The Journal of Open Source Software

spaghetti: spatial network analysis in PySAL

James D. Gaboardi[1], **Sergio Rey**[2], **and Stefanie Lumnitz**[3]

1 Pennsylvania State University **2** Center for Geospatial Sciences, University of California Riverside
3 Directorate of Earth Observation Programs, ESRIN, European Space Agency

DOI: 10.21105/joss.02826

Software

- Review
- Repository
- Archive

Editor: Bruce E. Wilson

Reviewers:

- @martibosch
- @usethedata

Submitted: 30 October 2020
Published: 04 June 2021

Summary

The role spatial networks, such as streets, play on the human experience cannot be overstated. All of our daily activities fall along, or in close proximity to, roads, bike paths, and subway systems to name a few. Therefore, when performing spatial analysis in many cases considering network space, as opposed to Euclidean space, allows for a more precise representation of daily human action and movement patterns. For example, people generally cannot get to work by driving in a straight line directly from their home, but move along paths within networks. To this end, `spaghetti` (**spa**tial **g**raph**s**: n**et**works, **t**opology, & **i**nference), a sub-module embedded in the wider PySAL ecosystem, was developed to address network-centric research questions with a strong focus on spatial analysis (Gaboardi et al., 2018; Rey et al., 2015; Rey & Anselin, 2007).

Through `spaghetti`, first, network objects can be created and analysed from collections of line data by various means including reading in a shapefile or passing in a `geopandas.GeoDataFrame` at which time the line data are assigned network topology. Second, `spaghetti` provides computational tools to support statistical analysis of so-called network-based events along many different types of previously loaded networks. Network based-events or near-network observations are events that happen along spatial networks in our daily lives, i.e., locations of trees along footpaths, biking accidents along roads or locations of coffee shops along streets. As with `spaghetti.Network` objects, `spaghetti.PointPattern` objects can be created from shapefiles, `geopandas.GeoDataFrame` objects or single `libpysal.cg.Point`objects. Near-network observations can then be snapped to nearest network segments enabling the calculation of observation distance matrices. Third, these observation distance matrices can be used both within `spaghetti` to perform clustering analysis or serve as input for other network-centric problems (e.g., optimal routing), or within the wider PySAL ecosystem to perform exploratory spatial analysis with `esda`. Finally, `spaghetti`'s network elements (vertices and arcs) can also be extracted as `geopandas.GeoDataFrame` objects for visualization and integrated into further spatial statistical analysis within PySAL (e.g., `esda`).

Related Work & Statement of Need

The most well-known network analysis package within the Python scientific stack is NetworkX (Hagberg et al., 2008), which can be used for modelling any type of complex network (e.g., social, spatial, etc.). OSMnx (Boeing, 2017) is built on top of NetworkX and queries OpenStreetMap for modelling street networks with resultant network objects returned within a `geopandas.GeoDataFrame` (Jordahl et al., 2021). Another package, `pandana` (Foti et al., 2012), is built on top of `pandas` (McKinney, 2010; Reback et al., 2021) with a focus on shortest path calculation and accessibility measures. Within the realm of Python, the functionality provided by `snkit` (Russell & Koks, 2019) is most comparable to `spaghetti`, though it's main purpose is the processing of raw line data into clean network objects. Outside of Python,

SANET (Okabe et al., 2006) is the most closely related project to `spaghetti`, however, it is not written in Python and provides a GUI plugin for GIS software such as QGIS. Moreover, SANET is not fully open source. While all the libraries above are important for network-based research, `spaghetti` was created and has evolved in line with the Python Spatial Analysis Library ecosystem for the specific purpose of utilizing the functionality of spatial weights in `libpysal` for generating network segment contiguity objects.

Current Functionality

Considering the related projects in the Related Work & Statement of Need section detailed above, `spaghetti` fills a niche for not only the processing of spatial network objects, but also post-processing analysis. In other words, this package can be used to study the network *itself* or provide the foundation for studying network-based phenomena, such as crimes along city streets, all within a fully open-source environment. Considering this, the primary purpose of `spaghetti` is creating network objects: collections of vertices and arcs, and their topological relationships. The creation of a network object is realized through the following general steps:

1. read in line data or create features (regular lattices)
2. generate the network representation
3. extract contiguity weights (if desired) as show in Figure 1
4. identify connected components (if desired)
5. extract graph representation of the network (if desired)

After the creation of a base network object it can be manipulated, analyzed, and utilized as the input for subsequent modelling scenarios. The following are several such examples:

- allocating (snapping) observation point patterns to the network (see Figure 2)
- calculating all neighbor distance matrices
  - point type A to point type A (auto)
  - point type A to point type B (cross)
- utilizing observation counts on network segments and network spatial weights within the Moran's *I* attribute to analyze global spatial autocorrelation (Cliff & Ord, 1981; Rey et al., 2019) as seen in Figure 2
- simulating point patterns that can be used within the *K* function attribute for cluster analysis (O'Sullivan & Unwin, 2010; Okabe & Sugihara, 2012)
- splitting the network into (nearly) uniform segments
- extracting features as `geopandas.GeoDataFrame` objects:
  - network arcs, vertices and point patterns
  - largest/longest components
  - shortest paths
  - minimum/maximum spanning trees

The following two demonstrations show several functionalities mentioned above, including feature creation, network instantiation, network allocation, and feature extraction, along with supplementary plots in Figure 1 and Figure 2.

```
import spaghetti
%matplotlib inline
# generate network
lattice = spaghetti.regular_lattice((0,0,3,3), 2, exterior=True)
```

```
ntw = spaghetti.Network(in_data=lattice)
# extract network elements
vertices_df, arcs_df = spaghetti.element_as_gdf(ntw, vertices=True, arcs=True)
# plot
base_kws = {"figsize":(12, 12), "lw":5, "color":"k", "zorder":0}
base = arcs_df.plot(**base_kws, alpha=.35)
node_kws, edge_kws = {"s":100, "zorder":2}, {"zorder":1}
w_kws = {"edge_kws":edge_kws, "node_kws":node_kws}
ntw.w_network.plot(arcs_df, indexed_on="id", ax=base, **w_kws)
vertices_df.plot(ax=base, fc="r", ec="k", markersize=50, zorder=2)
```

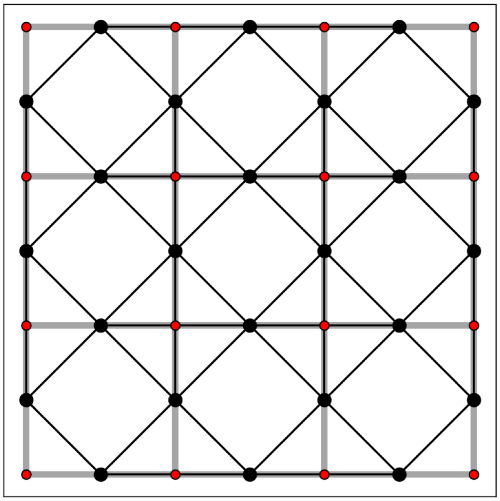

Figure 1: A 4x4 regular lattice with network arcs in gray and vertices in red. Connectivity is demonstrated with `libpysal` spatial weights, which are plotted over the network in black (Rey et al., 2020).

```
import spaghetti, libpysal, matplotlib
# create a network from a line shapefile
ntw = spaghetti.Network(in_data=libpysal.examples.get_path("streets.shp"))
# associate point observations with the network
pp_name = "schools"
pp_shp = libpysal.examples.get_path("%s.shp" % pp_name)
ntw.snapobservations(pp_shp, pp_name, attribute=True)
# calculation global spatial autocorrelation (Moran's I)
moran, yaxis = ntw.Moran(pp_name)
# extract network elements & observations
arcs_df = spaghetti.element_as_gdf(ntw, arcs=True)
schools = spaghetti.element_as_gdf(ntw, pp_name=pp_name)
schools_snapped = spaghetti.element_as_gdf(ntw, pp_name=pp_name, snapped=True)
# plot
base_kws = {"figsize":(7, 7), "lw":3, "color":"k", "zorder":0}
base = arcs_df.plot(**base_kws, alpha=.35)
schools.plot(ax=base, fc="b", ec="k", markersize=100, zorder=1, alpha=.5)
schools_snapped.plot(ax=base, fc="g", ec="k", markersize=50, zorder=2)
matplotlib.pyplot.title(f"Moran's $I$: {round(moran.I, 3)}", size="xx-large")
```

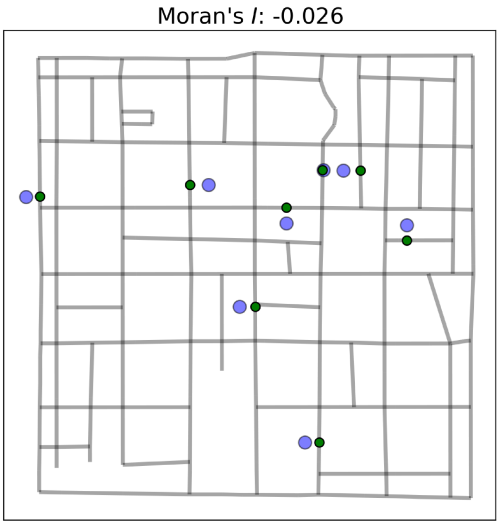

Figure 2: Demonstrating the creation of a network and point pattern from shapefiles, followed by spatial autocorrelation analysis. A shapefile of school locations (blue) is read in and the points are snapped to the nearest network segments (green). A Moran's *I* statistic of -0.026 indicates near complete spatial randomness, though slightly dispersed.

The overview presented here provides a high-level summary of functionality. More detailed examples and applications can be found in the *Tutorials* section of the `spaghetti` documentation.

Planned Enhancements

As with any software project, there are always plans for further improvements and additional functionality. Four such major enhancements are described here. The first addition will likely be network partitioning through use of voronoi diagrams generated in network space. Network-constrained voronoi diagrams can be utilized as tools for analysis in and of themselves and can also be input for further analysis, such as the voronoi extension of the Network *K* function (Okabe & Sugihara, 2012). Second, the current algorithm for allocating observations to a network within `spaghetti` allows for points to be snapped to a single location along the nearest network segment. While this is ideal for concrete observations, such as individual crime incidents, multiple network connections for abstract network events, such as census tract centroids, may be more appropriate (Gaboardi et al., 2020). Third, the core functionality of `spaghetti` is nearly entirely written with pure Python data structures, which are excellent for code readability and initial development but generally suffer in terms of performance. There are currently several functions that can be utilized with an optional `geopandas` installation, however, further integration with the `pandas` stack has the potential to greatly improve performance. Finally, `spaghetti` developers will assess together with PySAL developers how to best support visualization and visual analysis targeted towards `spaghetti` network objects, implemented within visualization packages like `splot` or `mapclassify` and exposed as high level plotting functionality in `spaghetti` (Lumnitz et al., 2020).

Concluding Remarks

Network-constrained spatial analysis is an important facet of scientific inquiry, especially within the social and geographic sciences (Marshall et al., 2018). Being able to perform this type

of spatial analysis with a well-documented and tested open-source software package further facilitates fully reproducible and open science. With these motivations and core values, the `spaghetti` developers and wider PySAL team look forward to creating and supporting research into the future.

Acknowledgements

Firstly, we would like to thank all the contributors to, and users of, this package. We would also like to acknowledge Jay Laura, who was the original lead developer of this package (`pysal.network`) prior to the introduction of the PySAL 2.0 ecosystem. The development of this package was partially supported by the Atlanta Research Data Center and National Science Foundation Award #1825768.

References

Boeing, G. (2017). OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*, *65*, 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004

Cliff, A. D., & Ord, J. K. (1981). *Spatial processes: Models and applications*. Pion.

Foti, F., Waddell, P., & Luxen, D. (2012). A generalized computational framework for accessibility: From the pedestrian to the metropolitan scale. *Proceedings of the 4th TRB Conference on Innovations in Travel Modeling. Transportation Research Board*.

Gaboardi, J. D., Folch, D. C., & Horner, M. W. (2020). Connecting Points to Spatial Networks: Effects on Discrete Optimization Models. *Geographical Analysis*, *52*, 299–322. https://doi.org/10.1111/gean.12211

Gaboardi, J. D., Laura, J., Rey, S., Wolf, L. J., Folch, D. C., Kang, W., Stephens, P., & Schmidt, C. (2018). *Pysal/spaghetti*. https://doi.org/10.5281/zenodo.1343650

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), *Proceedings of the 7th python in science conference (SciPy 2008)* (pp. 11–15).

Jordahl, K., Bossche, J. V. den, Fleischmann, M., McBride, J., Wasserman, J., Gerard, J., Badaracco, A. G., Snow, A. D., Tratner, J., Perry, M., Farmer, C., Hjelle, G. A., Cochran, M., Gillies, S., Culbertson, L., Bartos, M., Caria, G., Eubank, N., sangarshanan, … abonte. (2021). *Geopandas/geopandas: v0.9.0* (Version v0.9.0). Zenodo. https://doi.org/10.5281/zenodo.4569086

Lumnitz, S., Arribas-Bell, D., Cortes, R. X., Gaboardi, J. D., Griess, V., Kang, W., Oshan, T. M., Wolf, L., & Rey, S. (2020). Splot - visual analytics for spatial statistics. *Journal of Open Source Software*, *5*(47), 1–4. https://doi.org/10.21105/joss.01882

Marshall, S., Gil, J., Kropf, K., Tomko, M., & Figueiredo, L. (2018). Street Network Studies: from Networks to Models and their Representations. *Networks and Spatial Economics*, *18*(3), 735–749. https://doi.org/10.1007/s11067-018-9427-9

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

O'Sullivan, D., & Unwin, D. J. (2010). Point pattern analysis. In *Geographic information analysis* (pp. 121–156). John Wiley & Sons, Ltd. https://doi.org/10.1002/9780470549094.ch5

Okabe, A., Okunuki, K., & Shiode, S. (2006). SANET: A Toolbox for Spatial Analysis on a Network. *Geographical Analysis*, *38*, 57–66. https://doi.org/10.1111/j.0016-7363.2005.00674.x

Okabe, A., & Sugihara, K. (2012). *Spatial Analysis Along Networks*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781119967101

Reback, J., McKinney, W., jbrockmendel, Bossche, J. V. den, Augspurger, T., Cloud, P., gfyoung, Hawkins, S., Sinhrks, Roeschke, M., Klein, A., Petersen, T., Tratner, J., She, C., Ayd, W., Naveh, S., Garcia, M., patrick, Schendel, J., … h-vetinari. (2021). *Pandas-dev/pandas: Pandas 1.2.3* (Version v1.2.3). Zenodo. https://doi.org/10.5281/zenodo.4572994

Rey, S., & Anselin, L. (2007). PySAL: A Python Library of Spatial Analytical Methods. *The Review of Regional Studies*, *37*(1), 5–27. https://rrs.scholasticahq.com/article/8285.pdf

Rey, S., Anselin, L., Li, X., Pahle, R., Laura, J., Li, W., & Koschinsky, J. (2015). Open Geospatial Analytics with PySAL. *ISPRS International Journal of Geo-Information*, *4*(2), 815–836. https://doi.org/10.3390/ijgi4020815

Rey, S., Stephens, P., Wolf, L. J., Schmidt, C., jlaura, Oshan, T., Arribas-Bel, D., Gaboardi, J., Folch, D., mhwang4, Kang, W., Malizia, N., Amaral, P., Anselin, L., Knaap, E., Shao, H., Marynia, Winslow, A., Conceptron, … Reagan, A. (2020). *Pysal/libpysal: v4.2.2* (Version v4.2.2). Zenodo. https://doi.org/10.5281/zenodo.1472807

Rey, S., Wolf, L., Kang, W., Stephens, P., Laura, J., Schmidt, C., Arribas-Bel, D., Lumnitz, S., Duque, J. C., Folch, D., Anselin, L., Malizia, N., Gaboardi, J., Fernandes, F., Seth, M., mhwang4, & mlyons-tcc. (2019). *pysal/esda* (Version v2.1.1). Zenodo. https://doi.org/10.5281/zenodo.3265190

Russell, T., & Koks, E. (2019). *tomalrussell/snkit: v1.6.0* (Version v1.6.0). Zenodo. https://doi.org/10.5281/zenodo.3379659