Bioinformatics, 38(12), 2022, 3310–3311
https://doi.org/10.1093/bioinformatics/btac324
Advance Access Publication Date: 13 May 2022
Applications Note

Systems biology

Nezzle: an interactive and programmable visualization of biological networks in Python

Daewon Lee [1,2]

[1]School of Art and Technology, College of Art and Technology, Chung-Ang University, Anseong, Republic of Korea and [2]Graduate School of Advanced Imaging Sciences, Multimedia, and Film, Chung-Ang University, Seoul, Republic of Korea

Associate Editor: Pier Luigi Martelli

Received on October 7, 2021; revised on March 29, 2022; editorial decision on May 7, 2022; accepted on May 9, 2022

Abstract

Summary: High-quality visualization of biological networks often requires both manual curation for proper alignment and programming to map external data to the graphical components. Nezzle is a network visualization software written in Python, which provides programmable and interactive interfaces for facilitating both manual and automatic curation of the graphical components of networks to create high-quality figures.

Availability and implementation: Nezzle is an open-source project under MIT license and is available from https://github.com/dwgoon/nezzle.

Contact: dwlee@cau.ac.kr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

A variety of real-world phenomena such as biological pathways, communication networks and social relationships can be represented as networks (Barabási, 2016). An insightful visualization of networks facilitates understanding of the analyzed networks and leads to important discoveries. Therefore, open-source software for analyzing and visualizing networks such as GraphViz (Ellson *et al.*, 2001), NetDraw (Borgatti, 2002), Cytoscape (Shannon *et al.*, 2003), Pajek (Batagelj and Mrvar, 2004) and Gephi (Bastian *et al.*, 2009) have been developed, and they have made significant contributions to the field of network science.

As the Python programming language has become a *lingua franca* for scientific computing (Harris *et al.*, 2020), the demand of scientific communities for network visualization in Python has led to the development of the essential libraries and plug-ins. For example, PyGraphviz is a Python interface to the Graphviz, which allows programming the applications of GraphViz in Python (Hagberg *et al.*, 2004). Gephi provides a scripting plug-in for Python scripting based on Jython (Bastian *et al.*, 2009), and GephiStreamer is a third-party Python package that communicates with Gephi through WebSocket and REST API (Totet, 2014). Cytoscape also supports Python programming through CyREST and py4cytoscape (Demchak, 2020; Ono *et al.*, 2015). However, the aforementioned methods have some limitations in terms of interactive graphical user interface (GUI) and seamless programming in Python. PyGraphViz does not provide any interactive GUI. Python programming in Cytoscape and Gephi is inherently indirect, since they are implemented in Java and rely on a client–server communication.

To achieve both interactivity and seamlessness for visualizing biological networks with external data in Python, we have developed a network visualization software named Nezzle (it means **Net** + **Puzzle**, because adjusting nodes and edges of a network for visualization is similar to doing a puzzle). Nezzle provides interactive and programmable interfaces that allow users to adjust the positions of nodes and edges and automate the stylization of graphical components through Python programming.

Figure 1A shows the place of Nezzle in the ecosystem of network visualization in Python. It is located at the position that represents a compromise between interactive GUI and seamless programming in Python. PyGraphViz with Matplotlib (Hunter, 2007) and NetworkX (Hagberg *et al.*, 2008) is one of the successful solutions for automating network visualization. However, users may have a hard time arranging graphical components as this solution lacks an interactive GUI. On the other hand, Nezzle pursues a lightweight and highly customizable software rather than an enterprise-level and all-in-one software that presents an integrated visualization and analysis environment (IVAE) such as Cytoscape and Gephi. Therefore, the GUI of Nezzle is designed to be as minimal as possible, while solving most problems programmatically.

2 Features and use cases

2.1 Design concept

Visualization in Nezzle is designed primarily to be performed through code execution, and the graphical components are edited manually via the network view of GUI only when necessary (Fig. 1B, Supplementary Fig. S1). This is similar to refining the elements of a Matplotlib figure through the GUI after plotting by code execution (Hunter, 2007). One of the important features of Nezzle is seamless programming in Python, which means that any Python module or package can be a plug-in for extending the functionality of Nezzle without inter-process or server–client communications (Supplementary Fig. S2).

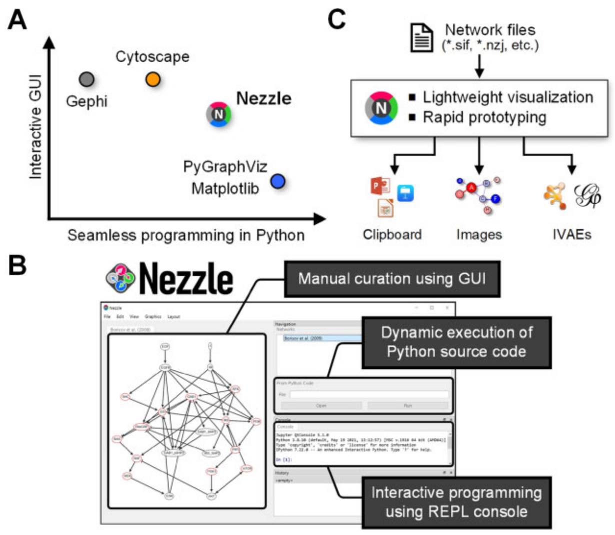
A
Interactive GUI
Seamless programming in Python
Cytoscape
Gephi
Nezzle
PyGraphViz
Matplotlib
B
Nezzle
Manual curation using GUI
Dynamic execution of Python source code
Interactive programming using REPL console
C
Network files (*.sif, *.nzj, etc.)
Lightweight visualization
Rapid prototyping
Clipboard
Images
IVAEs

Fig. 1. Overview of Nezzle. (**A**) Location of Nezzle in the ecosystem of network visualization in terms of interactive GUI and seamless programming in Python. (**B**) Main components of Nezzle GUI. (**C**) Use cases of Nezzle.

2.2 Lightweight visualization

Most scientists today use high-level languages such as Python, R, and Julia rather than Java and C++. With a few changes, scientists can develop their own visualization pipelines based on existing Python codes with Nezzle. In particular, in the case of simple styling automation or visualization of external data through programming, it is more productive to utilize a lightweight software that supports a high-level language like Python (see Supplementary Notes). Nezzle is expected to be an alternative to programming in low-level languages for network visualization.

2.3 Rapid prototyping

Nezzle provides a set of essential features for rapid prototyping to visualize biological networks (Fig. 1C). In general, researchers have to go through a trial-and-error process until they obtain a satisfactory visualization. Users can copy network images from Nezzle and paste them into presentation or word processor software via clipboard to accumulate and compare intermediate visualization results. After confirming the final visualization, users can also export networks as high-quality images or movies for publication.

Nezzle can be a testbed for rapidly evaluating the feasibility of algorithms related to biological networks in Python. For example, users can develop a prototype of network visualization algorithm that is optimized based on a GPU-accelerated deep learning framework such as PyTorch (Paszke *et al.*, 2019). After developing prototype-level algorithms in Nezzle, users may want to implement the algorithms as product-level plug-ins for IVAEs such as Cytoscape or Gephi (Fig. 1C).

3 Conclusion

To achieve both manual curation as well as automatic stylization for high-quality visualization of biological networks in Python, Nezzle provides interfaces for interactive graphics and dynamic code execution. Nezzle enables users to rapidly prototype network visualization while obtaining high-quality images for publication. We expect Nezzle will contribute to advancing the ecosystem of network visualization.

Acknowledgements

We thank the members of Complex Intelligent Systems Laboratory at Chung-Ang University (https://cislab.cau.ac.kr) for their critical reading and valuable comments.

Funding

This research was supported by Chung-Ang University Research Grants in 2020, and by Culture Technology R&D Program 2021 through the Korea Creative Content Agency funded by Ministry of Culture, Sports and Tourism (A Specialist Training of Content R&D based on Virtual Production, R2021040044).

Conflict of Interest: none declared.

References

Barabási,A.-L. (2014) Network science book. *Netw. Sci.*, **625**.
Barabási,A.-L. (2016) *Network Science*. 1st edn. Cambridge University Press, Cambridge, England.
Bastian,M. *et al.* (2009) *Gephi: An Open Source Software for Exploring and Manipulating Networks*. In *Proceedings of the Third International AAAI Conference on Web and Social Media, San Jose, California, USA*, Vol. 3, pp. 361–362.
Batagelj,V. and Mrvar,A. (2004) Pajek—analysis and visualization of large networks. In: *Graph Drawing Software*. Springer, Berlin, Heidelberg, pp. 77–103.
Borgatti,S.P. (2002) *Netdraw software for network visualization*. Analytic Technologies, Lexington, KY, p. 95.
Demchak,B. (2020) *py4cytoscape*. https://github.com/cytoscape/py4cytoscape (27 February 2022, date last accessed).
Ellson,J. *et al.* (2001) Graphviz—open source graph drawing tools. In: *International Symposium on Graph Drawing*. Springer, Berlin, Heidelberg, pp. 483–484.
Hagberg,A. *et al.* (2004) *PyGraphViz*. https://github.com/pygraphviz/pygraphviz (27 February 2022, date last accessed).
Hagberg,A. *et al.* (2008) Exploring network structure, dynamics, and function using networks. *Technical report*. Los Alamos National Lab. (LANL), Los Alamos, NM.
Harris,C.R. *et al.* (2020) Array programming with NumPy. *Nature*, **585**, 357–362.
Hunter,J.D. (2007) Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.*, **9**, 90–95.
Ono,K. *et al.* (2015) CyREST: turbocharging cytoscape access for external tools via a RESTful API. *F1000Research*, **4**, 478.
Paszke,A. *et al.* (2019). PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, *Vancouver, Canada*, pp. 8026–8037.
Shannon,P. *et al.* (2003) Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, **13**, 2498–2504.
Totet,M. (2014) *GephiStreamer*. https://github.com/totetmatt/GephiStreamer (27 February 2022, date last accessed).