The Journal of Open Source Software

ProbINet: Bridging Usability Gaps in Probabilistic Network Analysis

Diego Baptista[1], **Martina Contisciani**[2], **Caterina De Bacco**[3], **and Jean-Claude Passy**[1]

1 Max Planck Institute for Intelligent Systems, Tübingen, Germany. **2** Central European University, Vienna, Austria. **3** Delft University of Technology, Delft, Netherlands.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Probabilistic **I**nference on **Net**works (ProbINet) is a Python package that provides a unified framework to perform probabilistic inference on networks, enabling researchers and practitioners to analyze and model complex network data. The package integrates code implementations from several scientific publications, supporting tasks such as community detection, anomaly detection, and synthetic data generation using latent variable models. It is designed to simplify the use of cutting-edge techniques in network analysis by providing a cohesive and user-friendly interface. The package includes efficient implementations of probabilistic algorithms, tools for model evaluation, and visualizations to support data exploration.

Statement of need

Network analysis plays a central role in fields such as social sciences, biology, and fraud detection, where understanding relationships between entities is critical. Probabilistic generative models (Contisciani et al., 2020, 2022; Safdari et al., 2021, 2022; Safdari & De Bacco, 2022) have emerged as powerful tools for discovering hidden patterns in networks, detecting communities, identifying anomalies, and generating realistic synthetic data. Despite their potential, the practical use of these models remains challenging due to a lack of integration and accessibility. These methods are often implemented in fragmented codebases spread across individual publications, creating barriers for researchers and practitioners who wish to compare models, reproduce results, or apply them to their own data. ProbINet addresses this critical gap by consolidating recent approaches into a single, unified framework. It provides accessible tools for network analysis tasks, allowing users to explore advanced techniques without the overhead of navigating multiple repositories or inconsistent documentation. By integrating multiple models and workflows, this package promotes reproducibility, simplifies adoption, and enhances usability across disciplines.

Mathematical background

The mathematical foundation of our package builds on recent developments in probabilistic generative models for networks. These models assume that observed network structures arise from underlying latent variables and allow for flexible probabilistic modeling of joint distributions between data and latent variables. By relaxing several restrictive assumptions commonly made in earlier models, our framework supports more expressive methods to uncover hidden structures (e.g., communities and anomalies), model uncertainty, and generate realistic synthetic network data.

The Journal of Open Source Software

Main features

ProbINet offers a versatile and feature-rich framework to perform inference on networks using probabilistic generative models. Its design focuses on integrating diverse models, facilitating parameter selection, providing tools for evaluation and visualization, and enabling synthetic data generation. Key features include:

- **Diverse Network Models**: The package integrates various probabilistic generative models for different network types and analytical goals. The table below summarizes the models implemented in ProbINet:

Algorithm's Name	Description	Network Properties
MTCOV	Extracts overlapping communities in multilayer networks using topology and node attributes (Contisciani et al., 2020).	Weighted, Multilayer, Attributes, Communities
CRep	Models directed networks with communities and reciprocity (Safdari et al., 2021).	Directed, Weighted, Communities, Reciprocity
JointCRep	Captures community structure and reciprocity with a joint edge distribution (Contisciani et al., 2022).	Directed, Communities, Reciprocity
DynCRep	Extends CRep for dynamic networks (Safdari et al., 2022).	Directed, Weighted, Dynamic, Communities, Reciprocity
ACD	Identifies anomalous edges and node community memberships in weighted networks (Safdari & De Bacco, 2022).	Directed, Weighted, Communities, Anomalies

- **Synthetic Network Generation**: ProbINet enables users to generate synthetic networks that closely resemble the characteristics of the real ones. This feature is particularly useful for conducting further analyses on replicated networks, such as testing hypotheses, training algorithms, or exploring network variability.
- **Simplified Parameter Selection and Model Evaluation**: ProbINet includes a cross-validation module to optimize key parameters like the number of communities, providing performance results in a clear and easy-to-interpret dataframe.
- **Rich Set of Metrics for Analysis**: ProbINet includes metrics like F1 scores, Jaccard index, and advanced metrics for link and covariate prediction performance.
- **Powerful Visualization Tools**: ProbINet includes functions to plot community memberships, adjacency matrices, and performance metrics like precision and recall.
- **User-Friendly Command-Line Interface**: ProbINet provides an intuitive command-line interface for specifying models and data paths, fitting models, and outputting inferred parameters, making it accessible to users with minimal Python experience.
- **Modular and Extensible Codebase**: The package is designed with modularity in mind, enabling users to extend its functionality with minimal effort. New models can be easily integrated as long as they follow similar modeling principles, ensuring the framework remains adaptable.

These features are further illustrated in the **Usage** section below with practical examples, showcasing how to apply the package's capabilities to real-world network data.

The Journal of Open Source Software

Usage

Installation

The package can be installed using Python's package manager `pip` or directly from the source repository. Detailed installation instructions are provided in the documentation.

Example: Analyzing a Social Network with ProbINet

In this section, we demonstrate the use of ProbINet to analyze a social network representing friendship relationships among boys in a small high school in Illinois (Coleman, 1964). This network comprises 31 nodes and 100 directed edges, where each node represents a student, and the edges indicate reported friendships between them.

We analyze this network using JointCRep, one of the implemented algorithms in ProbINet, with the aim to infer the latent variables underlying these interactions. Specifically, this model assumes that communities and reciprocity are the main mechanisms for tie formation, a reasonable assumption for friendship relationships.

Steps to Analyze the Network with ProbINet

Using ProbINet, you can:

1. Load your network data as an edge list.
2. Select an appropriate algorithm (e.g., JointCRep) based on your objective.
3. Fit the model to your data and extract inferred latent variables.
4. Analyze the results. For instance, we can investigate the soft community memberships, which reveal how nodes interact with multiple communities through both incoming and outgoing connections.

These steps are exemplified in Figure 1. On the left, a network representation of the input data is displayed alongside the lines of code required for its analysis using ProbINet. The resulting output is shown on the right, where nodes are colored according to their inferred soft community memberships, while edge thickness and color intensity represent the inferred probability of edge existence.

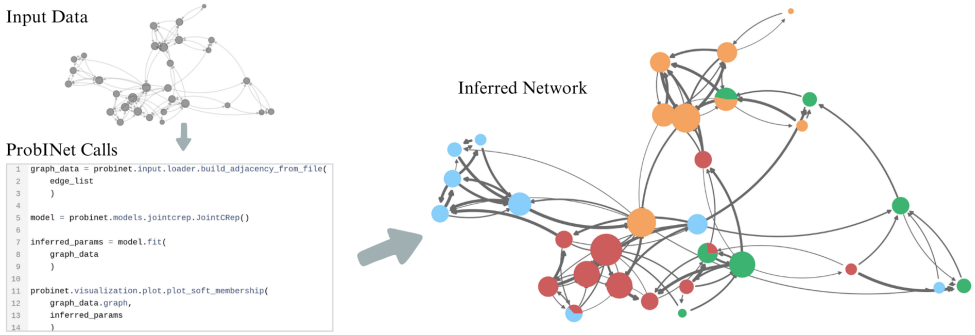

Figure 1: Usage of ProbINet on a social network representing friendship relationships among boys in a small high school in Illinois. (Top-left) A network representation of the input data, consisting of 31 nodes and 100 directed edges. (Bottom-left) A snapshot of the code required for analysis using ProbINet. (Right) The resulting output, where node colors indicate inferred soft community memberships, and edge thickness and color intensity represent the inferred probability of edge existence.

This example illustrates just a few of the various tasks that can be performed with ProbINet. For a more detailed tutorial on this dataset, along with additional use cases, please refer to the package documentation, where we provide numerous examples and guided tutorials.

Running Times of Algorithms

The table below summarizes the running times for ProbINet algorithms when the package is run using the CLI `run_probinet`. **N** and **E** represent the number of nodes and edges, respectively. Edge ranges indicate variation across layers or time steps. **L/T** indicates the number of layers or time steps, and **K** represents the number of communities. The networks used are from the tutorials.

Algorithm	N	E	L/T	K	Time (mean ± std, in seconds)
MTCOV	300	724-1340	4	2	1.51 ± 0.14
CRep	600	5512	1	3	3.00 ± 0.35
JointCRep	250	2512	1	2	3.81 ± 0.69
DynCRep	100	234-274	5	2	1.48 ± 0.06
ACD	500	5459	1	3	27.8 ± 3.2

These benchmarks were performed on a 12th Gen Intel Core i9-12900 CPU with 16 cores and 24 threads, using `hyperfine` and 10 runs. Runs required small amount of RAM (less than 1GB). This table provides a general overview of running times for the algorithms on the default networks. A detailed analysis should be performed on the user's specific data.

Acknowledgements

We extend our gratitude to the contributors of the seminal publications whose work is integrated into this package. We also thank Kibidi Neocosmos, Valkyrie Felso, and Kathy Su for their valuable feedback and suggestions during the development of this package.

References

Coleman, J. S. (1964). Introduction to mathematical sociology. *London Free Press Glencoe*.

Contisciani, M., Power, E. A., & De Bacco, C. (2020). Community detection with node attributes in multilayer networks. *Scientific Reports*, *10*(1), 15736.

Contisciani, M., Safdari, H., & De Bacco, C. (2022). Community detection and reciprocity in networks by jointly modelling pairs of edges. *Journal of Complex Networks*, *10*(4), cnac034.

Safdari, H., Contisciani, M., & De Bacco, C. (2021). Generative model for reciprocity and community detection in networks. *Physical Review Research*, *3*(2), 023209.

Safdari, H., Contisciani, M., & De Bacco, C. (2022). Reciprocity, community detection, and link prediction in dynamic networks. *Journal of Physics: Complexity*, *3*(1), 015010.

Safdari, H., & De Bacco, C. (2022). Anomaly detection and community detection in networks. *Journal of Big Data*, *9*(1), 122.