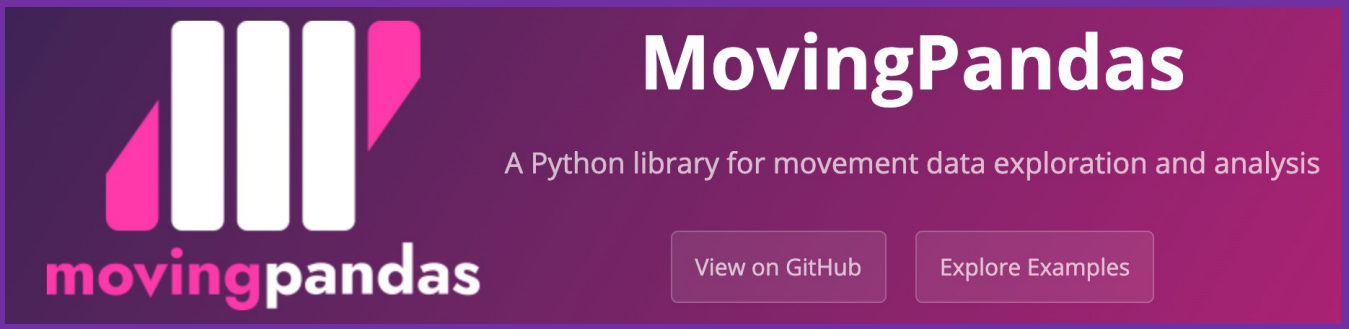
Sailing through AIS data with movingpandas

Ray Bell

PyData Miami 2022



Outline

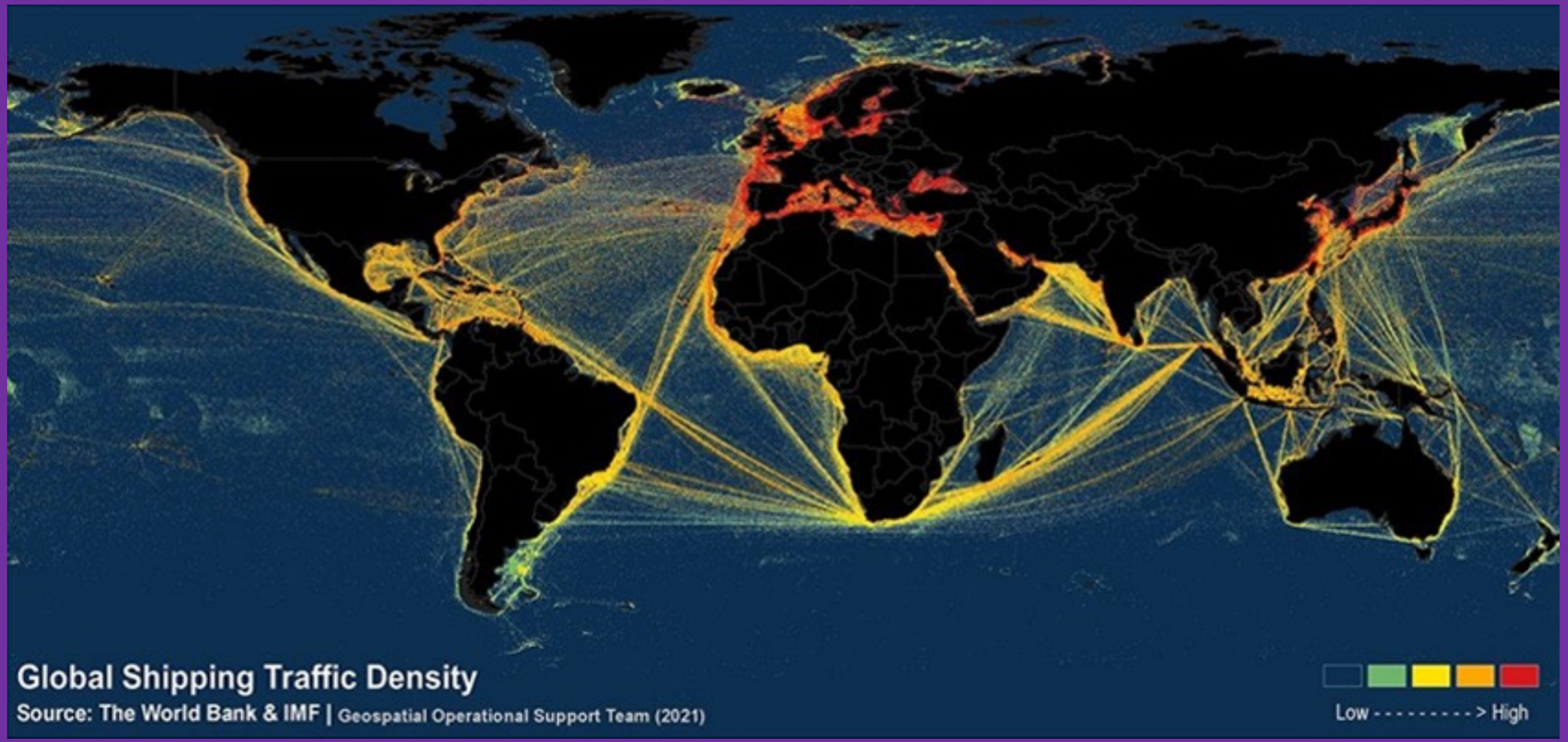
- What is AIS data?
- What is movingpandas?
- Whistle stop tour of geopandas
- movingpandas demo including:
  - base class
  - cleaning AIS data
  - voyage summary statistics
  - Clustering of location hot spots
  - Trajectory smoothing
- Future development of movingpandas

https://www.easyais.com/en/ais-info/what-is-ais/

What is AIS data?

- Automatic tracking system on ships defined by the IMO (International Maritime Organization)
- Like ADS-B (Automatic Dependent Surveillance-Broadcast) for aircraft tracking

https://www.easyais.com/en/ais-info/what-is-ais/



https://blogs.worldbank.org/opendata/using-marine-spatial-data-inform-development-work-and-public-policies

Why do we need AIS data?

- Avoid ships colliding
- Enables ports and coastal states to manage and supervise the traffic in their waters
- Monitor activity
- Maritime security
- Economic indicators – speed of trade

How to analysis AIS data?

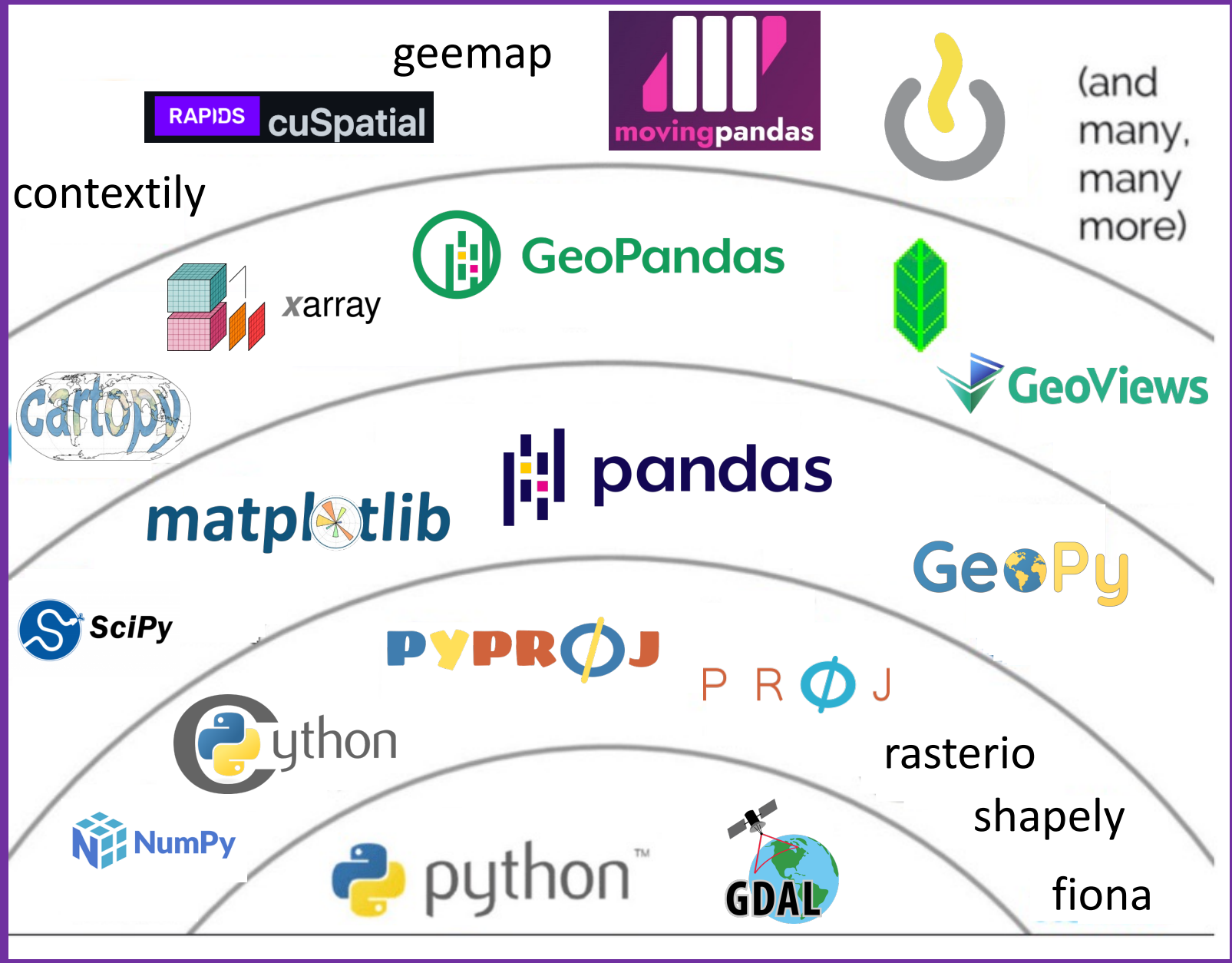
- Streaming – real-time applications/dashboards:
  - marinetraffic.com
  - vesselfinder.com
- Location filtering and cleaning
- Voyage analysis – help with fuel consumption/de-carbonization
- Heat maps – traffic density
- Smoothing trajectories

What is MovingPandas?

- A python library for movement data exploration and analysis
- Created in 2018 by Anita Graser
- Separated out of a QGIS plugin
- 24 contributors
- GeoPandas-contrib like
- https://github.com/anitagraser/movingpandas/discussions

Graser (2019) MovingPandas: Efficient Structures for Movement Data in Python, Jornal of Geographic Information Science, 1-2019, 54-68

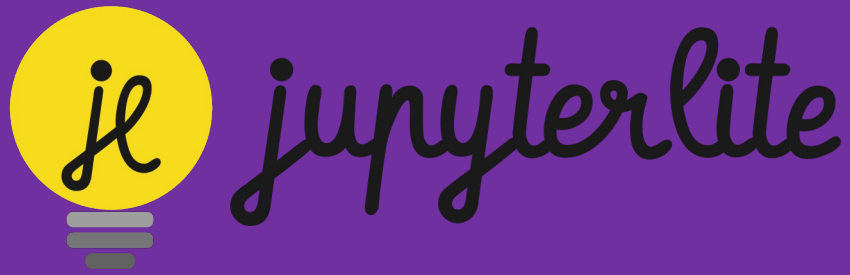
The Geo "Py" Data stack

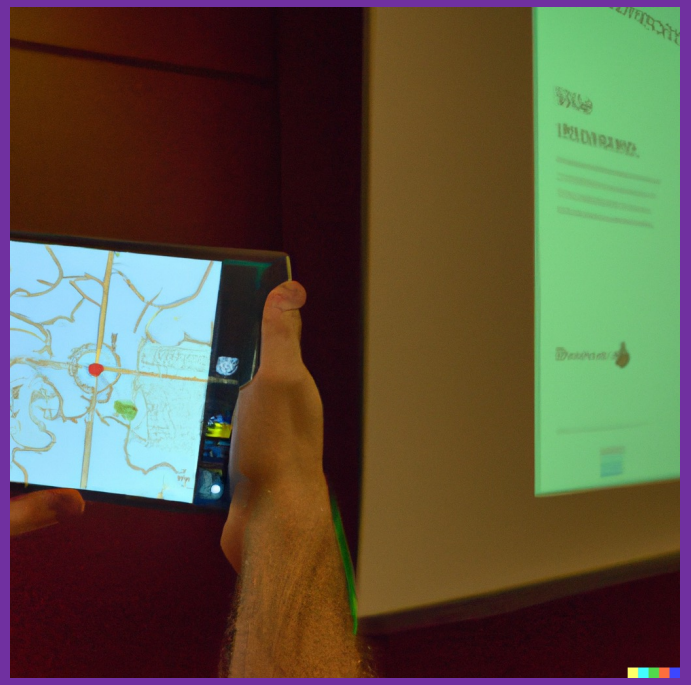


Adapted from VanderPlas, PyCon 2017

Technical Demo

Expectation





"A very exciting interactive demo of a geospatial python library presented on a cell phone at a PyData conference in Miami" – DALL-E

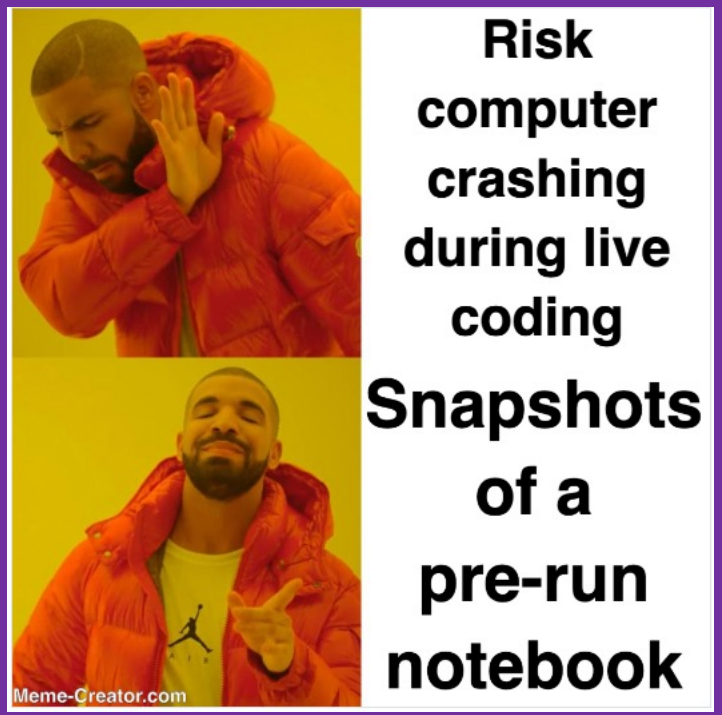
Technical Demo

Reality

“A bored PyData Miami conference attendee looking at static matplotlib images for geographic point data somewhere on the earth” – DALL-E

Technical demo of MovingPandas

- Whistle stop tour of GeoPandas
- Exploration of AIS data for Ever Given
- Data cleaning
- Voyage statistics
- Density maps and clustering
- Trajectory smoothing
- Other applications of MovingPandas
- Future development of MovingPandas



Read in Ever Given AIS data which is stored as a GeoParquet file (https://github.com/opengeospatial/geoparquet)

```
import geopandas as gpd

gdf = gpd.read_parquet("ever_given_ais.parquet")
```

Whatever you can do in Pandas you can likely do in GeoPandas and more

```
gdf.head()
```

Last executed at 2022-09-21 15:47:41 in 82ms

	timestamp	created_at	heading	speed_knots	rate_of_turn	collection_type	maneuver	course	timestamp_iso	longitude	latitude	geometry
0	2018-12-07 00:11:01+00:00	2018-12-07 00:11:20.834856+00:00	128.0	21.5	-5.0	terrestrial	0.0	127.9	2018-12-07T00:11:01.000Z	102.92997	1.53197	POINT (102.92997 1.53197)
1	2018-12-07 00:45:33+00:00	2018-12-07 07:28:16.447564+00:00	118.0	21.4	-3.0	satellite	0.0	118.7	2018-12-07T00:45:33.000Z	103.11421	1.44149	POINT (103.11421 1.44149)

GeoPandas is Pandas + a geometry column (for Shapely geometric objects) + a CRS (Coordinate Reference System)

```
gdf.crs
```

```
Last executed at 2022-09-21 15:47:41 in 7ms
```

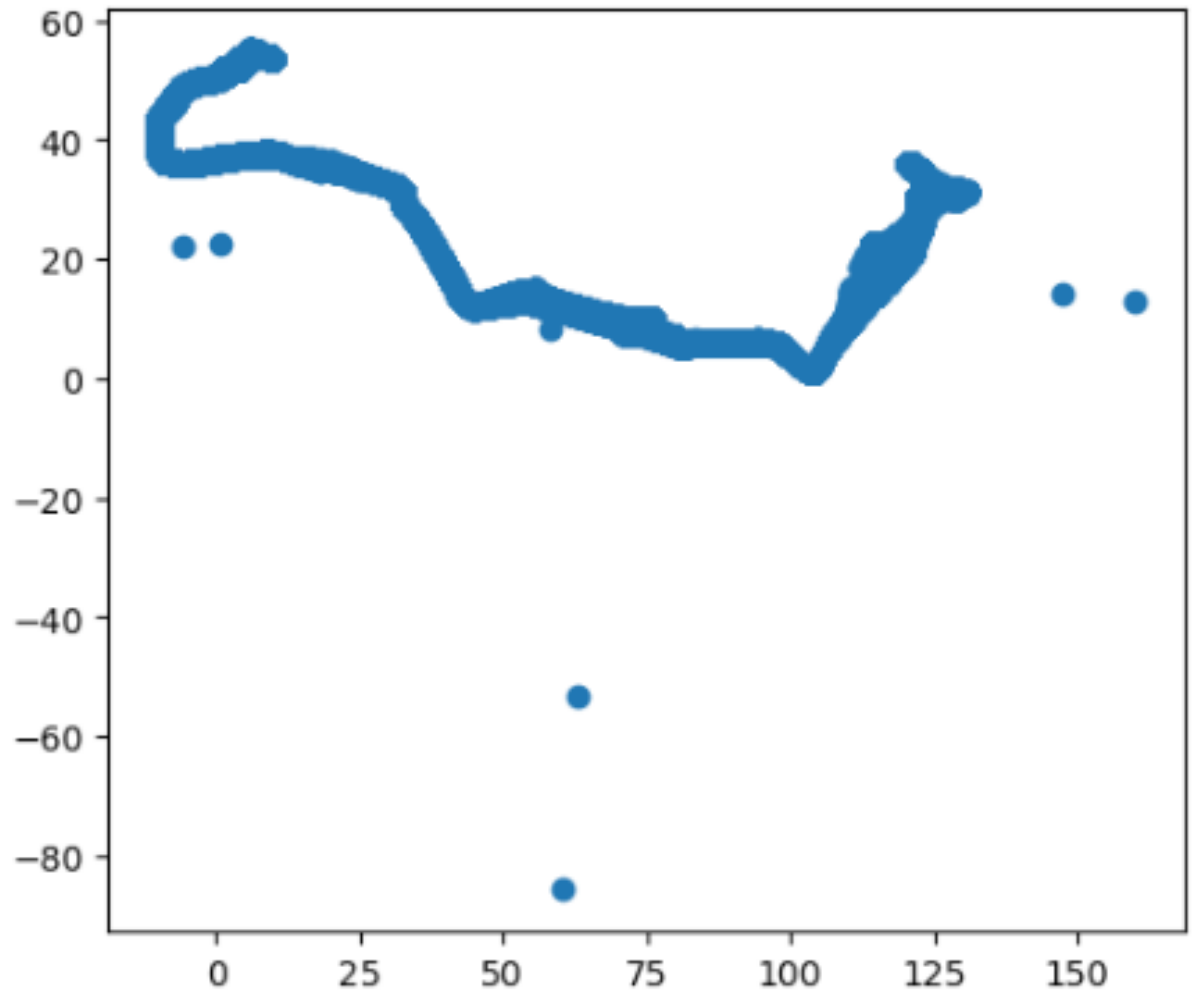
```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
- Lat[north]: Geodetic latitude (degree)
- Lon[east]: Geodetic longitude (degree)
Area of Use:
- name: World.
- bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984 ensemble
- Ellipsoid: WGS 84
- Prime Meridian: Greenwich
```

Call `.plot()` for a quick plot

```
gdf.plot()
```

```
Last executed at 2022-09-21 15:48:12 in 30.49s
```

```
<AxesSubplot: >
```

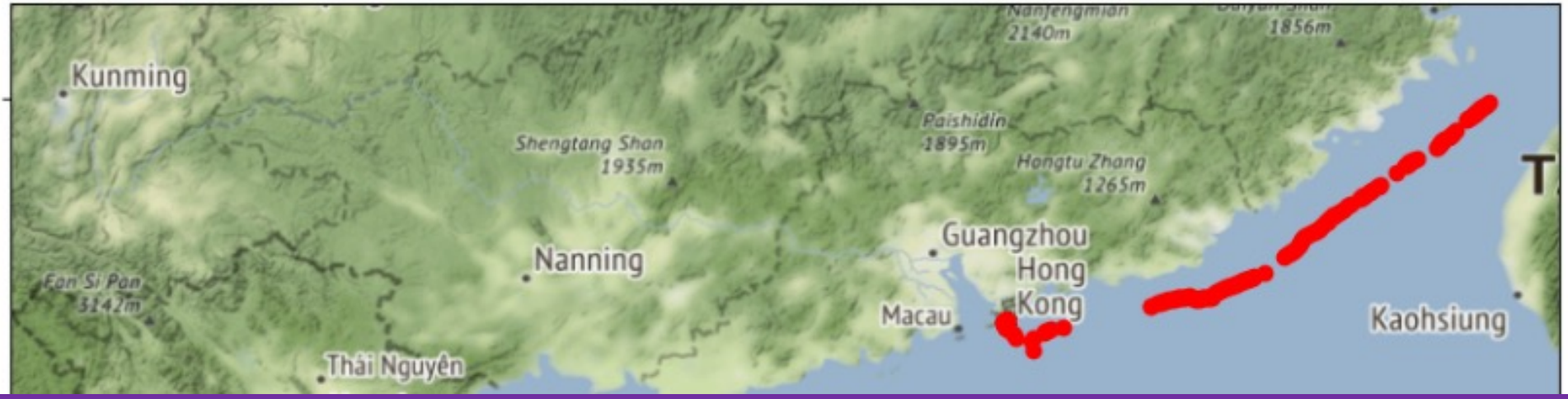


use **contextilty** to plot with a background tile (using **xyzservices** to request a tile from an API provider: https://xyzservices.readthedocs.io/en/stable/introduction.html)

```
import contextily as cx

cx.add_basemap(gdf[0:1000].plot(color="red", figsize=(16, 16)), crs=gdf.crs)
```

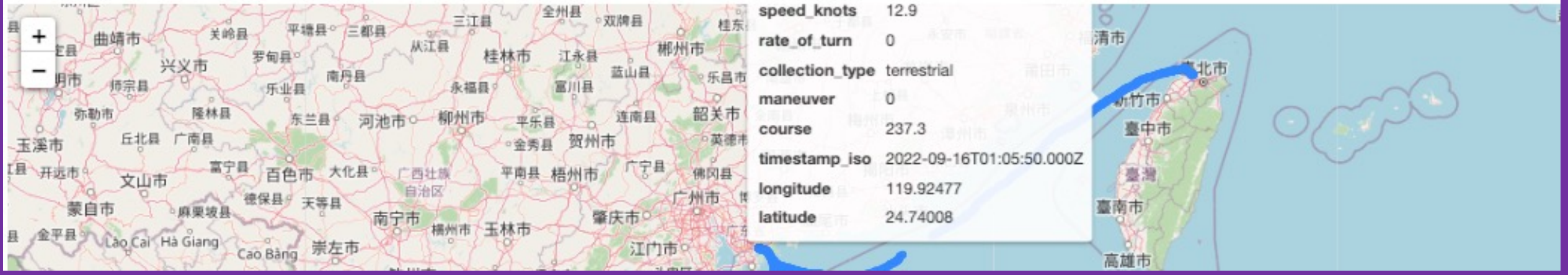
Last executed at 2022-09-21 15:48:22 in 10.38s



`.explore()` is a great tool for interactive analysis (which wraps Folium) but can be a bit sluggish on "big" data.

```
gdf[-1000:-1].select_dtypes(
    exclude="datetimetz"
).explore()  # drop datetime columns as not JSON serializable
```

```
Last executed at 2022-09-21 15:48:36 in 13.27s
```



Other tools can help such as **datashader** and **xarray_ipyleaflet**

```python
year = 2021
month = 3
gdf[
    (gdf["timestamp"].dt.year == year) & (gdf["timestamp"].dt.month == month)
].select_dtypes(exclude="datetimetz").explore()
```

Last executed at 2022-09-21 22:21:49 in 1.39s

The MovingPandas base class (Trajectory)

```
import movingpandas as mpd

traj = mpd.Trajectory(gdf[0:10000].set_index("timestamp"), 1)
```

Last executed at 2022-09-21 22:26:56 in 7.87s

The repr will give summary statistics

```
traj
```

Last executed at 2022-09-21 22:26:57 in 1.40s

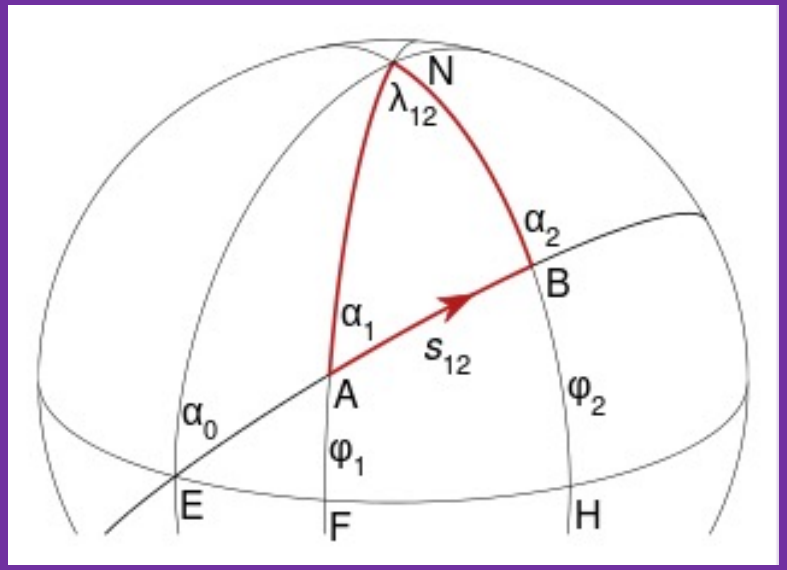
```
Trajectory 1 (2018-12-07 00:11:01+00:00 to 2019-03-15 05:33:58+00:00) | Size: 9997 | Length: 84889948.5m
Bounds: (-10.14075, 1.05323, 123.59713, 54.03032)
LINESTRING (102.92997 1.53197, 103.11421 1.44149, 103.14129 1.42546, 103.16029 1.41217, 103.18697 1.
```

Calulate speed between points and remove spurious points

```
traj.add_speed()
_gdf = traj.to_point_gdf()
_gdf = _gdf[
    _gdf["speed"] < 41.1556
]  # 80 knots. Taken from https://github.com/marinetraffic/mt-ais-toolbox/blob/main/src/mt/cleaning/data_cleaning.py#L131
traj = mpd.Trajectory(_gdf, 1)
traj
```

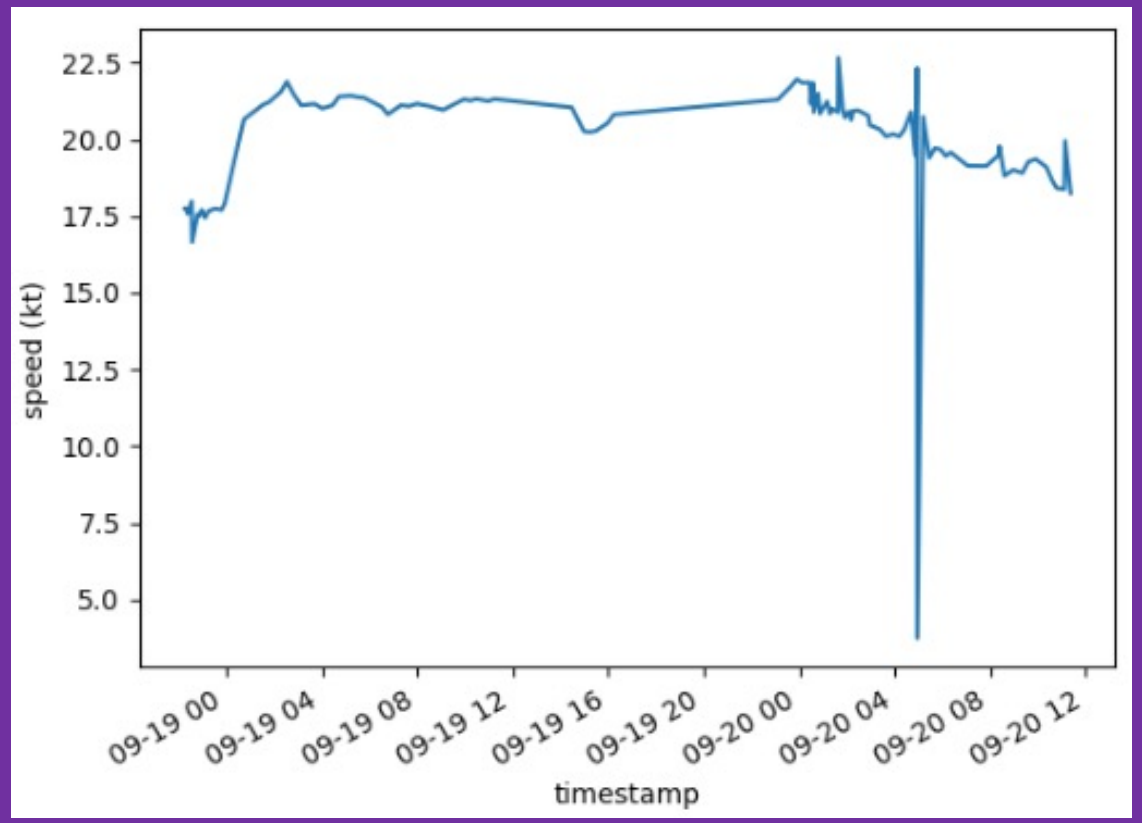
Last executed at 2022-09-21 15:48:58 in 4.31s

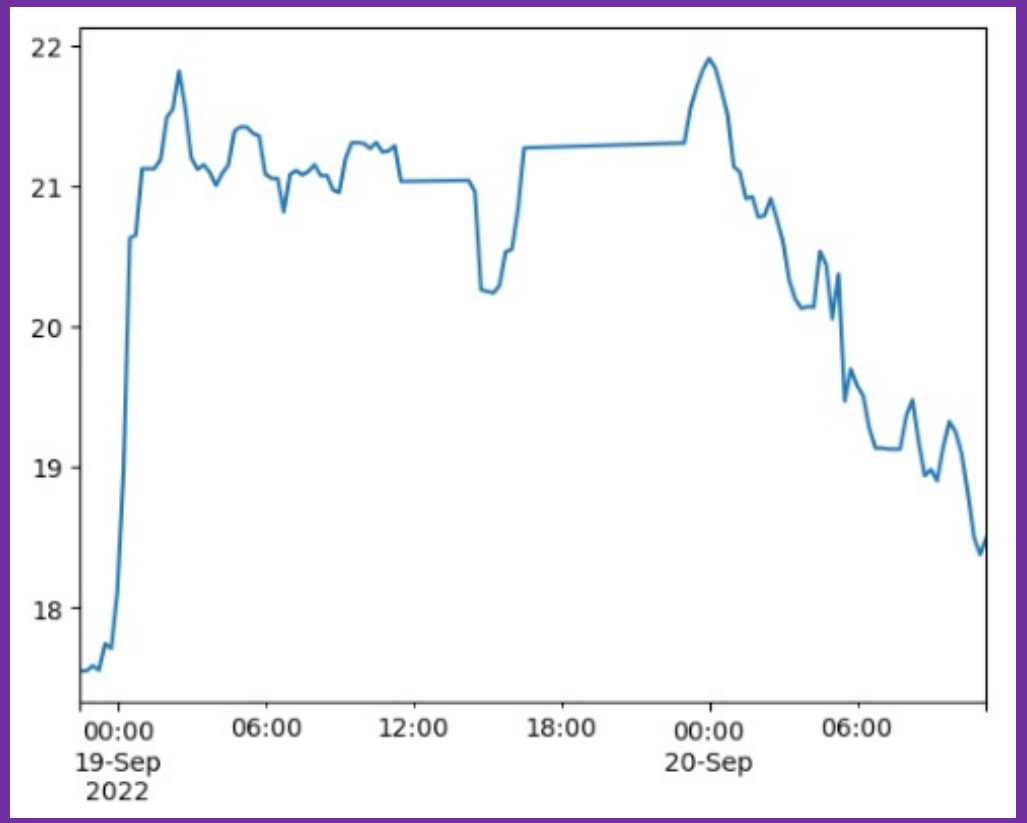
```
Trajectory 1 (2018-12-07 00:11:01+00:00 to 2019-03-15 05:33:58+00:00) | Size: 9981 | Length: 53454466.6m
Bounds: (-10.14075, 1.05323, 123.59713, 54.03032)
LINESTRING (102.92997 1.53197, 103.11421 1.44149, 103.14129 1.42546, 103.16029 1.41217, 103.18697 1.
```



Smooth data by interpolating

```
for dt in pd.date_range(
    start=pd.Timedelta.ceil(_gdf.index.min(), freq="15min"),
    end=pd.Timedelta.floor(_gdf.index.max(), freq="15min"),
    freq="15min",
):
    pts[dt] = traj.get_position_at(dt).wkt
```





Voyage statistics

- AIS data generated as Shortest path to and from Miami, US to Bristol, UK using DTN's routing API

Voyage statistics

- Split trajectory by time greater than expected in port

```
trajc = mpd.ObservationGapSplitter(traj).split(gap=timedelta(days=1))
trajc
```

```
Last executed at 2022-09-22 02:24:38 in 90ms
```

```
TrajectoryCollection with 2 trajectories
```

```
for traj in trajc:
    print(traj)
```

```
Last executed at 2022-09-22 02:24:40 in 47ms
```

```
Trajectory 1_0 (2022-09-23 12:00:00+00:00 to 2022-10-09 06:43:48+00:00) | Size: 99 | Length: 6988598.2m
Bounds: (-80.16958618164062, 25.75665283203125, -2.719085693359375, 51.50248718261719)
LINESTRING (-80.16958618164062 25.770172119140625, -80.16886901855469 25.7691650390625, -80.16438293
Trajectory 1_1 (2022-10-23 12:00:00+00:00 to 2022-11-09 12:10:44+00:00) | Size: 103 | Length: 7015720.6m
Bounds: (-80.16958618164062, 25.75665283203125, -2.719085693359375, 51.50248718261719)
LINESTRING (-2.719085693359375 51.49803161621094, -2.7212982177734375 51.496917724609375, -2.7223205
```

- This can be used to build ETA (Estimated Time of Arrival) models combined with weather data + current data + wave data

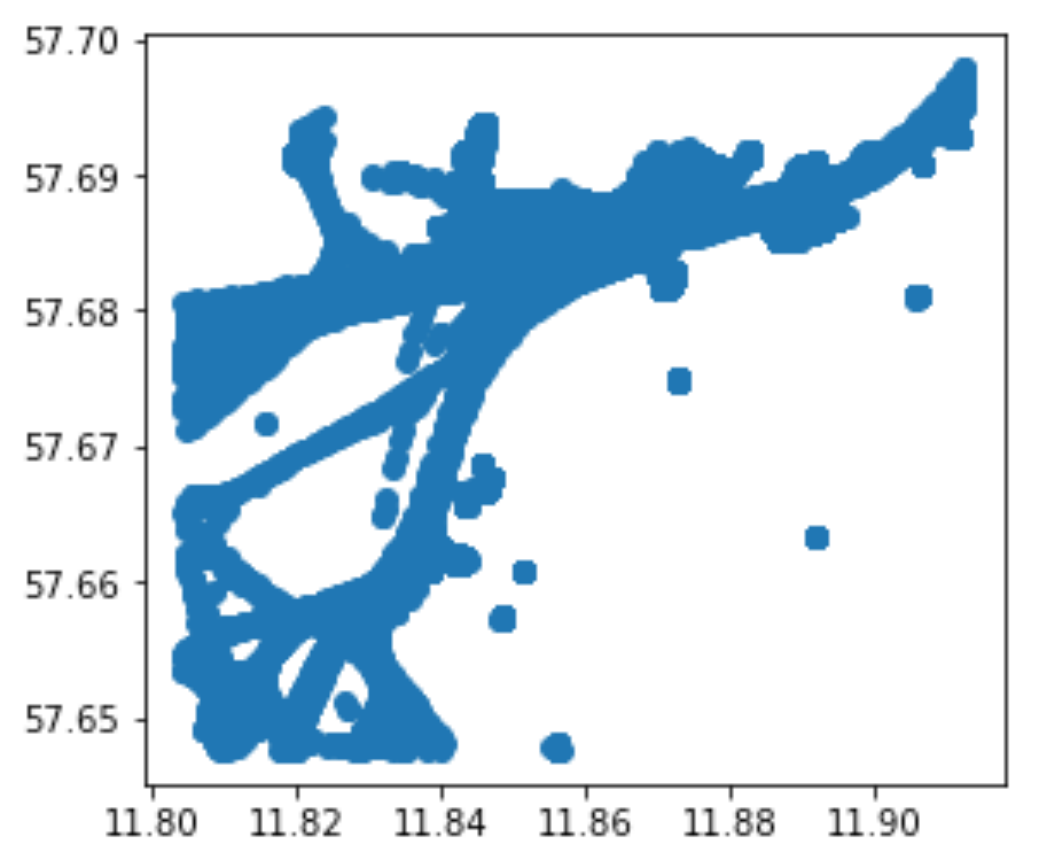
```
gdf = pd.concat(l, ignore_index=True)
traj = mpd.Trajectory(gdf.set_index("properties.eta"), 1)
trajc = mpd.ObservationGapSplitter(traj).split(gap=timedelta(days=1))
l = []
for _traj in trajc:
    l.append(_traj.get_duration())
pd.DataFrame(l)
```

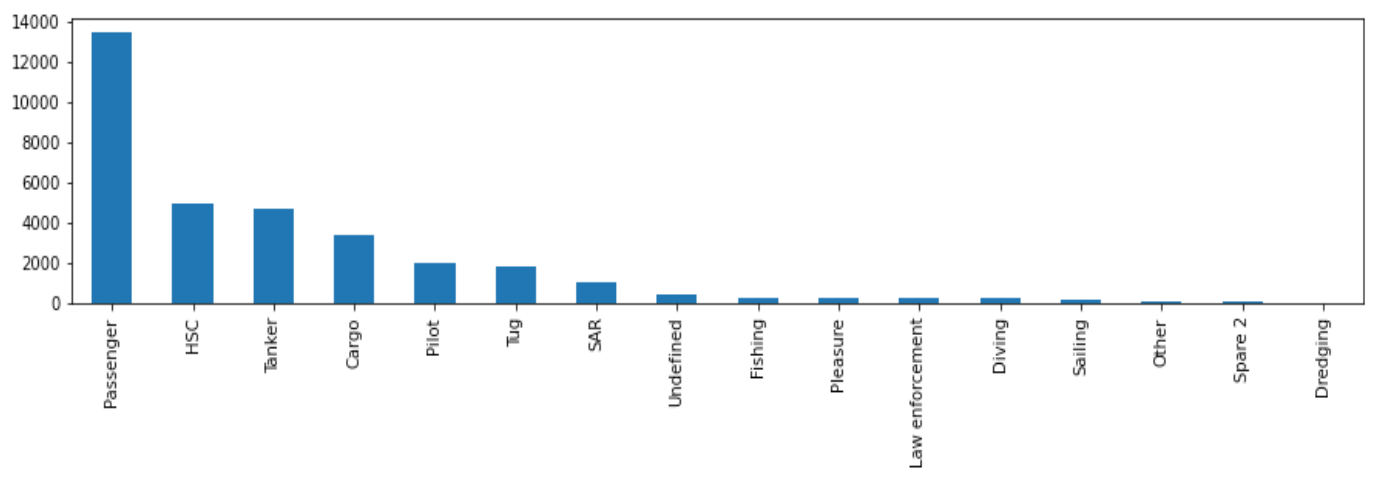
Last executed at 2022-09-22 02:55:58 in 206ms

	0
0	10 days 14:01:12
1	6 days 22:52:09
2	7 days 23:41:42
3	8 days 20:26:34
4	5 days 09:32:43

https://github.com/anitagraser/movingpandas-examples/blob/main/2-analysis-examples/ship-data.ipynb

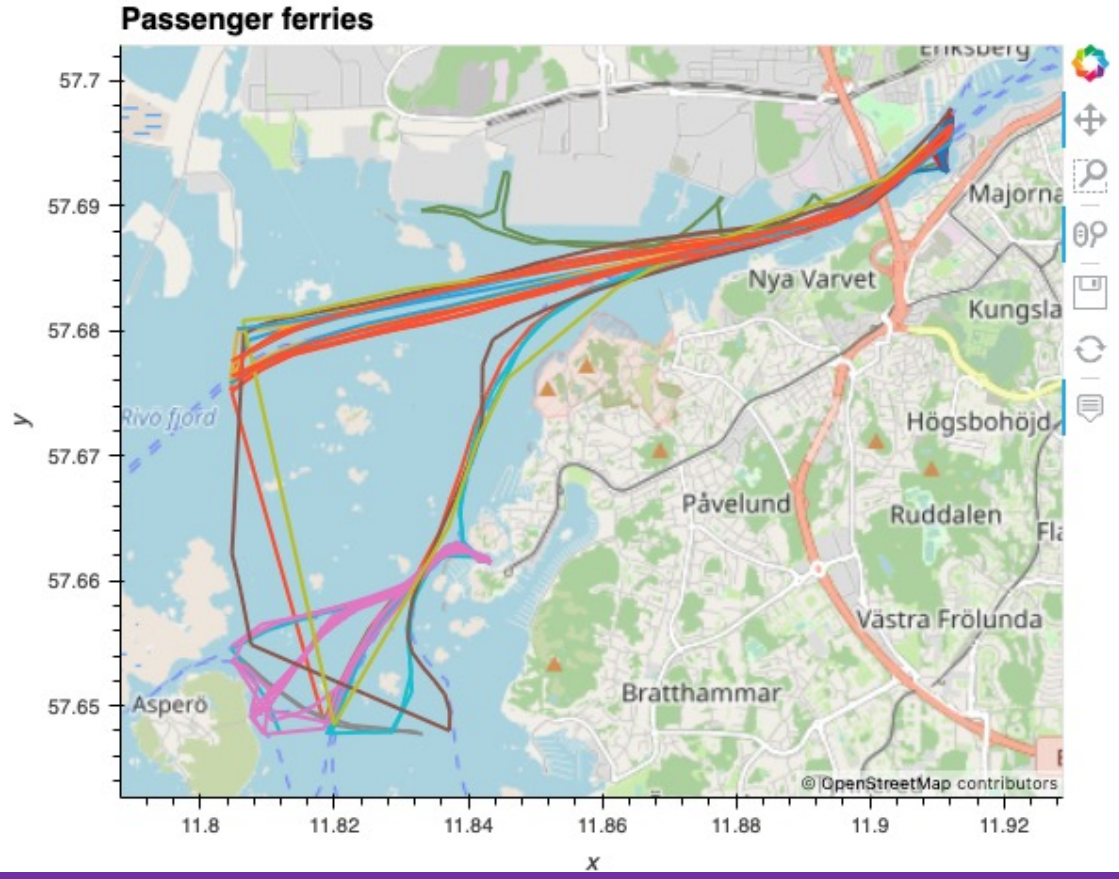
- All ship data for Danish waters in a day





```
df['t'] = pd.to_datetime(df['Timestamp'], format='%d/%m/%Y %H:%M:%S')
traj_collection = mpd.TrajectoryCollection(df, 'MMSI', t='t', min_length=100)
traj_collection = mpd.MinTimeDeltaGeneralizer(traj_collection).generalize(tolerance=timedelta(minutes=1))
```

```
passenger = traj_collection.filter('ShipType', 'Passenger')
passenger.hvplot(title='Passenger ferries', line_width=2, frame_width=700, frame_height=500)
```



Finding ships passing under Älvsborgsbron bridge

We can find ships passing under the bridge based on trajectory intersections with the bridge area.

```
area_of_interest = Polygon([(11.89935, 57.69270), (11.90161, 57.68902), (11.90334, 57.68967), (11.90104, 57.69354), (11.89935, 57.69270)])
```

```
intersecting = traj_collection.get_intersecting(area_of_interest)
print(f"Found {len(intersecting)} intersections")
```

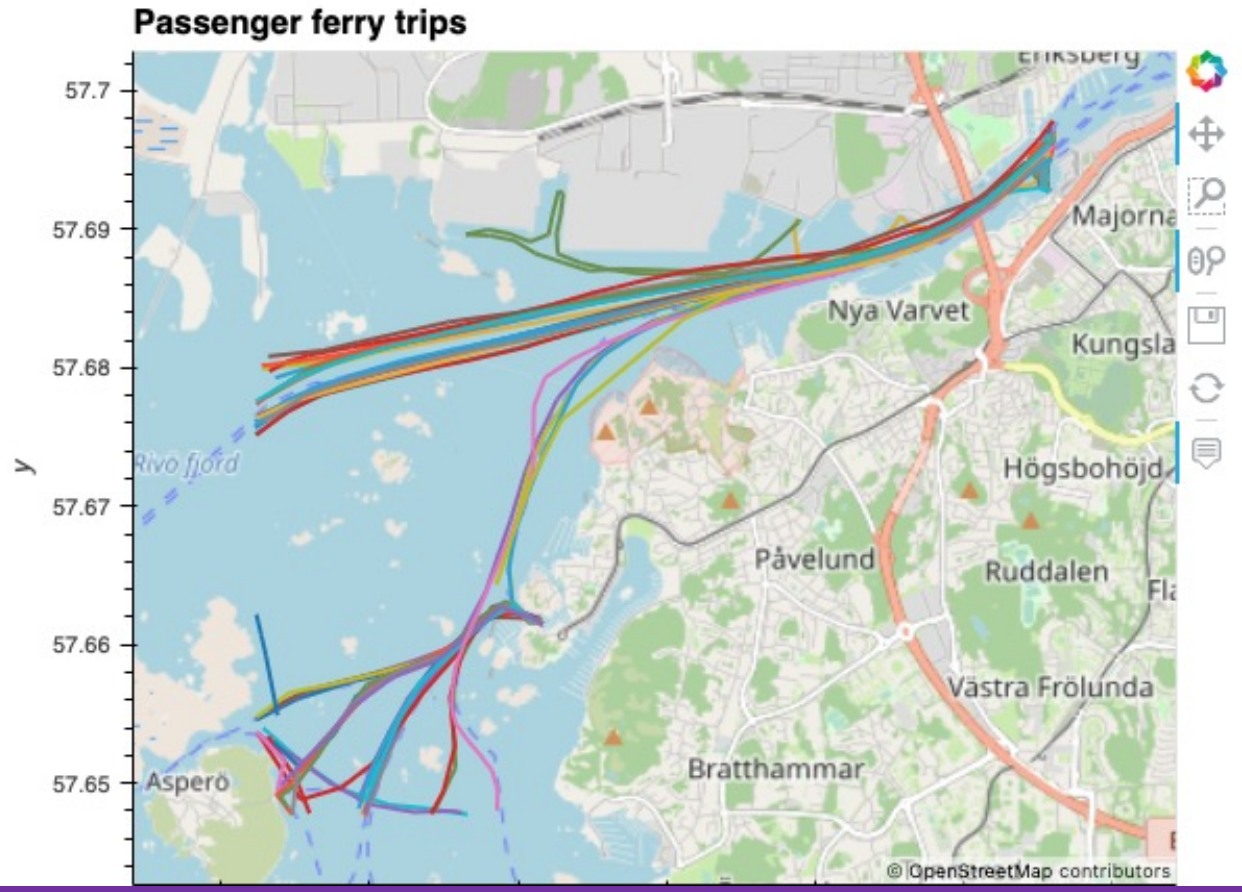
```
Found 20 intersections
```

```
bridge_traj = intersecting.trajectories[0]
bridge_traj.hvplot(title=f'Trajectory {bridge_traj.id}', frame_width=700, frame_height=500, line_width=5.0, c='NavStatus', cmap='Dark2')
```

Identifying trip origins and destinations

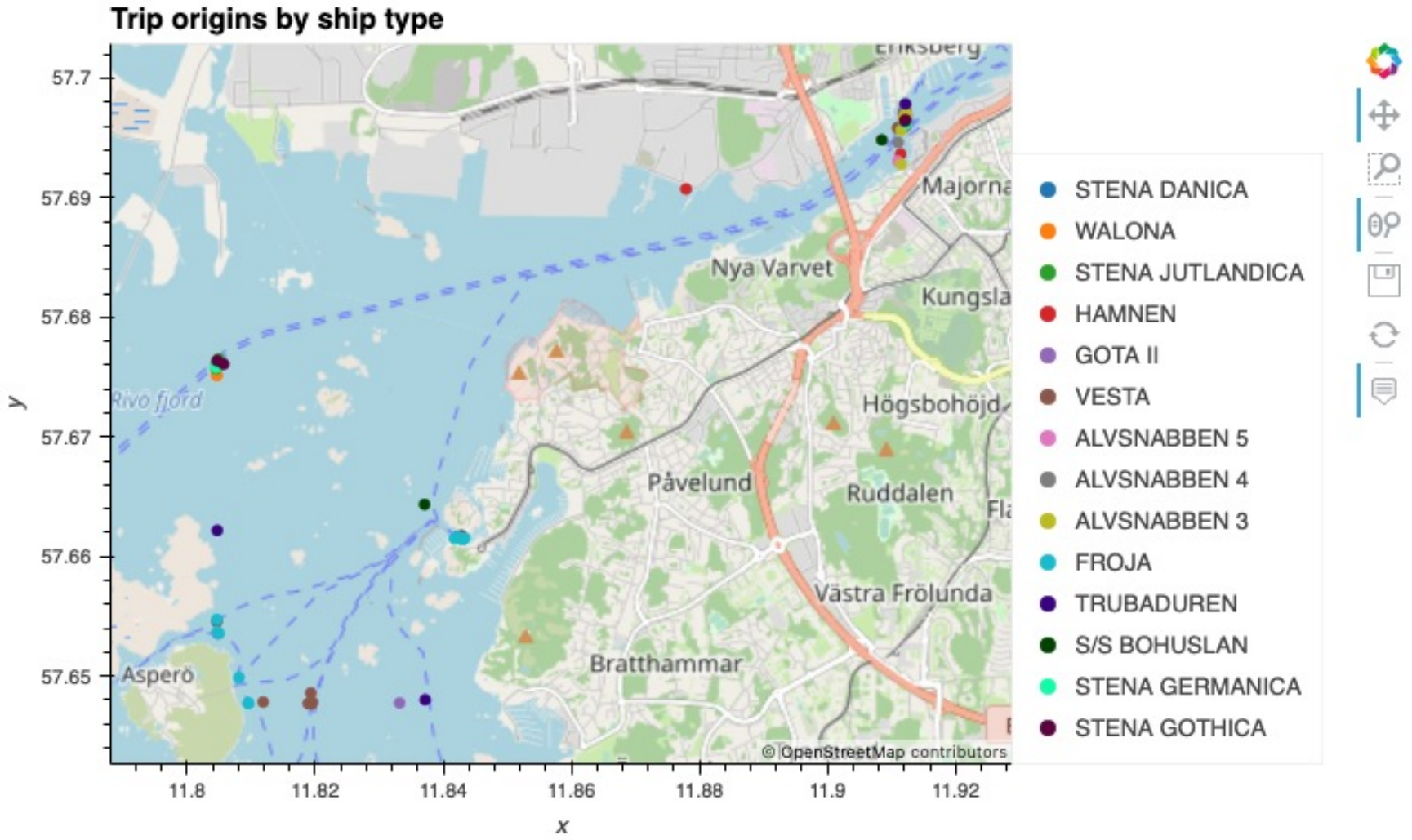
Since AIS records with a speed over ground (SOG) value of zero have been removed from the dataset, we can use the `ObservationGapSplitter()` class to split the continuous observations into individual trips:

```
trips = mpd.ObservationGapSplitter(passenger).split(gap=timedelta(minutes=5))
trips.hvplot(title='Passenger ferry trips', line_width=2, frame_width=700, frame_height=500)
```



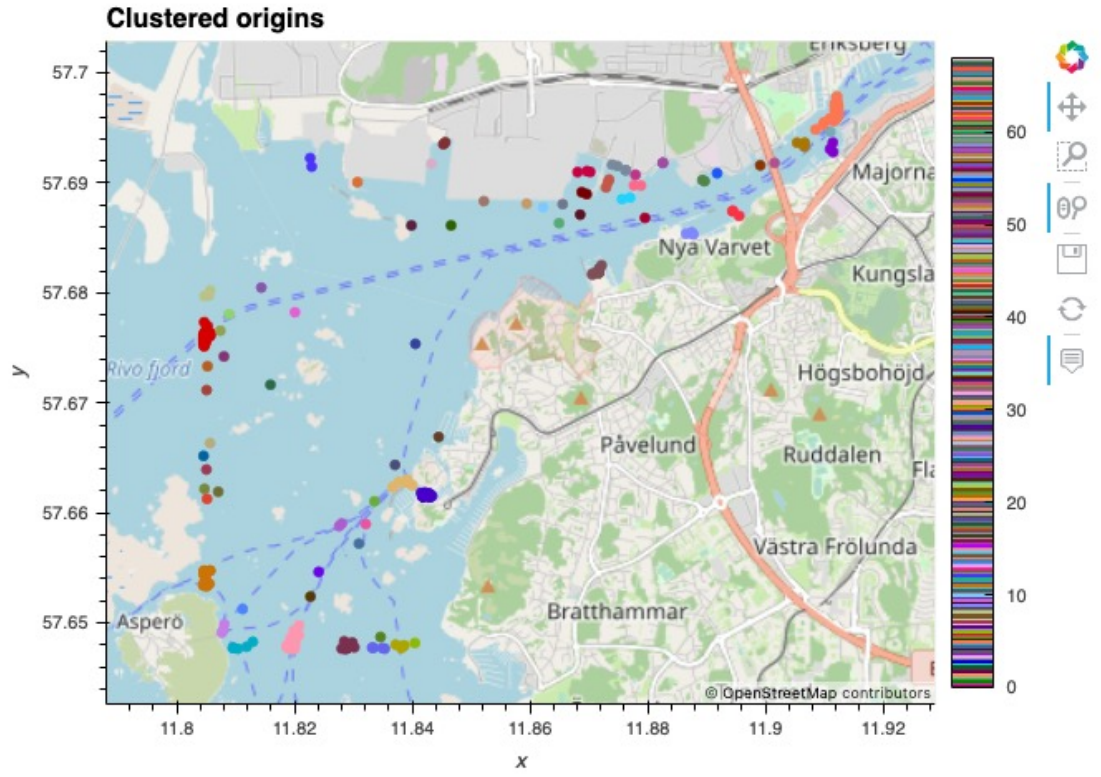
Next, let's get the trip origins:

```
origins = trips.get_start_locations()
origins.hvplot(title='Trip origins by ship type', c='Name', geo=True, tiles='OSM', frame_width=700, frame_height=500)
```

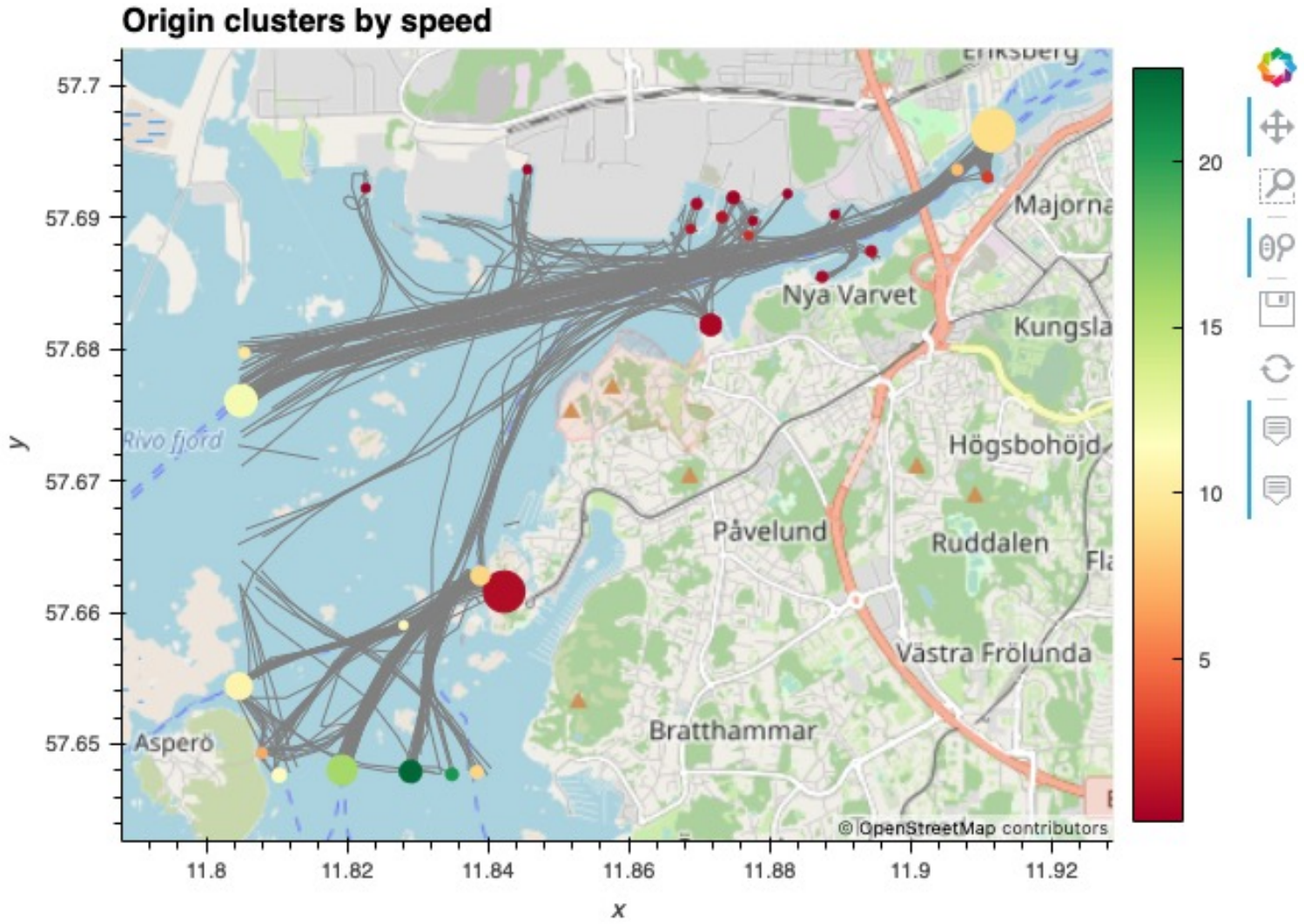


- Density based clusters with ball tree and haversine distance can describe regional patterns of ship activity

```
db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='haversine').fit(np.radians(matrix))
cluster_labels = db.labels_
num_clusters = len(set(cluster_labels))
clusters = pd.Series([matrix[cluster_labels == n] for n in range(num_clusters)])
origins['cluster'] = cluster_labels
def get_centermost_point(cluster):
    centroid = (MultiPoint(cluster).centroid.x, MultiPoint(cluster).centroid.y)
    centermost_point = min(cluster, key=lambda point: great_circle(point, centroid).m)
    return Point(tuple(centermost_point)[1], tuple(centermost_point)[0])
centermost_points = clusters.map(get_centermost_point)
origins.hvplot(title='Clustered origins', c='cluster', geo=True, tiles='OSM', cmap='glasbey_dark', frame_width=700, frame_height=500)
```



```
( trips.hvplot(title='Origin clusters by speed', color='gray', line_width=1, frame_width=700, frame_height=500) *
  GeoDataFrame(summary, crs=4326).hvplot(c='sog', size=np.sqrt(dim('n'))*3, geo=True,  cmap='RdYlGn')
)
```

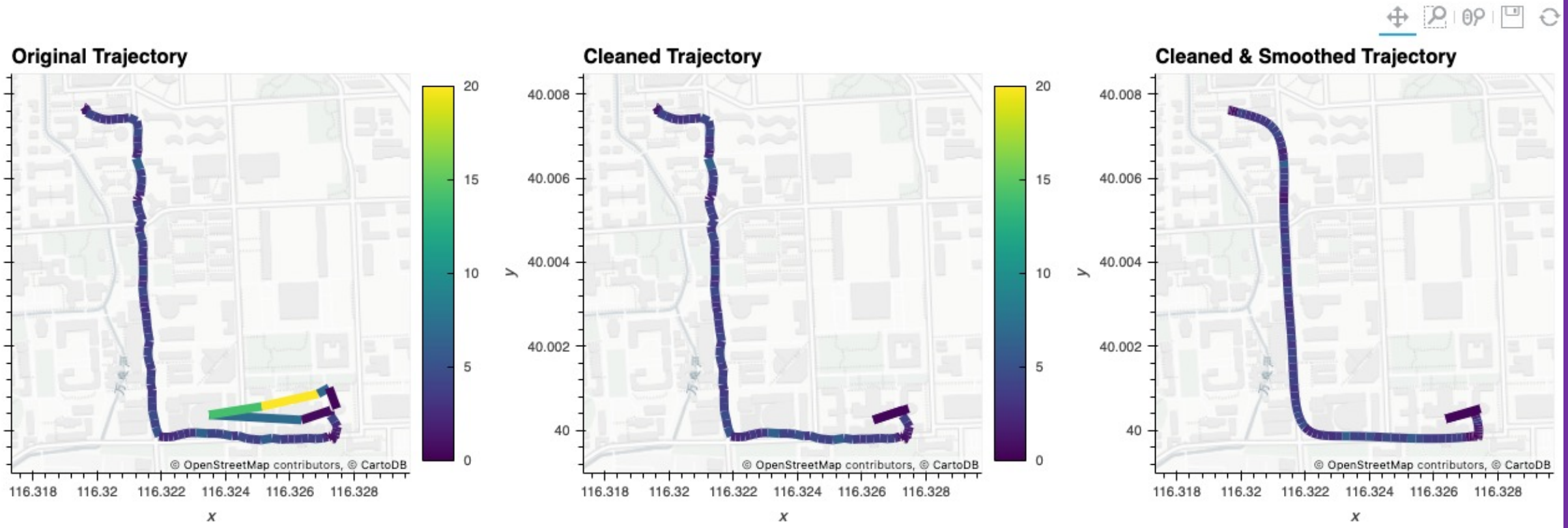


```
traj = split.trajectories[2]

cleaned = traj.copy()
cleaned.add_speed(overwrite=True)
for i in range(0,10):
    cleaned = mpd.OutlierCleaner(cleaned).clean({'speed': 1})

smoothed = mpd.KalmanSmootherCV(cleaned).smooth(process_noise_std=0.1, measurement_noise_std=10)

(traj.hvplot(title='Original Trajectory', **kwargs) +
 cleaned.hvplot(title='Cleaned Trajectory', **kwargs) +
 smoothed.hvplot(title='Cleaned & Smoothed Trajectory', **kwargs))
```



What's next for MovingPandas

- dask-movingpandas which uses dask-geopandas for big data
- Cast Trajectory to a TrajectoryCollection for Trajectory operations (similar to xarray Dataset and DataArray handling)
- Improved html repr
- Improved documentation
- Deployed dashboard(s) e.g. streamlit cloud…
- Share learnings/development between other trajectory libraries (CuSpatial, scikit-mobility)
- ???

Summary

- What is AIS data? – Ship location data
- What is movingpandas? – A libraries for trajectory data
- Whistle stop tour of geopandas – which moving pandas is build upon
- movingpandas demo including:
  - base class
  - cleaning AIS data – removing spurious points
  - voyage summary statistics – by splitting trajectories
  - Clustering of location hot spots – using end points with sklearn.cluster.DBSCAN
  - Trajectory smoothing – using stonesoup
- Future development of movingpandas – get involved!