GraSPy: Graph Statistics in Python

Jaewon Chung[1, †], Benjamin D. Pedigo[1, †], Eric W. Bridgeford[2], Bijan K.Varjavand[1], Hayden S. Helm[3], and Joshua T. Vogelstein[1, 3, 4, ⋆]

[1]Department of Biomedical Engineering, Johns Hopkins University, Baltimore, MD 21218
[2]Department of Biostatistics, Johns Hopkins University of Public Health, Baltimore, MD 21218
[3]Center for Imaging Science, Johns Hopkins University, Baltimore, MD 21218
[4]Institute for Computational Medicine, Johns Hopkins University, Baltimore, MD 21218
[†]Denotes equal contribution
[⋆]Corresponding author

We introduce `GraSPy`, a Python library devoted to statistical inference, machine learning, and visualization of random graphs and graph populations. This package provides flexible and easy-to-use algorithms for analyzing and understanding graphs with a `scikit-learn` compliant API. `GraSPy` can be downloaded from Python Package Index (PyPi), and is released under the Apache 2.0 open-source license. The documentation and all releases are available at https://neurodata.io/graspy.

1 Introduction

Graphs, or networks, are a mathematical representation of data that consists of discrete objects (nodes or vertices) and relationships between these objects (edges). For example, in a brain, regions of interest can be vertices, the edges represent the presence of a structural connection between them [1]. Since graphs necessarily deal with relationships between nodes, classical statistical assumptions about independence are violated. Thus, novel methodology is required for performing statistical inference on graphs and populations of graphs [2]. While the theory for inference on graphs is highly developed, to date, there has not existed a numerical package implementing these methods. `GraSPy` fills this gap by providing implementations of algorithms with strong statistical guarantees, such as graph and multi-graph embedding methods, two-graph hypothesis testing, and clustering of vertices of graphs. Many of the algorithms implemented in `GraSPy` are flexible and can operate on graphs that are weighted or unweighted, as well as directed or undirected.

2 Library Overview

`GraSPy` includes functionality for fitting and sampling from random graph models, performing dimensionality reduction on graphs or populations of graphs (embedding), testing hypotheses on graphs, and plotting of graphs and embeddings. The following provides brief overview of different modules of `GraSPy`. An example workflow using these modules is shown in Figure 1. More detailed overview and code usage can be found in the tutorial section of `GraSPy` documentation at https://graspy.neurodata.io/tutorial.

Simulations Several random graph models are implemented in `GraSPy`, including the Erdős-Rényi (ER) model, stochastic block model (SBM), degree-corrected Erdős-Rényi (DCER) model, degree-corrected stochastic block model (DCSBM), and random dot product graph

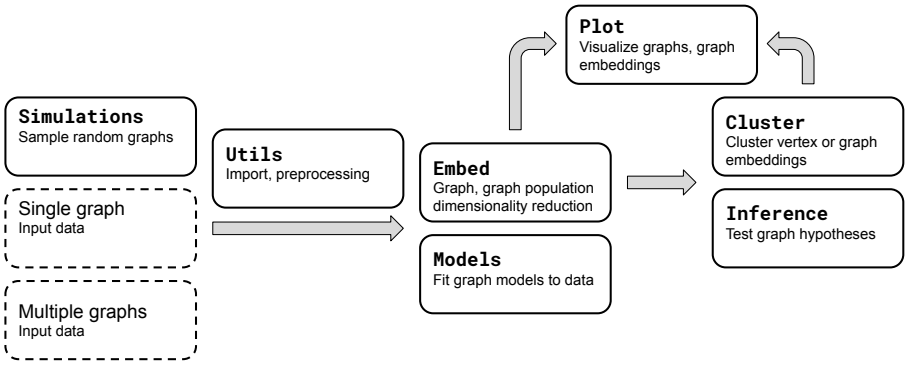

Figure 1: Illustration of modules and procedure for statistical inference on graphs, populations of graphs, or simulated data. A detailed description of each module is given in Section 2.

(RDPG) [3–5]. The simulations module allows the user to sample random graphs given the parameters of one of these models. Additionally, the user can specify a distribution on the weights of graph edges.

Utils `GraSPy` includes a variety of utility functions for graph and graph population importing and preprocessing. Some examples include finding the largest connected component of a graph, finding the intersection or union of connected components across multiple graphs, transforming the weights of a graph, or checking whether a graph is directed.

Embed Inference on random graphs depends on low-dimensional Euclidean representation of the vertices of graphs, known as *latent positions*, typically given by spectral decompositions of adjacency or Laplacian matrices [2]. Adjacency spectral embedding (ASE) and Laplacian spectral embedding (LSE) are methods for embedding a single graph, and omnibus embedding allows for embedding multiple graphs into the same dimensions such that the embeddings can be meaningfully compared [6]. `GraSPy` includes a method for choosing the number of embedding dimensions automatically [7].

Models `GraSPy` includes classes for fitting random graph models to an input graph (Figure 2). Currently, ER, SBM, DCER, DCSBM, and RDPG are supported for model estimation. After fitting a model to data, the model class can also output fit quality metrics (mean squared error, likelihood) and the number of model parameters that were estimated, allowing the user to perform model selection. The model classes can also be used to sample new simulated graphs based on the fit model.

Inference Given two graphs, a natural question to ask is whether these graphs are both random samples from the same generative distribution. `GraSPy` provides two types of test for this null hypothesis: a latent position test and a latent distribution test. Both tests are framed under the RDPG model, where the generative distribution for the graph can be modeled as a set of latent positions. The latent position test can only be performed on two graphs of the same size and with known correspondence between the vertices of the two graphs [9]. The latent distribution test can be performed on graphs without vertex alignment, or even with different numbers of vertices [10].

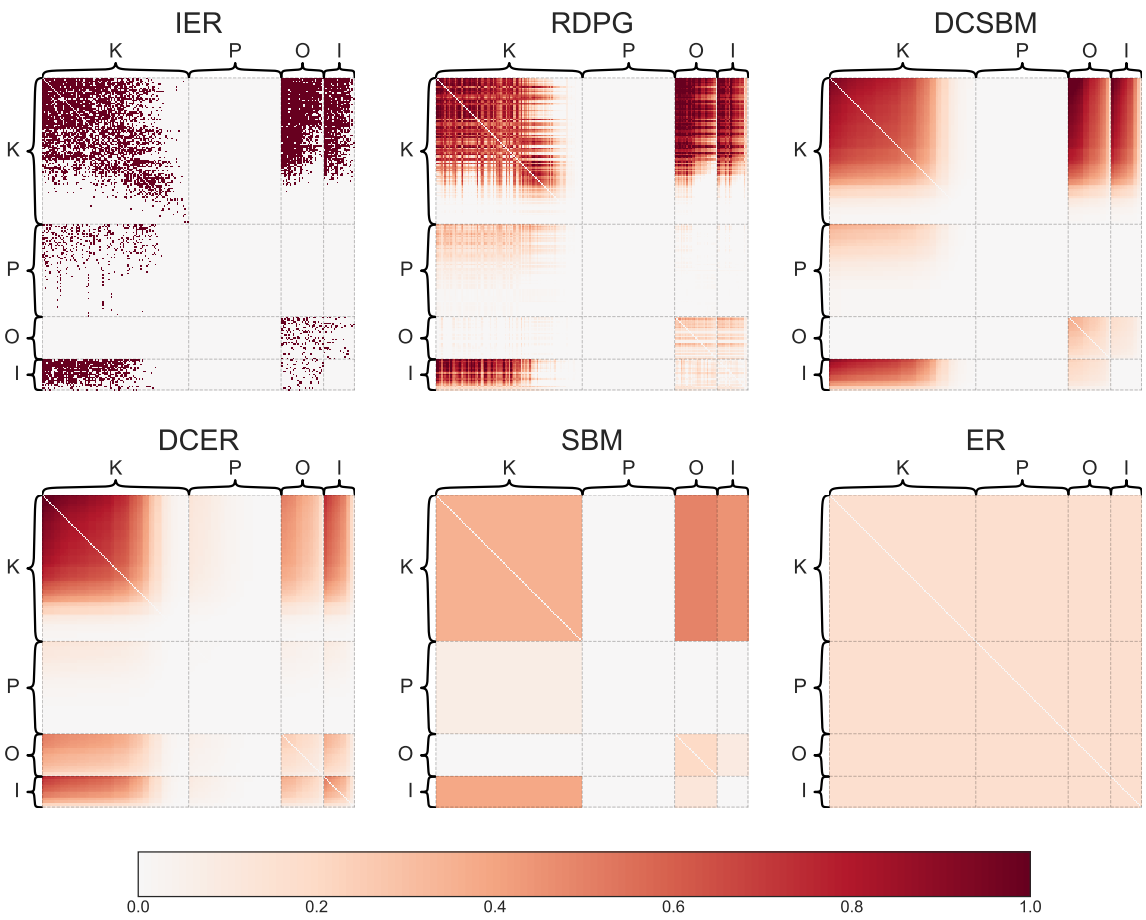

Figure 2: Connectome model fitting using `GraSPy`. Heatmaps show the probability of every potential edge for statistical models of graphs fit to the *Drosophila* larva right mushroom body connectome (unweighted, directed) [8]. The known node labels correspond to cell types: P) projection neurons, O) mushroom body output neurons, I) mushroom body input neurons. The graph models are: inhomogeneous Erdős-Rényi (IER) model in which all potential edges are specified, random dot product graph (RDPG), degree-corrected stochastic block model (DCSBM), degree-corrected Erdős-Rényi (ER), stochastic block model (SBM), and Erdős-Rényi (ER). Blocks (as defined by cell type) are sorted by number of member vertices and nodes are sorted by degree within each block.

Cluster `GraSPy` extends Gaussian mixture models (GMM) and k-means from `scikit-learn` to sweep over a specified range of parameters and choose a clustering that achieves the best performance on some metric [11]. The number of clusters and covariance structure for GMM is chosen by Bayesian information criterion (BIC), which is a penalized likelihood function to evaluate the quality of estimators [12]. Similarly, the silhouette score is used to choose the number of clusters for k-means [13]. In practice, this is often useful for computing the the grouping structure of vertices after embedding.

Plot `GraSPy` extends `seaborn` to visualize graphs as adjacency matrices and embedded graphs as paired scatter plots [14]. Individual graphs can be visualized using `heatmap`

function, and multiple graphs can be overlaid on top of each other using `gridplot` function. Both adjacency matrix visualizations can be sorted by various node metadata. `pairplot` can visualize high dimensional data, such as embeddings, as a pairwise scatter plot.

3 Conclusion

`GraSPy` is the first open-source Python package to perform statistical analysis on graphs and graph populations. Its compliance with the `scikit-learn` API makes it an easy-to-use tool for anyone familiar with machine learning in Python [15]. In addition, `GraSPy` is implemented with an extensible class structure, making it easy to modify and add new algorithms to the package. As `GraSPy` continues to grow and add functionality, we believe it will accelerate statistically principled discovery in any field of study concerned with graphs or populations of graphs.

Bibliography

[1] J. T. Vogelstein, E. W. Bridgeford, B. D. Pedigo, J. Chung, K. Levin, B. Mensh, and C. E. Priebe, "Connectal coding: discovering the structures linking cognitive phenotypes to individual histories," *Current Opinion in Neurobiology*, vol. 55, pp. 199–212, 2019.

[2] A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, Y. Qin, and D. L. Sussman, "Statistical inference on random dot product graphs: a survey," *Journal of Machine Learning Research*, vol. 18, no. 226, pp. 1–92, 2018. [Online]. Available: http://jmlr.org/papers/v18/17-448.html

[3] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[4] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical review E*, vol. 83, no. 1, p. 016107, 2011.

[5] S. J. Young and E. R. Scheinerman, "Random dot product graph models for social networks," in *International Workshop on Algorithms and Models for the Web-Graph*. Springer, 2007, pp. 138–149.

[6] K. Levin, A. Athreya, M. Tang, V. Lyzinski, and C. E. Priebe, "A central limit theorem for an omnibus embedding of multiple random dot product graphs," pp. 964–967, 2017.

[7] M. Zhu and A. Ghodsi, "Automatic dimensionality selection from the scree plot via the use of profile likelihood," *Computational Statistics & Data Analysis*, vol. 51, no. 2, pp. 918–930, 2006.

[8] K. Eichler, F. Li, A. Litwin-Kumar, Y. Park, I. Andrade, C. M. Schneider-Mizell, T. Saumweber, A. Huser, C. Eschbach, B. Gerber *et al.*, "The complete connectome of a learning and memory centre in an insect brain," *Nature*, vol. 548, no. 7666, p. 175, 2017.

[9] M. Tang, A. Athreya, D. L. Sussman, V. Lyzinski, Y. Park, and C. E. Priebe, "A semiparametric two-sample hypothesis testing problem for random graphs," *Journal of Computational and Graphical Statistics*, vol. 26, no. 2, pp. 344–354, 2017.

[10] M. Tang, A. Athreya, D. L. Sussman, V. Lyzinski, and C. E. Priebe, "A nonparametric two-sample hypothesis testing problem for random dot product graphs," *Journal of Computational and Graphical Statistics*, Sep. 2014.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[13] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[14] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, J. Ostblom, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Brunner, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, and A. Qalieh, "mwaskom/seaborn: v0.9.0 (july 2018)," Jul. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1313201

[15] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.