *show_plot*, *loc_std=0*)

Given a diffusion constant, get the posterior for a trajectory of length n_steps and timestep dt. Repeat n_reps times and report/plot hist of the percent error of the mean posterior values vs true diffusivity values.

Parameters

- **n_dim** – number of spatial dimensions
- **d_const** – diffusion constant (um2/s) whose estimator error we want to calculate
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **n_reps** – number of trajectory replicates
- **show_plot** – T/F flag of whether or not to display histograms of estimator errors
- **loc_std** – standard deviation of localization error (um)

Return p_error array of percent error between mean posterior estimation and true value for each run with each

number of dimensions

diffusive_distinguishability.ndim_homogeneous_distinguishability.**get_posterior_set**(*n_dim*, *d_const*, *n_steps*, *dt*, *n_reps*, *loc_std=0*)

Repeat analysis generating a posterior diffusion constant distribution per trajectory for multiple trajectories and return (1) full set and (2) median values of distribution fit parameters.

Parameters

- **n_dim** – number of spatial dimensions
- **d_const** – diffusion constant (um2/s)
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **n_reps** – number of trajectory replicates
- **loc_std** – standard deviation for Gaussian localization error (um)

Return alpha, beta, alpha_std, beta_std, alphas, betas medians, std deviations and arrays of scale and

shape parameters for inverse gamma posteriors for n_reps diffusive trajectories

diffusive_distinguishability.ndim_homogeneous_distinguishability.**get_single_error**(*dim*, *d_const*, *n_steps*, *dt*, *n*, *loc_std*)

Generate single posterior and calculate percent error of posterior mean relative to the true value.

Parameters

- **dim** – number of spatial dimensions
- **d_const** – diffusion constant (um2/s) whose estimator error we want to calculate
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **n** – trajectory number
- **loc_std** – standard deviation of localization error (um)

Returns percent error for a single posterior mean relative to true value

diffusive_distinguishability.ndim_homogeneous_distinguishability.**get_ticks**(*tick_values*, *n_round*, *n_ticks*)

Round tick values and keep only some ticks to improve readability.

Parameters

- **tick_values** – tick values
- **n_round** – number of decimal places to round to
- **n_ticks** – number of ticks to keep

Return ticks list of axis tick values to display

diffusive_distinguishability.ndim_homogeneous_distinguishability.**invgamma_fullparams**(*dist*)

Return the alpha,beta parameterization of the inverse gamma distribution.

Parameters **dist** – scipy inverse gamma distribution

Returns alpha and beta parameters characterizing this inverse gamma distribution

diffusive_distinguishability.ndim_homogeneous_distinguishability.**invgamma_kldiv**(*param1*, *param2*)

Compute KL divergence of two inverse gamma distributions (ref: https://arxiv.org/pdf/1605.01019.pdf).

Parameters

- **param1** – list containing alpha and beta parameters characterizing inverse gamma distribution 1
- **param2** – list containing alpha and beta parameters characterizing inverse gamma distribution 2

Returns KL divergence of two inverse gamma distributions

diffusive_distinguishability.ndim_homogeneous_distinguishability.**plot_df_results**(*df1*, *df2*, *n_round*, *n_ticks*, *size*, *title1*, *title2*, *x_lab*, *y_lab*)

Plot two df heatmaps as two subplots of one figure. They share x and y axis labels but have differing titles.

Parameters

- **df1** – df to visualize
- **df2** – second df to visualize (often log of df1)
- **n_round** – number of axis tick decimal places to round to
- **n_ticks** – number of axis ticks to keep
- **size** – figure size
- **title1** – plot title for left (df1) panel
- **title2** – plot title for left (df2) panel
- **x_lab** – x axis label
- **y_lab** – y axis label

diffusive_distinguishability.ndim_homogeneous_distinguishability.**show_error_hist**(*n_dim*, *p_error*)

Plot figure with 3 subplots, where each subplot is a histogram of the percent errors from all runs in a given number of spatial dimensions.

Parameters

- **n_dim** – number of spatial dimensions
- **p_error** – array of percent error for all runs in each number of spatial dimensions

diffusive_distinguishability.ndim_homogeneous_distinguishability.**simulate_diffusion_df**(*n_dim*, *d_const*, *n_steps*, *dt*, *loc_std*)

Simulate and output a single trajectory of homogeneous diffusion in a specified number of dimensions.

Parameters

- **n_dim** – number of spatial dimensions for simulation (1, 2, or 3)
- **d_const** – diffusion constant (um2/s)
- **n_steps** – trajectory length (number of steps)
- **dt** – timestep size (s)
- **loc_std** – standard deviation for Gaussian localization error (um)

Returns trajectory dataframe (position in n_dim dimensions, at each timepoint)

diffusive_distinguishability.ndim_homogeneous_distinguishability.**trajectory_df_from_data**(*tr*

If you are using experimental rather than simulated trajectories: this is an example function for how you might import your own timelapse trajectory and put into the required dataframe format, compatible with this notebook for analysis. This function will likely require edits for individual use, to make it compatible with your input trajectory format.

Parameters **trajectory** – list or array of spatial positions, where each entry is the position at a single timepoint

(may be 1D, 2D or 3D) :return: dataframe containing trajectory, n-dimensional displacement vectors for each timestep, and step size magnitudes for each timestep

4.1.4 Module contents

Top-level package for diffusive_distinguishability.

diffusive_distinguishability.**get_module_version**()

CHAPTER 5

Allen Institute Contribution Agreement

5.1 Terms

This document describes the terms under which you may make "Contributions" — which may include without limitation, software additions, revisions, bug fixes, configuration changes, documentation, or any other materials — to any of the projects owned or managed by the Allen Institute. If you have questions about these terms, please contact us at terms@alleninstitute.org.

You certify that:

- Your Contributions are either:

1. Created in whole or in part by you and you have the right to submit them under the designated license (described below); or
2. Based upon previous work that, to the best of your knowledge, is covered under an appropriate open source license and you have the right under that license to submit that work with modifications, whether created in whole or in part by you, under the designated license; or
3. Provided directly to you by some other person who certified (1) or (2) and you have not modified them.

- You are granting your Contributions to the Allen Institute under the terms of the [2-Clause BSD license](https://opensource.org/licenses/BSD-2-Clause) (the "designated license").
- You understand and agree that the Allen Institute projects and your Contributions are public and that a record of the Contributions (including all metadata and personal information you submit with them) is maintained indefinitely and may be redistributed consistent with the Allen Institute's mission and the 2-Clause BSD license.

5.1.1 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:

5.2 Types of Contributions

Report bugs at https://github.com/jcass11/diffusive_distinguishability/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.

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

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

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

The best way to send feedback is to file an issue at https://github.com/jcass11/diffusive_distinguishability/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 :)

5.3 Get Started!

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

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

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

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

```
$ mkvirtualenv diffusive_distinguishability
$ cd diffusive_distinguishability/
$ pip install -r requirements_dev.txt
$ pip install -e .
```

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 diffusive_distinguishability
$ make test-all
```

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 "Resolves gh-###. Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.4 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 and 3.7, and for PyPy. Check https://travis-ci.org/jcass11/diffusive_distinguishability/pull_requests and make sure that the tests pass for all supported Python versions.

5.5 Tips

To run a subset of tests:

```
$ py.test tests.test_diffusive_distinguishability
```

5.6 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:

```
$ bumpversion 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

- Julie Cass <juliec@alleninstitute.org>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2019-04-25)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index