




# TessPy: a python package for geographical tessellation

Siavash Saki <sup>1,2</sup>, Jonas Hamann <sup>1,2</sup>, and Tobias Hagen <sup>1,2</sup>

<sup>1</sup> Frankfurt University of Applied Sciences, Frankfurt am Main, Germany <sup>2</sup> Research Lab for Urban Transport, Frankfurt am Main, Germany

DOI: [10.21105/joss.04620](https://doi.org/10.21105/joss.04620)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Martin Fleischmann](#) 

## Reviewers:

- [@jGaboardi](#)
- [@BenjMy](#)

Submitted: 14 July 2022

Published: 23 August 2022

## License

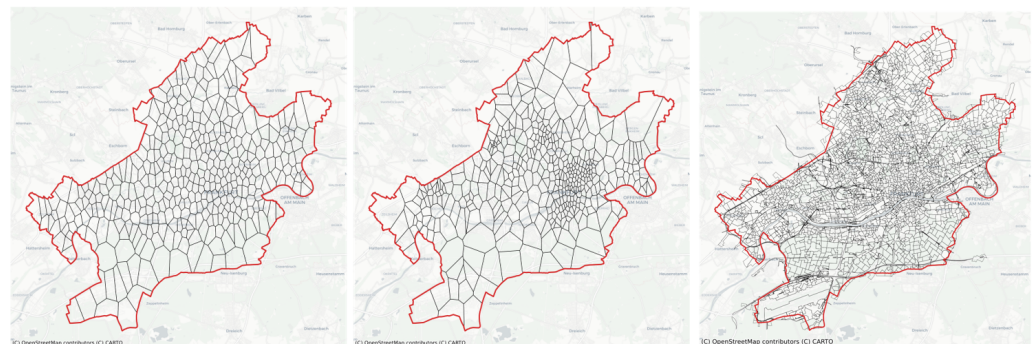
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

Discretization of urban areas is a crucial aspect in spatial analysis. It helps to understand geographical space and provides a framework for analyzing geospatial data ([Gold, 2016](#)). The process of discretization of space into subspaces without overlaps and gaps is called tessellation. The polygons, created using tessellation, are called tiles. Tiles can have the same shape and size (regular) or may differ in shape or size (irregular).

TessPy contains implementations of different tessellation methods for geographical areas and is designed to be flexible and easy to use. It is built on top of GeoPandas ([Jordahl et al., 2019](#)). Tiles are returned in a GeoDataFrame in shapely ([Gillies & others, 2007](#)) Polygon format. Regular methods implemented here are squares (based on Mercantile ([Mercantile Contributors, 2022](#))) and hexagons (based on h3-py ([h3-py Contributors, 2022](#))). The implemented irregular tessellations are adaptive squares, Voronoi diagrams, and city blocks. The irregular methods are data-driven and the required data is retrieved from the OpenStreetMap (OSM) ([OpenStreetMap contributors, 2022](#)). Points of Interest (POI) are downloaded using the overpass API, the official OSM API, to access the OSM database. To get the road network data, osmnx ([Boeing, 2017a](#)) is used (more info at Boeing ([2017b](#))). The input can be in different formats, e.g., a Polygon (or a Multipolygon) with a defined coordinate reference system (CRS), or an address. More methods and functions are explained in the API documentation. Several examples (e.g., clustering of urban areas) are demonstrated in TessPy documentation. Clustering algorithms within the tutorials and the methods use scikit-learn ([Pedregosa et al., 2011](#)) and hdbscan ([McInnes et al., 2017](#)). Further packages for conducting spatial analysis in the tutorials are esda ([S. Rey et al., 2021](#)) and libpysal ([S. Rey, pastephens, et al., 2022](#)), which are subsets of pysal ([S. Rey, Phil, et al., 2022](#)) (More info to be found at S. J. Rey & Anselin ([2007](#)) and S. J. Rey et al. ([2021](#))).

Being extensive and flexible, TessPy allows the user to customize the tessellation methods. This is essential, especially when it comes to irregular methods. For example, different initial clustering algorithms can be used to define centroids in the Voronoi method. [Figure 1](#) shows three examples for irregular tessellations for Frankfurt am Main.



**Figure 1:** Irregular tessellation methods: Voronoi diagrams using k-means (left) and hdbscan (center), and city blocks (right).

## Statement of Need

The first challenge in spatial analysis is defining a (statistical) unit to proceed with quantitative analysis (or, in other words discretizing the studied region). TessPy addresses the process of creating the units for spatial analysis. It uses tessellation to build the tiles, which are then defined as units. TessPy is extensive and flexible at creating these units since various tessellation methods are implemented and all the tessellation methods are customizable, e.g., by providing flexible spatial scaling (Thakur & Fan, 2021). Therefore, it allows for many variations in generating these units.

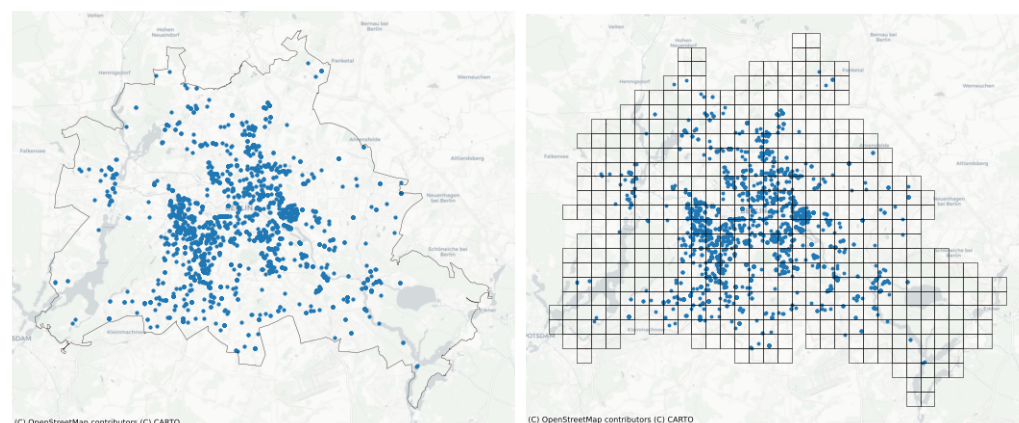
As White & Kiester (2008) argue, topology affects the outcome of geospatial models. Therefore, it is important to consider different tessellation methods while conducting spatial analysis. Regular tessellations are suitable for basic applications or when uniform/congruent tiles are required. For instance, Goovaerts (2000) uses square tessellation to analyze rainfall, and Asamer et al. (2016) uses hexagons tessellation to optimize charging station locations for electric vehicles. The irregular tessellation methods are more sophisticated and complex. The created tiles are flexible and can adapt to the data structure depending on the spatial attribute. For example, Befej & Figurska (2020) use Voronoi tessellation for geospatial analysis of real estate prices.

Regarding the available dataset and the characteristics of the studied variable, researchers in the field of spatial analysis can test and explore different topologies (different tessellation methods or different units) and finally achieve what, in that particular case, leads to the most efficient quantitative analysis. Besides tessellation, TessPy provides further functionalities. For example, users can combine spatial discretization with additional data, in this case POI data are assigned to each tile (Hagen et al., 2022). Overall, TessPy provides the framework for geospatial analysis. By using the retrieved POI data and the created tiles, it allows further analyses.

## Example Usage

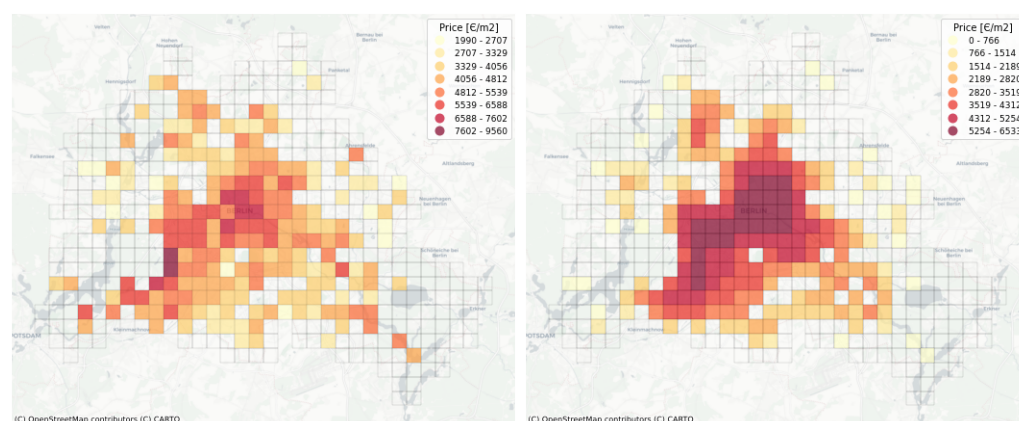
Assume we have a dataset of real estate prices in Berlin. It contains locations and the corresponding prices. Visualizing this dataset (retrieved from ImmoScout24<sup>1</sup>) results in Figure 2 (left), which does not indicate any insight into the real estate prices in Berlin. To start any quantitative analysis, we need first to define the units. We can do this by tessellating Berlin using TessPy, as shown in Figure 2 (right). In this case, we create a square grid. This is the prerequisite for any statistical analysis.

<sup>1</sup><https://www.immobilienscout24.de/>



**Figure 2:** Real estate locations visualized on the map (left), and Berlin tessellated in squares using TessPy (right).

Statistical analyses such as a heatmap of prices [Figure 3](#) (left) or a test for spatial autocorrelation can be conducted based on the created tiles. [Figure 3](#) (right) shows the spatial lag of the price, which is considered a key element in spatial analysis and shows the behavior of the studied variable in the neighbor units (by creating a mean value).



**Figure 3:** A heatmap of real estate prices in Berlin (left), and spatial lag of prices (right) on the basis of the generated tiles.

All the above analyses can be demonstrated using another tessellation method, such as Voronoi polygons, with just a few lines of code. This simplicity of TessPy offers the user the opportunity to easily find the most suitable method to build the quantitative units for further steps.

Using TessPy, the number of different POI categories in each unit can be calculated. For example, a dataset with the number of restaurants in each tile can be created, and by merging this information, we can investigate if there is a correlation between the number of restaurants and real estate prices. Using TessPy, all the POI data can be retrieved to even build a model to explain or predict an outcome variable (such as the price).

## Related Software Packages

Fleischmann (2019) published momepy, a package for urban morphology, with the main purpose of quantitative analysis of urban forms. Using momepy, the user can generate a building-based

morphological tessellation using Voronoi diagrams. This can be used, for example, to calculate which ratio of each tile is covered by a related building. However, this is only a special case of tessellation (Voronoi diagrams and buildings as generators). Moreover, created tiles may not cover the whole defined area since it tessellates the buffered building center points. `momepy` is not developed for geographical tessellation and none of the tessellation approaches of `TessPy` for geographical areas is mentioned by or implemented in `momepy`.

There are further python packages that provide tessellation. For example, `tessagon` (Want, 2022) is a package for the tessellation of 3D surfaces using triangles, hexagons, and more (to use in 3D printers). `Pytess` (Bahgat & Currie, 2022) tessellates space using Delauney triangulation and Voronoi polygons based on points' coordinates. `libpysal` can determine finite Voronoi diagram for a 2-d point set. `NURBS-Python` (`geomdl`) (Bingol & Krishnamurthy, 2019) is a pure python package that offers triangular and quadrilateral tessellations. Tessellation frameworks are not limited to python. The `sf` package (Pebesma, 2018) in R provides a set of tools for working with geospatial vectors and can create a square or hexagonal grid covering the bounding box of a geometry. `spatstat.geom` is a subset of `spatstat` (Baddeley & Turner, 2005), which is another package in R that supports geometrical operations and creates a grid of rectangles for a given spatial region. `bleiglas` (Schmid & Schiffls, 2021) is also an R package that provides tessellation functions. It makes 3D tessellation using point clouds as a mean for data visualization and interpolation. In Julia, the package `VoronoiDelaunay.jl` (Contributors, 2022) provides 2D Delaunay and Voronoi tessellations on generic point types.

A common aspect of most of the above-mentioned packages is their focus on geometry-based tessellation. However, `Tesspy` is distinguished due to its geographical tessellation capabilities. A further unique characteristic of `Tesspy` is creating POI-based tiles and also city blocks.

## Acknowledgements

`TessPy` is the result of the research project `ClusterMobil` conducted by the `Research Lab for Urban Transport`. This research project is funded by the state of Hesse and `HOLM` funding under the *Innovations in Logistics and Mobility* measure of the Hessian Ministry of Economics, Energy, Transport and Housing. [HA Project No.: 1017/21-19]

## References

- Asamer, J., Reinthaler, M., Ruthmair, M., Straub, M., & Puchinger, J. (2016). Optimizing charging station locations for urban taxi providers. *Transportation Research Part A: Policy and Practice*, 85, 233–246. <https://doi.org/10.1016/j.tra.2016.01.014>
- Baddeley, A., & Turner, R. (2005). `spatstat`: An R package for analyzing spatial point patterns. *Journal of Statistical Software*, 12(6), 1–42. <https://doi.org/10.18637/jss.v012.i06>
- Bahgat, K., & Currie, M. (2022). `Pytess`: Pure python tessellation of points into polygons, including delauney/thiessin, and voronoi polygons. In *GitHub*. <https://github.com/karimbahgat/Pytess>
- Belej, M., & Figurska, M. (2020). 3D modeling of discontinuity in the spatial distribution of apartment prices using voronoi diagrams. *Remote Sensing*, 12(2), 229. <https://doi.org/10.3390/rs12020229>
- Bingol, O. R., & Krishnamurthy, A. (2019). `NURBS-Python`: An open-source object-oriented NURBS modeling framework in Python. *SoftwareX*, 9, 85–94. <https://doi.org/10.1016/j.softx.2018.12.005>
- Boeing, G. (2017a). `OSMnx`: A python package to work with graph-theoretic OpenStreetMap street networks. *Journal of Open Source Software*, 2(12). <https://doi.org/10.21105/joss.21105>



00215

- Boeing, G. (2017b). 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>
- Fleischmann, M. (2019). Momepy: Urban morphology measuring toolkit. *Journal of Open Source Software*, 4(43), 1807. <https://doi.org/10.21105/joss.01807>
- Gillies, S., & others. (2007). *Shapely: Manipulation and analysis of geometric objects*. <https://github.com/shapely/shapely>
- Gold, C. (2016). Tessellations in GIS: Part i—putting it all together. *Geo-Spatial Information Science*, 19(1), 9–25. <https://doi.org/10.1080/10095020.2016.1146440>
- Goovaerts, P. (2000). Geostatistical approaches for incorporating elevation into the spatial interpolation of rainfall. *Journal of Hydrology*, 228(1-2), 113–129. [https://doi.org/10.1016/s0022-1694\(00\)00144-x](https://doi.org/10.1016/s0022-1694(00)00144-x)
- h3-py Contributors. (2022). h3-py: Uber’s H3 hexagonal hierarchical geospatial indexing system in python. In *GitHub*. <https://github.com/uber/h3-py>
- Hagen, T., Hamann, J., & Saki, S. (2022). *Discretization of urban areas using POI-based tessellation*. <https://doi.org/10.48718/7jjr-1c66>
- Jordahl, K., Bossche, J. V. den, Wasserman, J., McBride, J., Gerard, J., Fleischmann, M., Tratner, J., Perry, M., Farmer, C., Hjelle, G. A., Gillies, S., Cochran, M., Bartos, M., Culbertson, L., Eubank, N., maxalbert, Rey, S., Bilogur, A., Arribas-Bel, D., ... Greenhall, A. (2019). *Geopandas/geopandas: v0.6.1* (Version v0.6.1) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.3483425>
- McInnes, L., Healy, J., & Astels, S. (2017). Hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11), 205. <https://doi.org/10.21105/joss.00205>
- Mercantile Contributors. (2022). Mercantile: Spherical mercator coordinate and tile utilities. In *GitHub*. <https://github.com/mapbox/mercantile>
- OpenStreetMap contributors. (2022). *OpenStreetMap: Map of the world*. <https://www.openstreetmap.org>
- Pebesma, E. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1), 439–446. <https://doi.org/10.32614/RJ-2018-009>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. In *Journal of Machine Learning Research* (Vol. 12, pp. 2825–2830).
- Rey, S. J., & Anselin, L. (2007). PySAL: A python library of spatial analytical methods. *Review of Regional Studies*, 37, 5–27. <https://doi.org/10.52324/001c.8285>
- Rey, S. J., Anselin, L., Amaral, P., Arribas-Bel, D., Cortes, R. X., Gaboardi, J. D., Kang, W., Knaap, E., Li, Z., Lumnitz, S., & others. (2021). The pysal ecosystem: Philosophy and implementation. *Geographical Analysis*. <https://doi.org/10.1111/gean.12276>
- Rey, S., pastephens, Wolf, L. J., Gaboardi, J., Schmidt, C., jlaura, Arribas-Bel, D., Oshan, T., Folch, D. C., mhwang4, Kang, W., Malizia, N., Amaral, P., Anselin, L., Shekhar, M., Fleischmann, M., Andrade, E. S. de, & knaap, eli. (2022). *Libpysal: Python spatial analysis library core* (Version v4.6.2) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.6326050>
- Rey, S., Phil, Oshan, T., Schmidt, C., jlaura, Wolf, L. J., Arribas-Bel, D., Folch, D. C., mhwang4, Malizia, N., Kang, W., Amaral, P., Gaboardi, J., Anselin, L., knaap, eli,

- Qunshan, Lumnitz, S., Winslow, A., Couwenberg, B., & Robinson, C. (2022). *PySAL: Python spatial analysis library meta-package* (Version v2.7.0) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.6946292>
- Rey, S., Wolf, L. J., Arribas-Bel, D., Gaboardi, J., Kang, W., mhwang4, jlaura, pastepens, Schmidt, C., Lumnitz, S., Folch, D. C., Fleischmann, M., Duque, J. C., Anselin, L., Malizia, N., knaap, eli, Filipe, matthewborish, Morales, L., & Seth, M. (2021). *Estda: Exploratory spatial data analysis in PySAL* (Version v2.4.1) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.5139815>
- Schmid, C., & Schiffels, S. (2021). Bleiglas: An r package for interpolation and visualisation of spatiotemporal data with 3D tessellation. *Journal of Open Source Software*, 6(60), 3092. <https://doi.org/10.21105/joss.03092>
- Thakur, G., & Fan, J. (2021). *MapSpace: POI-based multi-scale global land use modeling*. <https://doi.org/10.25436/E2Z59N>
- VoronoiDelaunay.jl Contributors. (2022). VoronoiDelaunay.jl: Fast and robust voronoi & delaunay tessellation creation with julia. In *GitHub*. <https://github.com/JuliaGeometry/VoronoiDelaunay.jl>
- Want, C. (2022). Tessagon: Tessellation / tiling with python. In *GitHub*. <https://github.com/cwant/tessagon>
- White, D., & Kiester, A. R. (2008). Topology matters: Network topology affects outcomes from community ecology neutral models. *Computers, Environment and Urban Systems*, 32(2), 165–171. <https://doi.org/10.1016/j.compenvurbsys.2007.11.002>