Notebook-based Visual Analysis of Large Tracking Datasets

Demo paper

Anita Graser
AIT Austrian Institute of Technology, Vienna, Austria
University of Salzburg, Salzburg, Austria
anita.graser@ait.ac.at

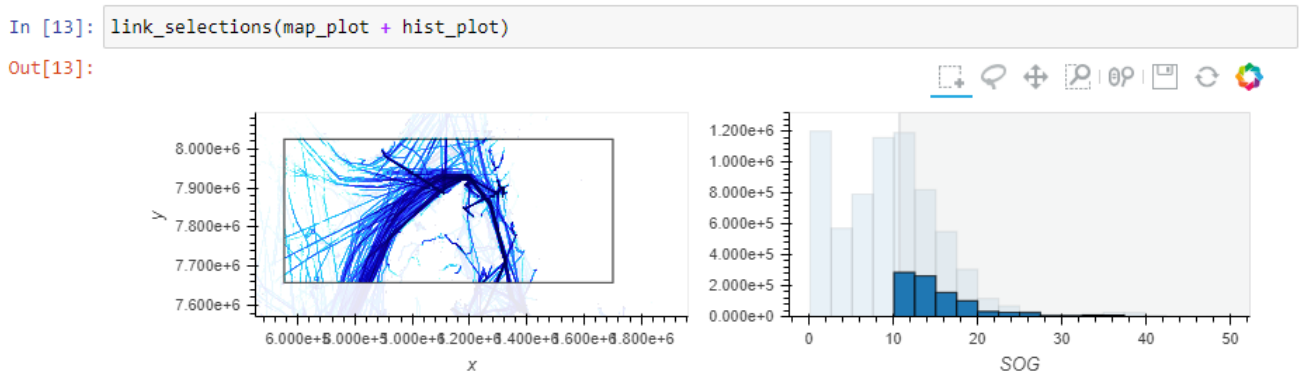

Figure 1: Linked interactive point density map and histogram plots visualizing 12 million GPS location records in a Jupyter notebook cell, with spatial filter by bounding box and attribute filter by speed (SOG).

ABSTRACT

This paper demonstrates some of the latest developments for the visual analysis of large tracking datasets within the Python ecosystem revolving around the Pandas and HoloViz libraries. These software tools enable the quick and interactive notebook-based exploration of large datasets with millions of location records on commodity hardware.

1 INTRODUCTION

Movement datasets collected by systems tracking vehicles, people, goods or wildlife have the potential to improve our understanding of complex mobility systems [8]. Movement data analysis involves multiple interconnected steps, including collection, cleaning, (pre)processing, and data mining which often go through multiple iterations. Conceptually, movement analysis tasks can focus on one of three fundamental aspects of movement [2]: objects/movers (focusing on the moving objects and their trajectories), space (focusing on locations), and time (focusing on linear or cyclic time units).

Visualizations of movement data often suffer from visual clutter [1]. Patterns that may be visible in one type of visualization (with a specific set of parameter settings) can be completely obscured in another visualization or with different parameter settings. For an efficient data exploration workflow, analysts therefore need to be able to explore multiple different visualization types and to vary the corresponding parameter settings [7]. This makes notebook-style interfaces attractive since they enable analysts to track "the steps in a data analysis workflow in a narrative way, for reporting, and for collaboration"[14].

Recent years have seen an increase in movement data analysis functionality in Python, including libraries such as MovingPandas [5, 6] or scikit-mobility [13]. However, these libraries suffer from poor rendering speed that makes them unsuitable for visualizing large datasets [7]. Of course, besides Python, the other large data analysis languae is R and there are dozens of R libraries for movement data [11] which may be used in R notebooks[1]. However, we are not aware of any R notebook examples for movement data analysis.

In this paper, we demonstrate cutting-edge tools for the interactive exploration of large tracking datasets in Python notebooks building on the data handling capabilities of Pandas[12, 15] and the visualization capabilities of HoloViz[2] family of tools. Amongst others, this family of tools includes[3]:

- **hvPlot**[10]: creates interactive HoloViews, GeoViews or Panel objects from Pandas, Xarray, or other data structures
- **HoloViews**[9]: builds interactive Bokeh[4] plots from high-level specifications
- **Datashader**[4]: rasterizes huge datasets quickly as fixed-size images

The key tool for the visualization of large datasets in the following examples is HoloViz *Datashader*. Computation-intensive steps in the Datashader process are written in Python that is compiled to machine code using Numba[5] to increase performance. Furthermore, Datashader distributes the computations across CPU cores and processors (using Dask[6]) or GPUs (using CUDA[7]). Conventional visualization packages for interactive plotting used

[1] https://blog.rstudio.com/2016/10/05/r-notebooks/
[2] http://holoviz.org
[3] https://holoviz.org/background.html
[4] https://bokeh.org
[5] https://numba.pydata.org
[6] https://dask.org
[7] https://developer.nvidia.com/cuda-zone

in notebook environments (for example, Folium[8] used by scikit-mobility) pass all data records directly to the browser where they are then displayed. This enables interaction with each data point but it quickly runs into limitations on how much data can be visualized. Datashader instead generates a fixed-size (rasterized) data structure (regardless of the original number of records) and transfers that to the browser for display.

In the following section we illustrate the capabilities of the HoloViz family of tools for the purposes of movement data exploration with a focus on the moving object trajectories using a dataset of vessel tracking data (AIS) published by the Danish Maritime Authority with more than 12 million location records. Afterwards, we discuss current limitations that movement data analysts should be aware of and have an outlook at what might come next.

2 ANALYSIS EXAMPLES

The analysis examples covered in this demo are primarily mover-focused. The first examples show point-based visualisations of individual location records, as well as segment-based visualizations of consecutive records in their geographic context. Later examples show non-spatial visualizations that can provide further insight into other important characteristics of tracking datasets, such as sampling intervals and mover identification. The data requirements for these examples are minimal: columns for mover ID (*MMSI* in our AIS data example), timestamp (*Timestamp*), location (*Longitude* and *Latitude*), and optionally speed (*SOG* which is short for 'speed over ground' used in the histogram in Figure 1).

2.1 Individual records

Classic scatter plots, as shown in Figure 1, are popular tools (particularly for early steps in the movement data exploration process) to assess the spatial extent of the dataset, find outliers, and identify gaps in the data coverage. This commonly reoccurring task should therefore be straightforward to support data analysts in their work. Indeed, as Listing 1 shows, creating a scatterplot from a Pandas DataFrame and linking it to another plot (a histogram of speed values in this case) requires only a few lines of code[9]. To plot the locations on top of background map tiles, the original latitude and longitude values need to be reprojected to a metric coordinate system (Pseudo-Mercator, EPSG:3857). For this purpose, Datashader provides the convenience function *lnglat_to_meters*.

The linked functionality is automatically created by calling *link_selections*. The linking is based on the common IDs of records in the DataFrame *df* that is visualized in both plots. Another noteworthy feature is the simple way of arranging plots. For example, arranging plots side-by-side in one notebook cell output is achieved by connecting them using a simple plus sign (+). To add geographic context, background map tiles can be added as shown in the following example in Listing 2. The different layers of the map plot are combined using simple multiplication operators (*).

Figure 1 also illustrates how the user can apply spatial filters (by drawing a bounding box in the map view) and attribute-based filters (by selecting certain bin ranges in the histogram). This way, location records of vessels travelling faster than 10 knots are highlighted in the map.

Listing 1: Source code for the linked scatter plot and histogram in Figure 1

```
import pandas as pd
import hvplot.pandas
from datashader.utils import lnglat_to_meters
from holoviews.selection import link_selections

df = pd.read_csv('E:/AISDK/aisdk_20170701.csv',
    usecols=['MMSI', 'Timestamp', 'Latitude',
             'Longitude', 'SOG'])
df.loc[:, 'x'], df.loc[:, 'y'] = lnglat_to_meters(
    df.Longitude, df.Latitude)

map_plot = df.hvplot.scatter(
    x='x', y='y', datashade=True)
hist_plot = df.hvplot.hist('SOG')
link_selections(map_plot + hist_plot)
```

2.2 Track segments

Segment-based visualizations (Figure 2-3) enable more advance analysis tasks, such as the identification of gaps in trajectories or discovery of large jumps and other common movement data quality issues [3]. HoloViews provides a quick solution for plotting segments between consecutive location records. In contrast to other libraries the HoloViews solution does not create expensive spatial geometry objects (such as Shapely[10] LineStrings used by MovingPandas). Instead, it relies on efficient implementations of HoloViews Path objects. Analysts only have to take care of creating distinct paths for each mover by setting a timestamp-based index (to ensure the correct temporal order) and grouping the Pandas DataFrame by mover ID, as shown in Listing 2.

```
In [44]: plot = datashade(path).opts(frame_height=300, frame_widt
         BG_TILES * plot
```

Out[44]:

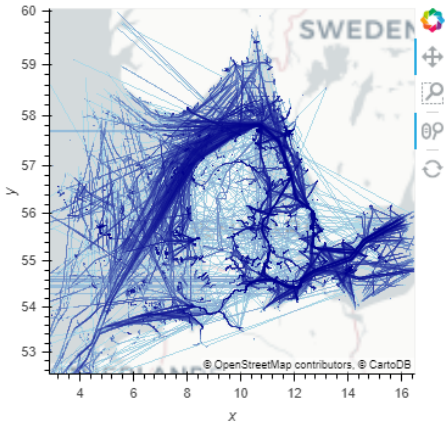

Figure 2: Datashaded density map of track segments.

[8] https://python-visualization.github.io/folium/

[9] Linked brushing demo notebook at https://github.com/anitagraser/movingpandas-examples/blob/bmda2021/tech-demos/linked-brushing.ipynb

[10] https://shapely.readthedocs.io/en/stable/manual.html

Listing 2: Source code for the trajectory plot in Figure 2

```
import holoviews as hv
from holoviews.element import tiles
from holoviews.operation.datashader
    import datashade

df['Timestamp'] = pd.to_datetime(
    df['Timestamp'], format='%d/%m/%Y␣%H:%M:%S')
df.set_index('Timestamp', inplace=True)
grouped = [dfx[['x','y']] for name, dfx in
            df.groupby(['MMSI'])]
path = hv.Path(grouped, kdims=['x','y'])

BG_TILES = tiles.CartoLight()
plot = datashade(path)
BG_TILES * plot
```

For more advanced segment-based analysis, for example, to filter segments based on their length (to detect gaps and jumps), as shown in Figure 3, additional computing steps need to be added. This step (using a custom *compute_segment_info* function in the notebook[11]) is currently considerably more computationally expensive than the plotting steps because of the necessary pair-wise operations such as distance computations between consecutive records of the same mover.

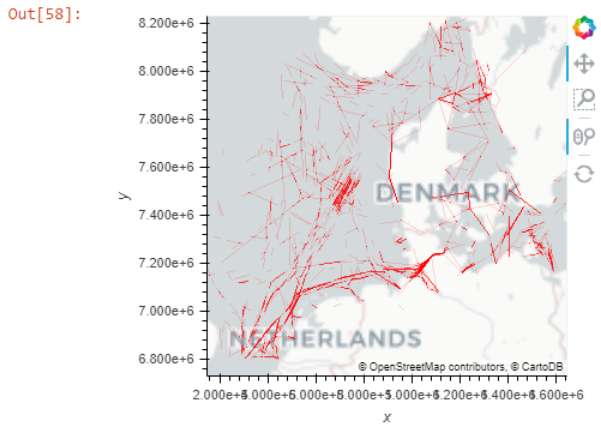

Figure 3: Datashaded density map of gaps in trajectories based on segment lengths between 10 and 100km.

As the previously presented point and segment density map examples show, Datashader is efficient at visualizing large sets of points or paths in two-dimensional plots by rasterizing the inputs. However, Datashader is of course not limited to plots in geographic space. For example, Figure 4 shows a coordinate change plot which visualizes the location change between consecutive location records. These coordinate change plots can help identify issues related to systematic changes in sampling intervals [3] which can be caused, for example, by resampling strategies that discard locations that are too close to the previous reported location. Like the trajectory gap plot (Figure 3), this coordinate change plot requires that the differences in x and y values are computed beforehand.

[11] Trajectory exploration notebook at https://github.com/anitagraser/movingpandas-examples/blob/bmda2021/analysis-examples/5-exploration-protocol.ipynb

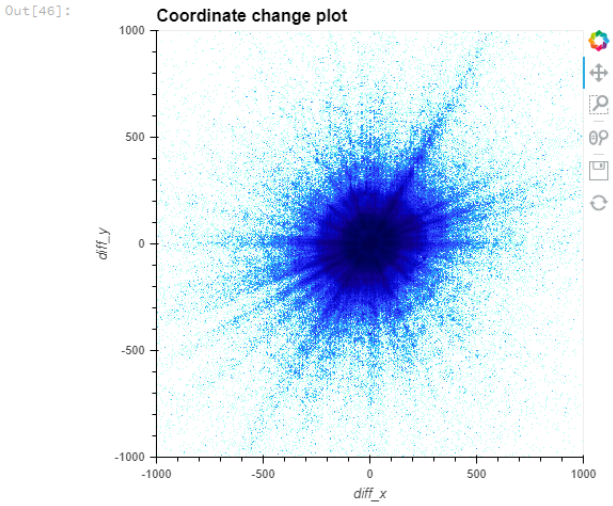

Figure 4: Datashaded coordinate change plot for assessing systematic changes in sampling intervals.

2.3 Trajectories

Of course, the visual analysis process does not stop at the segment level. By dividing the raw continuous movement tracks into individual trajectories, analysts can perform more advanced analyses of trajectory properties, such as length, duration, start and end time and location, mean speed, and overall direction. The results of these trajectory-based analyses, of course, vary depending on the method used to extract trajectories from the raw continuous tracks, for example, by splitting at regular time intervals, at stops, or at temporal gaps.

A convenient hvplot tool for exploring the relationships between trajectory properties are scatter matrixes (a combination of scatter plots and histograms). For example, a scatter matrix of trajectory duration versus start time, as shown in Figure 5, can help identify issues related to unstable object IDs. These issues can be caused by reassignment of IDs at certain points in time (often at midnight) or in certain intervals which lead to visible start time or duration clusters, respectively. These scatter matrixes can be used to visualized the relationship between two or more DataFrame columns. The scatter matrix in Figure 5 is not datashaded, however, since trajectories are aggregates of the raw tracking data, the resulting DataFrame of trajectory properties (*traj_df*) is much smaller than the original DataFrame of individual location records (*df*) and can therefore be plotted using conventional means, that is, using hvplot without Datashader.

3 DISCUSSION AND OUTLOOK

Notebook-based visualizations continue to advance and many of the limitations listed in prior work (such as a lack of linked views [14] or limited support for large datasets [7]) are already addressed in HoloViz. However, currently, the cartographic capabilities are rather limited when compared to dedicated visual analytics tools or desktop geographic information systems. For example, Datashader currently does not provide a convenient way to color track segments based on their attribute values (as shown in Figure 6).

Using regular Pandas DataFrame, the size of the dataset is limited by the available memory since the whole DataFrame has to fit into memory. This may be addressed by adopting Dask

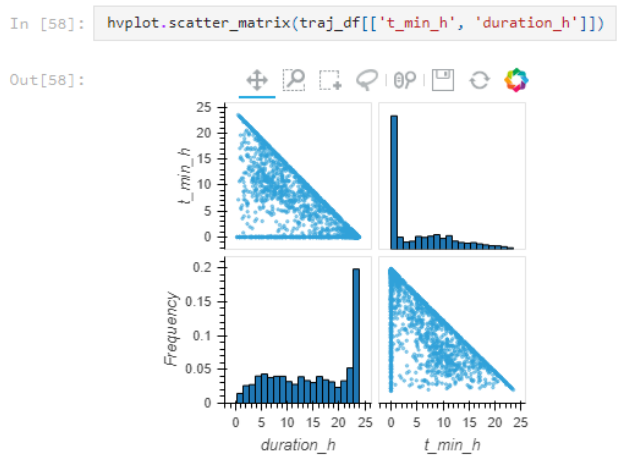

Figure 5: Hvplot scatter matrix of trajectory start time (*t_min_h* in hours) and duration (*duration_h* in hours).

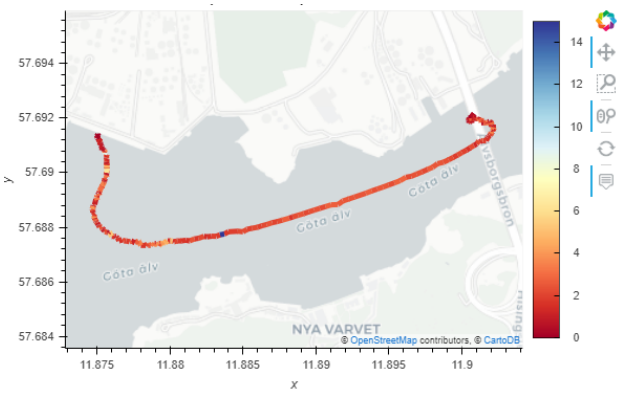

Figure 6: MovingPandas plot of speed along an individual trajectory.

DataFrames which can break down larger datasets into manageable chunks.

While Bokeh plots can usually be exported (that is saved as images), this is currently not possible for map plot, as indicated by a lack of a save icon in the Bokeh tool bar in Figures 2/3/6. This makes the process of generating plots for publications less convenient.

While linked plots allow for intuitive data exploration, their interactive nature makes it hard to keep track of the changes and to reproduce results [14]. Analysts need to be aware of these limitations and select tools accordingly, meaning that static plots may be preferable in settings where certain specific results need to be reproducible by others.

Further research will look into the possibility of creating notebook-based animations of movement data, similar to what is possible in desktop GIS such as QGIS with TimeManager extension. This may be achieved using hvPlot's support for streaming DataFrames or other similar tools that still need to be evaluated.

ACKNOWLEDGMENTS

This work was supported by the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) within the "IKT der Zukunft" programme under Grant 861258 (project MARNG).

REFERENCES

[1] Natalia Adrienko and Gennady Adrienko. 2010. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on visualization and computer graphics* 17, 2 (2010), 205–219.

[2] Gennady Andrienko, Natalia Andrienko, Peter Bak, Daniel Keim, Slava Kisilevich, and Stefan Wrobel. 2011. A conceptual framework and taxonomy of techniques for analyzing movement. *Journal of Visual Languages & Computing* 22, 3 (2011), 213–232.

[3] Gennady Andrienko, Natalia Andrienko, and Georg Fuchs. 2016. Understanding movement data quality. *Journal of location Based services* 10, 1 (2016), 31–46.

[4] The datashader development team. 2020. *holoviz/datashader.* https://doi.org/10.5281/zenodo.3844614

[5] The MovingPandas development team. 2020. *anitagraser/movingpandas: v0.5rc1.* https://doi.org/10.5281/zenodo.4051343

[6] Anita Graser. 2019. Movingpandas: Efficient structures for movement data in python. *GIForum* 1 (2019), 54–68.

[7] Anita Graser and Melitta Dragaschnig. 2020. Exploring movement data in notebook environments. *IEEE VIS 2020 Workshop on Information Visualization of Geospatial Networks, Flows and Movement (MoVis)* (2020).

[8] Anita Graser, Peter Widhalm, and Melitta Dragaschnig. 2020. Extracting Patterns from Large Movement Datasets. *GI_Forum 2020* 8 (2020), 153–163.

[9] The holoviews development team. 2020. *holoviz/holoviews.* https://doi.org/10.5281/zenodo.596560

[10] The holoviz development team. 2020. *holoviz/hvplot.* https://doi.org/10.5281/zenodo.3634719

[11] Rocio Joo, Matthew E Boone, Thomas A Clay, Samantha C Patrick, Susana Clusella-Trullas, and Mathieu Basille. 2020. Navigating through the R packages for movement. *Journal of Animal Ecology* 89, 1 (2020), 248–267.

[12] The pandas development team. 2020. *pandas-dev/pandas: Pandas.* https://doi.org/10.5281/zenodo.3509134

[13] Luca Pappalardo, F Simini, G Barlacchi, and R Pellungrini. 2019. scikit-mobility: A Python library for the analysis, generation and risk assessment of mobility data. *arXiv preprint arXiv:1907.07062* (2019).

[14] Johanna Schmidt and Thomas Ortner. 2020. Visualization in Notebook-Style Interfaces. *The Eurographics Association* (2020).

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