




# GeoHexViz: A Python package for the visualizing hexagonally binned geospatial data

Tony M. Abou Zeidan<sup>1</sup> and Mark Rempel<sup>2</sup>

<sup>1</sup> Canadian Joint Operations Command, Ottawa, Canada, 1600 Star Top Road, K1B 3W6  
<sup>2</sup> Defence Research and Development Canada, Ottawa, Canada, 101 Colonel By Dr., K1A 0K2

DOI: [10.21105/joss.0XXXX](https://doi.org/10.21105/joss.0XXXX)

## Software

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

Editor: [Editor Name](#) 

Submitted: 01 January XXXX

Published: 01 January XXXX

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

Geospatial visualization is often used in military operations research to convey analyses to both analysts and decision makers. For example, it has been used to help commanders coordinate units within a geographic region (Feibush et al., 2000), depict how terrain impacts vehicle performance (Laskey et al., 2010); and to inform training decisions in order to meet mission requirements (Goodrich et al., 2019). When such analyses include a large amount of point-like data, combining geospatial visualization and binning — in particular, hexagonal binning given its properties such as having the same number of neighbours as sides, the centre of each hexagon being equidistant from the centres of its neighbours, and that hexagons tile densely on curved surfaces (Sinha, 2019) — is an effective way to summarize and communicate the data. Recent examples in the military and public safety domains include assessing the impact of infrastructure on Arctic operations (Hunter et al., 2021) and the communicating the distribution COVID-19 cases (Shaito & Elmasri, 2021) respectively.

However, creating such visualizations may be difficult for many since it requires in-depth knowledge of both Geographic Information Systems and analytical techniques, not to mention access to software that may require a paid license, training, and in some cases knowledge of a programming language such as Python or JavaScript. To help reduce these barriers, GeoHexViz — which produces publication-quality geospatial visualizations with hexagonal binning — is a Python package that provides a simple interface, requires minimal in-depth knowledge, and either limited or no programming. This results in an analyst being able to spend more time doing analysis and less time producing visualizations.

GeoHexViz is accessible at **FILL ME IN** and is installed via a `setup.py` script.

## Statement of need

Creating geospatial visualizations is often time-consuming and laborious (Vartak et al., 2014). For example, an analyst must make a variety of design decisions, including which map projection to use, the colour scheme, the basemap, and how to organize the data in layers. An analyst may use one of many software applications may be used to construct a visualization, such as:

- **ArcGIS** (Dangermond & Dangermond, 2021) which provides a wide range of capabilities, but requires a paid license and a solid foundation in geospatial information processing (GISGeography, 2021);
- **QGIS** (Sherman, 2021) which is free and open source, but like ArcGIS requires in-depth knowledge of geospatial information processing to be used effectively (GrindGIS, 2021);

- **D3** (Bostock, 2021) which emphasizes web standards rather than a proprietary framework, but requires extensive knowledge of JavaScript; and
- **Plotly** (Plotly, 2021) which is a free and open source Python graphing library, but like D3 and other packages requires knowledge of a programming language.

Common across these applications is the requirement to have knowledge of geospatial concepts, and acquiring this knowledge has been identified as a significant challenge (Sipe & Dale, 2003). In addition, the latter two options require programming. While many analysts have programming experience, not all do and in time-sensitive situations, as often encountered in a military setting, writing code to produce a visualization may not be feasible. With this in mind, GeoHexViz aims to reduce the time, in-depth knowledge, and programming required to produce publication-quality geospatial visualizations that use hexagonal binning. Implemented in Python, it seamlessly integrates several existing Python packages — Pandas, GeoPandas, Uber H3, Shapely, and Plotly — and extends their functionality to achieve these goals. Although originally designed for use within the military operations research community, it is expected that that GeoHexViz may be of use other communities as well.

## Features

In order to generate a publication-quality geospatial visualization, GeoHexViz requires an analyst to specify a set of *layers* — where each layer is defined as a “[group] of point, line, or area (polygon) features representing a particular class or type of real-world entities” (Caliper, 2021) — to be visualized. At a minimum, an analyst must specify one layer, the *hexbin layer*, through a set of required properties: first, a single reference to the point-like data to be hexagonally binned; and second, references to the data containing the latitude, longitude, and value at each coordinate. If a value at each coordinate is not specified, a value of one is assumed by default. In addition, optional properties may be defined, such as the function to be applied to the values, e.g., count, sum, max, and the resolution of the hexagons. Default values for these properties are provided by GeoHexViz; for example, the default function applied is `sum`.

With this single layer, GeoHexViz generates a publication-quality visualization. However, if the visualization is not satisfactory, GeoHexViz enables an analyst to modify the visualization’s properties. These properties may be categorized into two groups: first, those that use functionality provided by GeoHexViz that both integrate and extend its underlying libraries; and second, those that are passed directly to Plotly without modification. The first group of properties are:

- **scale**: the data displayed in the visualization may be on a linear (default) or logarithmic scale;
- **colour scale**: the colour scale of the visualization may be continuous (default) or discrete;
- **colour scale opacity**: the opacity of the colour scale may be set between opaque (default) to transparent;
- **focus**: the visualization may have no focal point (default), showing a view of the whole Earth, or may be focused on one or more layers; and
- **filter**: all the data may be present in the visualization (default) or may be clipped to a geographic region.

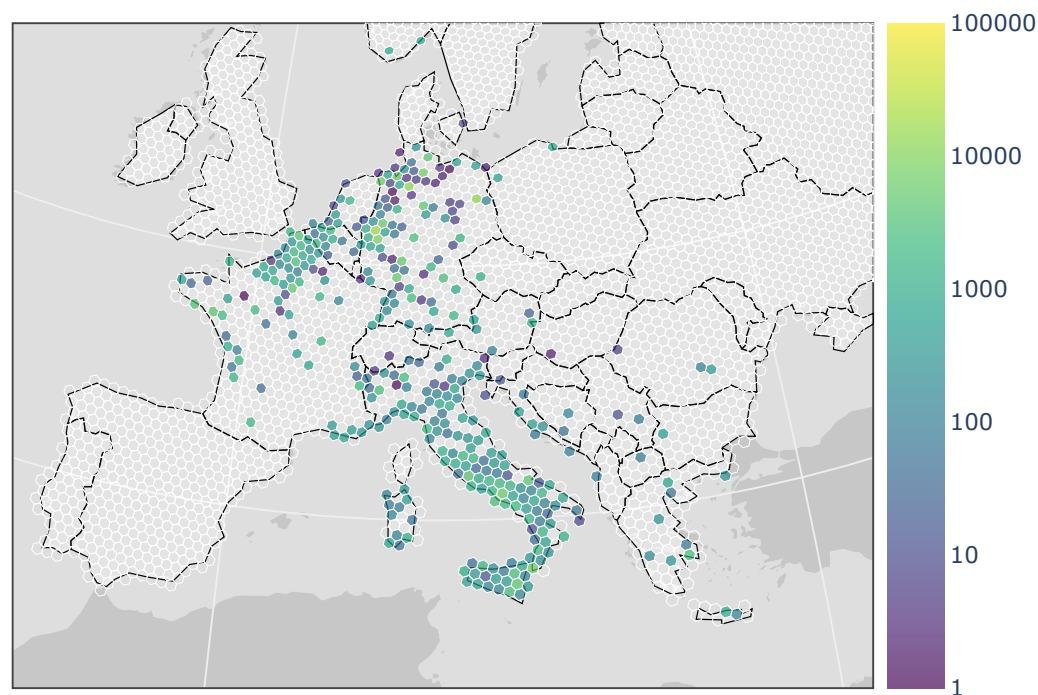
The second group includes a range of properties provided by Plotly, such as border colour, land colour, sea colour, figure size, etc. While default values for these are set by Plotly, many of these defaults are overwritten by GeoHexViz in order to create a publication-quality visualization out of the box.

86 Beyond the *hexbin* layer, an analyst may specify a variable number of optional layers, which  
87 include four types: *regions*, *outlines*, *point*, and *grid*. *Region* layers are plotted as filled  
88 polygons via Plotly Choropleth traces. *Outline* layers behave similarly to *region* layers; however,  
89 they are plotted as empty polygons via Plotly Scattergeo traces instead. *Point* layers enable an  
90 analyst to display additional point-like data, such as cities, on top of the hexagonally binned  
91 data. This layer is plotted via Plotly Scattergeo traces. In situations in which the data to be  
92 hexagonally binned does not cover the entire area of interest, *grid* layers, which are empty  
93 hexagons, may be specified to form a continuous grid of hexagons. Similar to the *hexbin* layer,  
94 each optional layer has its own set of properties, some of which are required in order to define  
95 a layer and others which are optional.

96 GeoHexViz enables an analyst to create a visualization in two ways. First, an analyst may use  
97 GeoHexViz's command-line script `GeoHexSimple` to read a JSON file that specifies the layers'  
98 properties for the visualization. Second, an analyst may generate a visualization by writing a  
99 Python script that imports GeoHexViz's Python module and invokes its functions. In either  
100 case, the data to be hexagonally binned may be provided in a variety of formats, including  
101 Shapefile and CSV. In addition, when writing a Python script the data may be provided as a  
102 `DataFrame` (McKinney, 2021) or `GeoDataFrame` (Jordahl, 2021). The visualization may be  
103 saved in a variety of formats, including PDF, PNG, JPEG, WEBP, SVG, and EPS formats.

## 104 **Example: Aerial bombings in World War 2**

105 Allied aerial bombing in World War 2 occurred across a vast geographic region, with the focus  
106 shifting as the war progressed. In this example, a data set compiled by Lt Col Jenns Robertson  
107 of the United States Air Force and posted on Kaggle (Robertson, 2017) is used in conjunction  
108 with GeoHexViz to depict how this focus in Europe shifted over time. The `examples/ww2_b`  
109 `ombings` directory in the GeoHexViz repository contains a JSON file `json_structure.json`  
110 that, in combination with command-line script, creates a visualization for either 1943, 1944,  
111 or 1945 by setting the path to the corresponding CSV file. For example, the total mass of  
112 bombs dropped by Allied forces in 1943 is depicted in Figure 1. Visualizations for 1944 and  
113 1945 are provided in the directory, as well `json_walkthrough.md` which explains the contents  
114 of the JSON file.



**Figure 1:** Allied aerial bombings in 1943

115 The directory also includes the corresponding Python script `python_walkthrough.py` that  
116 creates these visualizations, and a Jupyter Notebook `python_walkthrough.ipynb` that  
117 explains the Python script.

## 118 Limitations

119 GeoHexViz uses the GeoJSON format to plot data sets. With GeoJSON comes difficulties  
120 when geometries cross the 180th meridian they may be interpreted as wrapping around the  
121 globe. (MacWright, n.d.). In GeoHexViz, hexagonal geometries are supplied via Uber H3  
122 (Smith et al., 2020), and as such hence this issue has been discussed with the its developers  
123 (Abou Zeidan, 2021b). GeoHexViz provides a simple solution to address this problem; it tracks  
124 geometries that cross the meridian, and shifts their coordinates making all of the coordinates  
125 either positive or negative as previously proposed (MacWright, n.d.). However, it should be  
126 noted that when hexagons contain either the North or South Pole, the 180th meridian issue  
127 persists resulting in what appears to be a colour bleeding throughout the visualization and  
128 leaving a hexagon (or hexagons) empty.

129 A second issue issue related to the positioning and height of the colour bar with respect to  
130 the plot area exists. When the dimensions of the plot area are not within a specific range of  
131 aspect ratios, the colour bar position and height may not be optimal. This issue has been  
132 raised with the Plotly development team (Abou Zeidan, 2021a). As this is an issue with the  
133 Plotly itself, the library's developers have indicated that a calculation of plot area dimensions  
134 may be available in the future which would address in this issue.

135 GeoHexViz relies on the Python binding of the Uber H3 package in order to generate hexagons  
136 over polygons. This is done by passing the GeoJSON format of the polygon(s) to Uber H3.  
137 In some cases over large areas, grids may not generate properly resulting in no hexagons, or  
138 multiple invalid hexagons, being retrieved from Uber H3. This issue does not seem to be  
139 widely discussed. This tends to only be an issue for data sets that span large areas.

## Acknowledgements

Thank you to Nicholi Shiell for his input in testing, and providing advice for the development of this package and of its supporting documents.

## References

- Abou Zeidan, T. (2021a). *[feature request | bug report] plot area / colorbar size variance (geos)*. <https://github.com/plotly/plotly.py/issues/3288>
- Abou Zeidan, T. (2021b). *Q: Invalidity of polygons in GeoJSON, GeoPandas*. <https://github.com/uber/h3-py/issues/187>
- Bostock, M. (2021). *D3.js - data-driven documents*. <https://d3js.org/>
- Caliper. (2021). *What is a layer?* Caliper Mapping; Transportation Glossary. <https://www.caliper.com/glossary/what-is-a-map-layer.htm>
- Dangermond, J., & Dangermond, L. (2021). *ArcGIS online*. <https://www.arcgis.com/>
- Feibush, E., Gagvani, N., & Williams, D. (2000). Visualization for situational awareness. *IEEE Computer Graphics and Applications*, 20(5), 38–45. <https://doi.org/10.1109/38.865878>
- GISGeography. (2021). *ArcGIS review: Is ArcMap the best GIS software?* <https://gisgeography.com/esri-arcgis-software-review-guide/>
- Goodrich, D. C., Heilman, P., Guertin, D., Levick, L. R., Burns, I., Armendariz, G., & Wei, H. (2019). *Automated geospatial watershed assessment (AGWA) to aid in sustaining military mission and training*. USDA-ARS Southwest Watershed Research Center (SWRC) Tucson United States.
- GrindGIS. (2021). *Pros and cons of QGIS*. <https://grindgis.com/software/pros-and-cons-of-qgis>
- Hunter, G., Chan, J., & Rempel, M. (2021). *Assessing the impact of infrastructure on arctic operations* (Scientific Report DRDC-RDDC-2021-R024). Defence Research; Development Canada.
- Jordahl, K. (2021). *GeoPandas (0.9.0)*. <https://geopandas.org/index.html>
- Laskey, K. B., Wright, E. J., & Paulo C.G., da C. (2010). Envisioning uncertainty in geospatial information. *International Journal of Approximate Reasoning*, 51(2), 209–223. <https://doi.org/https://doi.org/10.1016/j.ijar.2009.05.011>
- MacWright, T. (n.d.). The 180th meridian. In *macwright.com*. <https://macwright.com/2016/09/26/the-180th-meridian.html>
- McKinney, W. (2021). *Pandas.DataFrame*. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html>
- Plotly. (2021). *Plotly python open source graphing library*. Plotly. <https://plotly.com/python/>
- Robertson, L. C. J. (2017). *Aerial bombing operations in world war II*. <https://www.kaggle.com/usaf/world-war-ii>
- Shaito, M., & Elmasri, R. (2021). Map visualization using spatial and spatio-temporal data: Application to COVID-19 data. *The 14th PErvasive Technologies Related to Assistive Environments Conference*, 284–291. <https://doi.org/10.1145/3453892.3461336>
- Sherman, G. (2021). *QGIS - a free and open source geographic information system*. <https://qgis.org/en/site/>

- 181 Sinha, A. (2019). *Spatial modelling tidbits: Honeycomb or fishnets?* Towards Data Science.  
182 <https://towardsdatascience.com/spatial-modelling-tidbits-honeycomb-or-fishnets-7f0b19273aab>
- 183 Sipe, N., & Dale, P. (2003). Challenges in using geographic information systems (GIS) to  
184 understand and control malaria in indonesia. *Malaria Journal*, 4(1). [https://doi.org/10.](https://doi.org/10.1186/1475-2875-2-36)  
185 [1186/1475-2875-2-36](https://doi.org/10.1186/1475-2875-2-36)
- 186 Smith, A. M., Thaney, K., & Hahnel, M. (2020). H3: A hexagonal hierarchical geospatial  
187 indexing system. In *GitHub repository*. GitHub. <https://github.com/uber/h3>
- 188 Smith, A. M., Thaney, K., & Hahnel, M. (2020). H3: A hexagonal hierarchical geospatial  
189 indexing system. In *GitHub repository*. GitHub. <https://github.com/uber/h3>
- 190 Vartak, M., Madden, S., Parameswaran, A., & Polyzotis, N. (2014). *SeeDB: Automatically*  
191 *generating query visualizations*. 7(13), 1581--1584. [https://doi.org/10.14778/2733004.](https://doi.org/10.14778/2733004.2733035)  
192 [2733035](https://doi.org/10.14778/2733004.2733035)

DRAFT