Exploring movement data in notebook environments

Anita Graser*†, Melitta Dragaschnig*
*AIT Austrian Institute of Technology, Vienna, Austria
†University of Salzburg, Salzburg, Austria
Email: anita.graser@ait.ac.at
ORCID: AG: 0000-0001-5361-2885, MD: 0000-0001-5100-2717

***Abstract*—Notebooks environments have become popular for data analysis but their default visualization capabilities are limited. We present ongoing work on the open geospatial library MovingPandas that enables the exploration of movement data in notebooks.**

***Index Terms*—exploratory spatial data analysis, movement data, flows, visual analytics, notebooks**

I. Introduction

Visualizations of movement data often suffer from visual clutter, thus ending up looking like a "haystack" or "ball of yarn". These visualization challenges make it hard to perform exploratory analysis on movement data. Patterns that may be visible within one type of visualization (with a specific set of parameter settings) can be completely obscured in another visualization or with different parameter settings. For efficient data exploration, analysts therefore need to be able to explore multiple different visualization types and to vary the corresponding parameter settings.

Notebook-style interfaces enable analysts to track "the steps in a data analysis workflow in a narrative way, for reporting, and for collaboration." [1] However, visualization techniques in common notebook-style interfaces, such as Jupyter or Zeppelin, are not on par with stand-alone visual analytics tools yet. [1]

We present our work towards an efficient exploration of movement data in notebook-style environments. The visualizations are implemented in the open source Python library MovingPandas[1] [2] which builds on the data handling capabilities of Pandas and Geopandas[2], as well as the visualization capabilities of Holoviz GeoViews[3].

II. Movement data exploration with MovingPandas & Jupyter

To support data analysts exploring movement data using notebook environments, software libraries should automate commonly reoccurring tasks and make it easy to interact with the data. Therefore, MovingPandas and similar libraries implement commonly used analysis functions for trajectories, such as length, duration, and speed computations [2]. In addition to these basic functions, the key features of MovingPandas for movement data exploration are related to data import, visualization, and spatiotemporal analysis.

Flexible data import capabilities are key to fast exploration iterations. Data loading is delegated to the underlying GeoPandas library which implements reading from numerous file formats, as well as from databases. GeoPandas only needs to provide functions to convert regular GeoPandas GeoDataframes into MovingPandas Trajectories or TrajectoryCollections.

Listing 1. Creation of a TrajectoryCollection from GeoPandas GeoDataframe

```
import geopandas as read_file
import movingpandas as mpd
gdf = read_file('data.gpkg')
gdf['t'] = pd.to_datetime(gdf['t'])
gdf = gdf.set_index('t')
trajectory_collection =
    mpd.TrajectoryCollection(gdf, 'traj_id')
```

Interactive visualizations of trajectory objects are drawn using linear interpolation between consecutive points, as shown in Figure 1. The underlying Holoviz libraries (particularly Geoviews) provide map interaction tools (pan, zoom by box or scroll wheel, reset, and hover map tips), as well as multiple background maps. The Panel library makes it possible to add further control elements, such as the sliders and dropdown menu in Figure 1.

Listing 2. Plot speed along trajectories (with legend and specified figure size)

```
trajectory_collection.hvplot(
    c='speed', line_width=7.0, width=700,
    height=400, colorbar=True)
```

Spatiotemporal analysis capabilities, such as filtering trajectories by geographic area of interest or geometry generalization, enhance basic spatial analysis capabilities of GeoPandas to also handle the temporal dimension. Support for geographic (latitude/longitude) as well as projected coordinates ensures that analysts can directly use a variety of input datasets. Furthermore, trajectory segmentation (see Figure 2) and aggre-

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

[1]http://movingpandas.org; doi:10.5281/zenodo.3710950
[2]http://geopandas.org, doi:10.5281/zenodo.2585848
[3]http://holoviz.org, doi:10.5281/zenodo.3727305

gation tools (see Figure 3, including methods by [3]) enable the exploration of movement patterns.

Listing 3. Trajectory segmentation (splitting) options

```
mpd.TemporalSplitter(traj).split(mode="year")
mpd.ObservationGapSplitter(traj).split(
    gap=timedelta(hours=1))
mpd.SpeedSplitter(traj).split(
    speed=10, duration=timedelta(minutes=5))
mpd.StopSplitter(traj).split(
    max_diameter=30,
    min_duration=timedelta(seconds=60))
```

III. Discussion & outlook

Notebook-based visualization continue to advance. The examples in Figures 1-3 show how visualizations can be composed from different interacting elements, including maps, graphs, and control elements. By default, the map views are synced, that is, panning or zooming one map automatically changes the second one as well. However, zooming into the map does not automatically apply a spatial filter to the histogram values in Figure 1.

Another limitation of the current MovingPandas hvplot implementation is that it does not yet provide linked brushing (for example, to select certain speed bins and have them highlighted on the map) even though the underlying Holoviews library supports brushing.

The cartographic capabilities are currently rather limited when compared to dedicated visual analytics tools or desktop geographic information systems. As a consequence, for example, it is not possible to offset the lines in Figure 3 to prevent directed flows between two nodes from overlapping[4]. There is also no straight-forward way to create animated trajectories (similar to, for example, QGIS TimeManager[5] animations) with MovingPandas in Jupyter.

Other limitations concern rendering speed, which may be addressed using Datashader, a Holoviz library for fast distributed rendering on CPUs and GPUs.

Beyond notebook environments, MovingPandas can interface with the moving objects database MobilityDB[6] [6] through mobilitydb-sqlalchemy[7] to build movement data analysis applications where expensive analytical computations can be delegated to a (potentially distributed) database.

Besides MovingPandas, more work on building reusable movement data analysis functionality in Python is also going on in the scikit-mobility project [7]. Other related work includes dozens of R libraries [8] which may be used in R notebooks[8]. However, we are not aware of any R notebook examples for movement data analysis.

[4]https://github.com/holoviz/geoviews/issues/431
[5]https://anitagraser.com/projects/time-manager
[6]http://mobilitydb.com
[7]https://github.com/adonmo/mobilitydb-sqlalchemy
[8]https://blog.rstudio.com/2016/10/05/r-notebooks/

References

[1] J. Schmidt, T. Ortner, "Visualization in Notebook-Style Interfaces," in Proc. of VisGap, 2020. doi:10.2312/visgap.20201104.

[2] A. Graser, "MovingPandas: Efficient Structures for Movement Data in Python," GI_Forum, vol. 1-2019, pp.54–68, 2019. doi:10.1553/giscience2019_01_s54.

[3] N. Andrienko, G. Andrienko, "Spatial generalization and aggregation of massive movement data," IEEE Trans. Vis. Comput. Graph, vol. 17(2), 205–219, 2011.

[4] M. Wikelski, E. Arriero, A. Gagliardo, et al., "Data from: True navigation in migrating gulls requires intact olfactory nerves. Movebank Data Repository." doi:10.5441/001/1.q986rc29, 2015.

[5] Y. Zheng, Q. Li, Y. Chen, et al., "Understanding Mobility Based on GPS Data," in Proc. of ACM UbiComp, 2008.

[6] E. Zimányi, M. Sakr, A. Lesuisse, et al., "MobilityDB: A Mainstream Moving Object Database System," in Proc. of ACM SSTD, pp.206–209, 2019.

[7] L. Pappalardo, F. Simini, G. Barlacchi, R. Pellungrini, "scikit-mobility: a Python library for the analysis, generation and risk assessment of mobility data," 2019. https://arxiv.org/abs/1907.07062.

[8] R. Joo, M.E. Boone, T.A. Clay, et al., "Navigating through the R packages for movement," J. Anim. Ecol, 89(1), 248-267, 2020.

```
def my_plot(tolerance=0.001, generalizer='douglas-peucker'):
    if generalizer=='douglas-peucker':
        generalized = mpd.DouglasPeuckerGeneralizer(my_traj).generalize(tolerance=tolerance)
    else:
        generalized = mpd.MinDistanceGeneralizer(my_traj).generalize(tolerance=tolerance)
    generalized.add_speed(overwrite=True)
    return ( generalized.hvplot(title='Trajectory {}'.format(generalized.id), c='speed', cmap='Viridis', colorbar=True, clim=(0,20), lin
             generalized.df['speed'].hvplot.hist(title='Speed histogram', width=300, height=500)
           )
```

```
kw = dict(tolerance=(0, 1000), generalizer=['douglas-peucker', 'min-distance'])
pn.interact(my_plot, **kw)
```

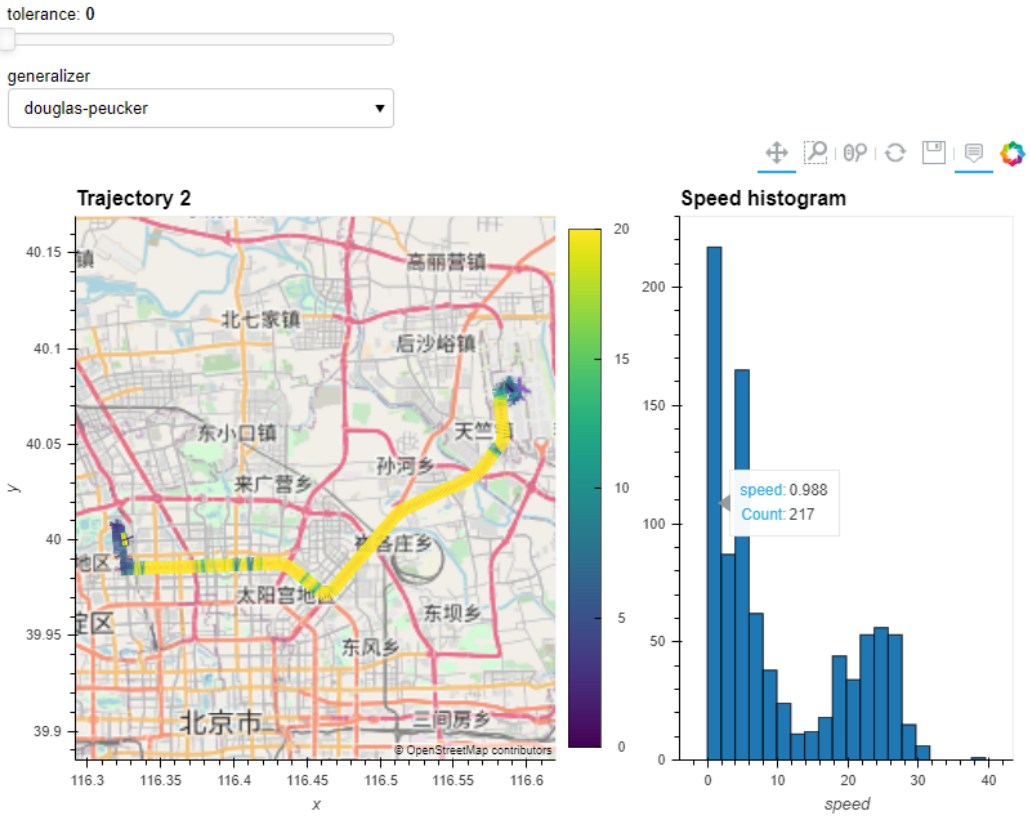

Fig. 1. **Exploring the effects of trajectory generalization:** analysts can switch between generalizer implementations and adjust the generalization tolerance using controls at the top. The map view shows the trajectory in its spatial context and segment speed is color-coded. The distribution of segment speeds is summarized in the histogram on the right. Geolife trajectory data by [5]. Online live demo at: https://mybinder.org/v2/gh/anitagraser/movingpandas/v0.5rc1?filepath=tutorials/demo_panel.ipynb

```
%%time
stops = mpd.TrajectoryStopDetector(my_traj).get_stop_segments(min_duration=timedelta(seconds=60), max_diameter=100)
len(stops)
```

```
Wall time: 566 ms

7
```

```
%%time
split = mpd.StopSplitter(my_traj).split(min_duration=timedelta(seconds=60), max_diameter=100)
```

```
Wall time: 625 ms
```

```
( my_traj.hvplot(title='Stops in Trajectory {}'.format(my_traj.id), line_width=7.0, tiles='CartoLight', color='slategray', width=400,
 stops.get_start_locations().hvplot(geo=True, size=200, color='deeppink')  *
 split.hvplot(title='Trajectory {} split at stops'.format(my_traj.id), line_width=7.0, tiles='CartoLight', width=400, height=700)
)
```

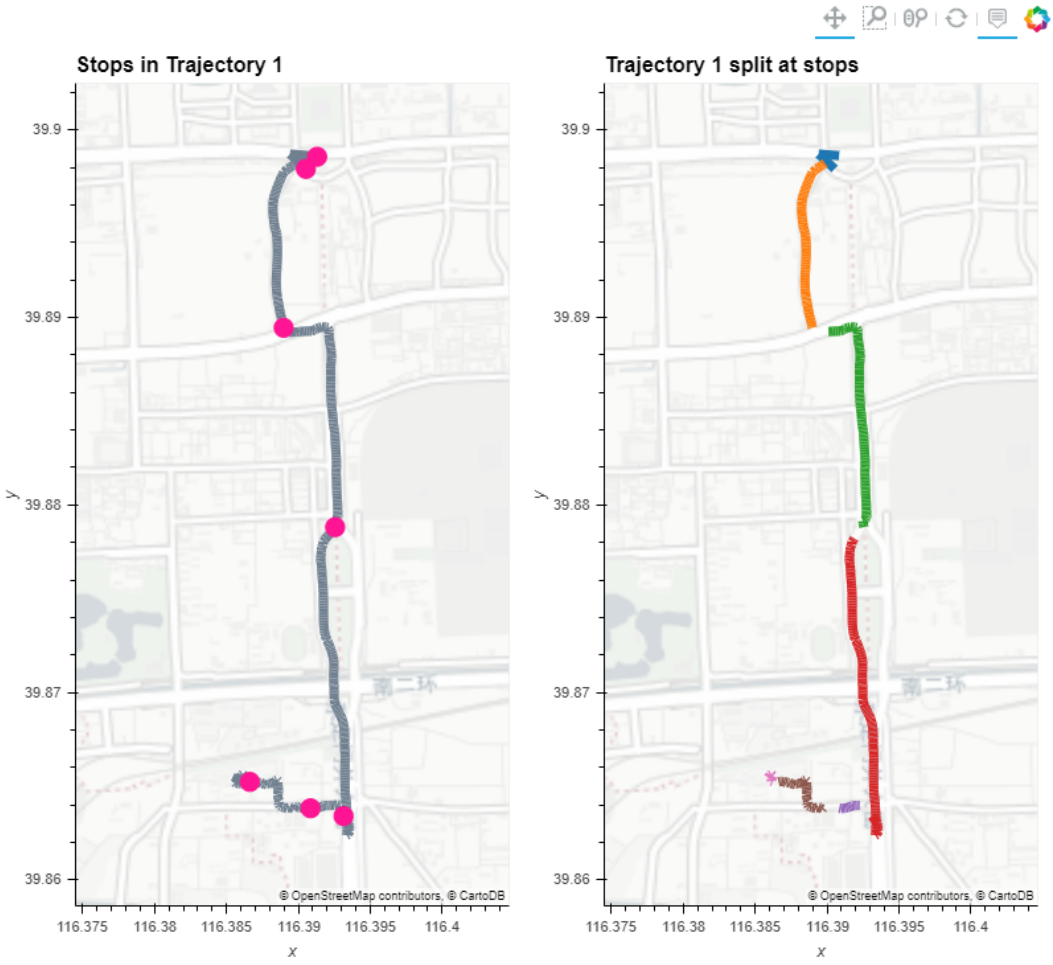

Fig. 2. **Splitting trajectories at stops:** seven stops detected along a trajectory on the left and the same trajectory split into segments at these stops on the right. Geolife trajectory data by [5]. Online live demo at: https://mybinder.org/v2/gh/anitagraser/movingpandas/v0.5rc1?filepath=tutorials/demo_stop_detection.ipynb

```
( flows.hvplot(title='Generalized aggregated trajectories', geo=True, hover_cols=['weight'], line_width='weight', alpha=0.5, colo
  clusters.hvplot(geo=True, color='red', size='n')
+
 GeoDataFrame(od_matrix).hvplot(title='OD flows', geo=True, tiles='OSM', hover_cols=['n'], line_width='n', alpha=0.5, frame_heigh
  GeoDataFrame(matrix_nodes).hvplot(c='n', size='symbol_size', hover_cols=['cluster', 'n'], geo=True,  cmap='RdYlGn')
)
```

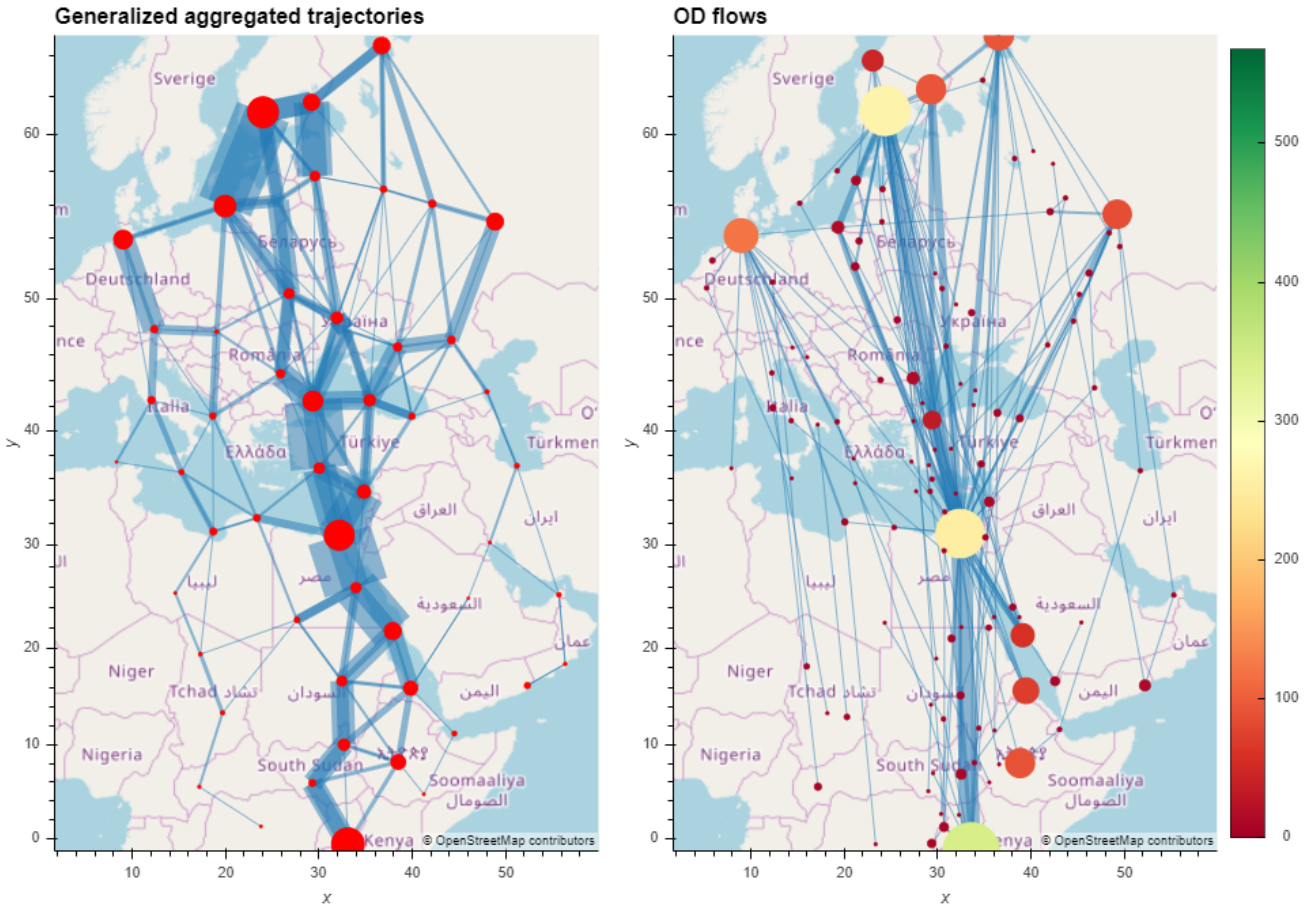

Fig. 3. **Comparing trajectory summarizations**: generalized aggregation [3] on the left and clustered origin-destination flows on the right. Gull migration data by [4]. Online live demo at: https://mybinder.org/v2/gh/anitagraser/movingpandas/v0.5rc1?filepath=tutorials/demo_od_matrix.ipynb