The Journal of Open Source Software

splot - visual analytics for spatial statistics

Stefanie Lumnitz[1, 2], Dani Arribas-Bell[3], Renan X. Cortes[2], James D. Gaboardi[4], Verena Griess[1], Wei Kang[2], Taylor M. Oshan[7], Levi Wolf[5,6], and Sergio Rey[2]

1 Department of Forest Resource Management, University of British Columbia **2** Center for Geospatial Sciences, University of California Riverside **3** Geographic Data Science Lab, Department of Geography & Planning, University of Liverpool **4** Department of Geography, Pennsylvania State University **5** School of Geographical Sciences, University of Bristol **6** Alan Turing Institute **7** Department of Geographical Sciences, University of Maryland, College Park

DOI: 10.21105/joss.01882

Software

- Review
- Repository
- Archive

Editor: Leonardo Uieda

Reviewers:

- @ResidentMario
- @martinfleis

Submitted: 26 October 2019
Published: 22 January 2020

Summary

Geography is an intensely visual domain. Its longstanding dependence on visualization and cartography shows as much, with John Snow's cholera map serving as one of the first instances of geovisual analytics in science (Arribas-Bel, de Graaff, & Rey, 2017; Johnson, 2007), and the perennial presence of maps as statistical displays in seminal works on visualization (Tufte, 2001). As such, the existence and continued focus on maps in geographical analysis demands serious, dedicated attention in scientific computing. However, existing methods in Python, specifically for *statistical* visualization of spatial data, are lacking. General-purpose mapping provided by `geopandas` is not fine-tuned enough for statistical analysis (Jordahl et al., 2019). The more analytically-oriented views offered by `geoplot`, while useful, are limited in their statistical applications (Bilogur, Karve, Marsano, & Fleischmann, 2019). Thus, the need remains for a strong, analytically-oriented toolbox for visual geographical analysis.

This need is heightened by the fact that the collection and generation of geographical data is becoming more pervasive (Arribas-Bel, 2014; Goodchild, 2007). With the proliferation of high-accuracy GPS data, many datasets are now *becoming* spatial datasets; their analysis and visualization increasingly requires explicitly spatial methods that account for the various special structures in geographical data (Anselin & Griffith, 1988). Geographical questions about dependence, endogeneity, heterogeneity, and non-stationarity require special statistical tools to diagnose, and spatial analytic software to visualize (Anselin & Rey, 2014). Further, with the increasing importance of code and computation in geographical curricula (Rey, 2009, 2018; University Consortium of Geographic Information Science, 2019), it has become critical for both pedagogical and research reasons to support geographical analyses with robust visualization tools. To date there are few toolkits for geovisualization developed in the scientific Python stack to fill this need and none for visualization of the process and outcome of spatial analytics. It is this niche that `splot` is designed to fill.

Implemented in Python, `splot` extends both *spatial analytical methods* like that found in the Python Spatial Analysis Library (`PySAL`) and *general purpose visualization* functionality provided by popular packages such as `matplotlib`, in order to simplify visualizing spatial analysis workflows and results. The `splot` package was developed in parallel to the ecosystem of tools to store, manage, and analyze spatial data, which evolved in ways that gave more relevance to integrated command-line oriented environments such as `Jupyter`; and less to disconnected, one-purpose point-and-click tools such as traditional desktop GIS packages. In this context, visual analytics done with `splot` allows for more general scientific workflows via the integration of spatial analytics with the rest of the Python data science ecosystem.

As a visual steering tool, `splot` facilitates analyses and interpretation of results, and streamlines the process of model and method selection for many spatial applications. Our high-level API allows quick access to visualizing popular `PySAL` objects generated through spatial statistical analysis. The `PySAL` ecosystem can hereby be understood as a library, integrating many spatial analytical packages (called *sub-modules*) under one umbrella. These sub-modules range in purpose from exploratory data analysis to explanatory statistical models of spatial relationships. As a separate standing package within the ecosystem, `splot` implements a multitude of views for different spatial analysis workflows to give users the opportunity to assess a problem from different perspectives.

Building on top of our users' feedback, `splot`'s functionality can be accessed in two main ways. First, basic `splot` visualization is exposed as `.plot` methods on objects found in various packages in the `PySAL` ecosystem. Integrating simple `splot` visualizations in other `PySAL` packages ensures that users have the quickest possible access to visualizations. This is especially useful for an instantaneous sanity check to determine if the spatial analysis done in `PySAL` is correct, or if there are any errors present in the data used.

Second, all visualizations can be found and called using a `splot.'PySAL_submodule'` name space, depending on the previously analysed object that needs to be visualized (e.g. `splot.giddy`). Directly calling visualizations through `splot` has the advantage to extend users' spatial analysis workflows with more general cartographic and visual methods in `splot.mapping`. One example of this is a Value-by-Alpha (Roth, Woodruff, & Johnson, 2010) (vba) map, a multivariate choropleth mapping method useful to visualize geographic data with uncertainty or visually compare characteristics of populations with varying sizes. A conventional workflow could look like this: after cleaning and preparing data, a `PySAL` Local Moran object is created that estimates whether crimes tend to cluster around one another or disperse far from one another. In order to assess whether the occurrences of crime in the neighborhood of Columbus, Ohio USA, are clustered (or, *spatially autocorrelated*), Local Indicators of Spatial Autocorrelation (LISA) hot and cold spots, Moran I scatterplots and a choropleth map can quickly be created to provide visual analysis (see fig. 1).

```
from splot.esda import plot_local_autocorrelation
plot_local_autocorrelation(moran_loc, gdf, 'Crime')
plt.show()
```

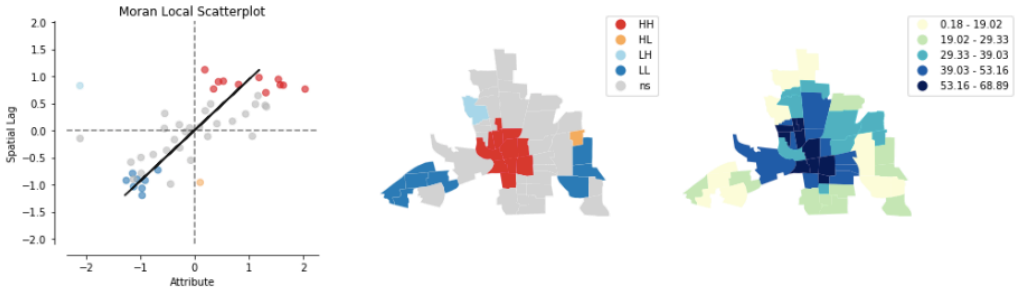

Fig. 1: Local spatial autocorrelation

The user can now further visually assess whether there is dependency between high crime rates (fig. 2, rgb variable) and high income in this neighborhood (fig. 2, alpha variable). Darker shades of the colormap correspond to higher crime and income values, displayed through a static Value-by-Alpha Choropleth using `splot.mapping.vba_choropleth`.

```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
vba_choropleth(x, y, gdf,
               alpha_mapclassify=dict(classifier='quantiles', k=5,
```

```
                rgb_mapclassify=dict(classifier='quantiles', k=5,
                cmap='Blues',
                legend=True, divergent=True, ax=ax)
plt.show()
```

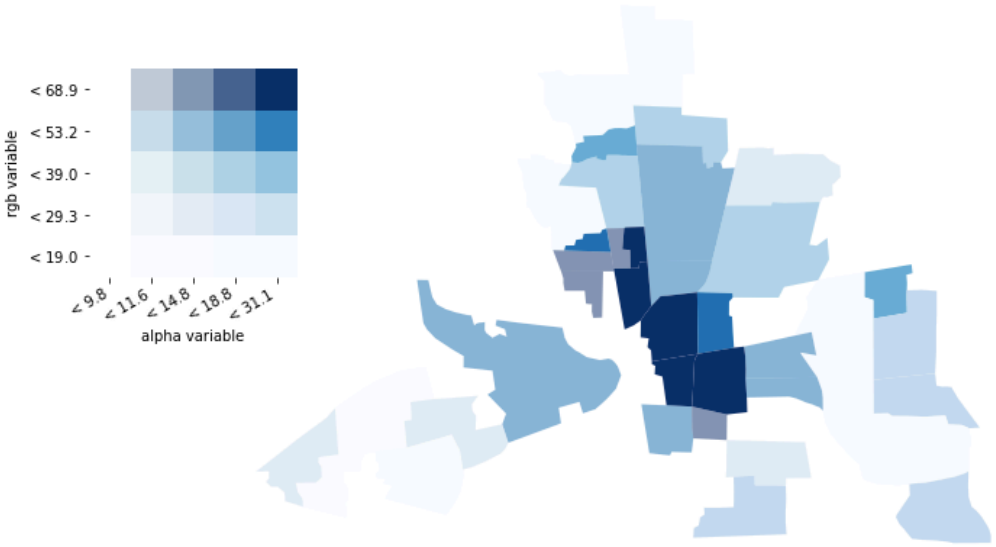

Fig. 2: Value-by-alpha mapping

Ultimately, the `splot` package is designed to facilitate the creation of both static plots ready for publication, and interactive visualizations for quick iteration and spatial data exploration. Although most of `splot` is currently implemented with a `matplotlib` backend, `splot` is framework independent. In that sense, `splot` offers a "grammar" of views that are important and useful in spatial analyses and geographic data science. The `splot` package is not restricted or limited to the current `matplotlib` implementation and can be advanced by integrating emerging or succeeding interactive visualization toolkits, such as `altair` or `bokeh`.

In conclusion, `splot` tightly connects visual analytics with statistical analysis and facilitates the integration of spatial analytics into more general Python workflows through it's compatibility with integrated code-based environments like Jupyter. From spatial autocorrelation analysis to value by alpha choropleths, `splot` is designed as a grammar of views that can be applied to a multitude of spatial analysis workflows. As `splot` developers, we strive to expand `splot`'s grammar of views through new functionality (e.g. in flow mapping methods), as well as provide different backend implementations, including interactive backends, such as `bokeh`, in the future.

Acknowledgements

We acknowledge contributions from, and thank, all our users for reporting bugs, raising issues and suggesting changes to `splot`'s API. Thank you, Joris Van den Bossche and the `geopandas` team for timing releases in accordance with `splot` developments. Thank you, Rebecca Bilbro and Benjamin Bengfort for sharing your insights in how to structure and build API's for visualizations. Thank you Ralf Gommers for guidance on how to design library code for easy maintainability.

References

Anselin, L., & Griffith, D. A. (1988). Do spatial effects really matter in regression analysis? *Papers Regional Science*, *65*, 11–34.

Anselin, L., & Rey, S. J. (2014). *Modern Spatial Econometrics in Practice, a Guide to GeoDa, GeoDaSpace, and PySAL*. Chicago, IL: GeoDa Press.

Arribas-Bel, D. (2014). Accidental, open and everywhere: Emerging data sources for the understanding of cities. *Applied Geography*, *49*, 45–53. doi:10.1016/j.apgeog.2013.09.012

Arribas-Bel, D., de Graaff, T., & Rey, S. J. (2017). Looking at John Snow's Cholera Map from the Twenty First Century: A Practical Primer on Reproducibility and Open Science. In R. Jackson & P. Schaeffer (Eds.), *Regional Research Frontiers - Vol. 2: Methodological Advances, Regional Systems Modeling and Open Sciences*, Advances in Spatial Science (pp. 283–306). Cham: Springer International Publishing. doi:10.1007/978-3-319-50590-9_17

Bilogur, A., Karve, A., Marsano, L., & Fleischmann, M. (2019, October). ResidentMario/geoplot 0.3.3. doi:10.5281/zenodo.3475569

Goodchild, M. F. (2007). Citizen as sensors: The world of volunteered geography. *GeoJournal*, *69*, 211–221.

Johnson, S. (2007). *The Ghost Map: The Story of London's Most Terrifying Epidemic–and How It Changed Science, Cities, and the Modern World* (Reprint edition.). London: Riverhead Books.

Jordahl, K., Bossche, J. V. den, Wasserman, J., McBride, J., Gerard, J., Tratner, J., Perry, M., et al. (2019, July). Geopandas/geopandas: V0.5.1. doi:10.5281/zenodo.3333010

Rey, S. J. (2009). Show me the code: Spatial analysis and open source. *Journal of Geographical Systems*, *11*(2), 191–207. doi:10.1007/s10109-009-0086-8

Rey, S. J. (2018). Code as Text: Open Source Lessons for Geospatial Research and Education. In J.-C. Thill & S. Dragicevic (Eds.), *GeoComputational Analysis and Modeling of Regional Systems*, Advances in Geographic Information Science (pp. 7–21). Cham: Springer International Publishing. doi:10.1007/978-3-319-59511-5_2

Roth, R. E., Woodruff, A. W., & Johnson, Z. F. (2010). Value-by-alpha maps: An alternative to the cartogram. *The Cartographic Journal*, *47*(2), 130–140. doi:10.1179/000870409X124887534553372

Tufte, E. R. (2001). *The visual display of quantitative information*. Graphics Press Cheshire, CT, USA.

University Consortium of Geographic Information Science. (2019). *Geographic Information Science and Technology Body of Knowledge*.