GI_Forum 2019, Issue 1
Page: 54 - 68
Full Paper
Corresponding Author:
anita.graser@ait.ac.at
DOI: 10.1553/giscience2019_01_s54

MovingPandas: Efficient Structures for Movement Data in Python

Anita Graser

AIT Austrian Institute of Technology, Vienna, Austria
University of Salzburg, Austria

Abstract

Movement data analysis is a high-interest topic in many scientific domains. Even though Python is the scripting language of choice in the GIS world, currently there is no Python library that would enable researchers and practitioners to interact with and analyse movement data efficiently. To close this gap, we present MovingPandas, a new Python library for dealing with movement data. Its development is based on an analysis of state-of-the-art conceptual frameworks and existing implementations (in PostGIS, Hermes, and the R package *trajectories*). We describe how MovingPandas avoids limitations of Simple Feature-based movement data models commonly used to handle trajectories in the GIS world. Finally, we present the current state of the MovingPandas implementation and demonstrate its use in stand-alone Python scripts, as well as within the context of the desktop GIS application QGIS. This work represents the first step towards a general-purpose Python library that enables researchers and practitioners in the GIS field and beyond to handle and analyse movement data more efficiently

Keywords:

trajectory, spatio-temporal analysis, python, movement data analysis

1 Introduction

Movement data sources are highly heterogeneous. Datasets vary with respect to temporal resolution (frequent to sparse, regular or irregular), spatial resolution (fine to coarse), spatial dimensions (2D or 3D), movement constraints (network-constrained or not), movement models (Lagrangian or Eulerian), tracking system (cooperative or uncooperative), data size and privacy constraints. In geography, traditionally, movement data availability was often limited to information about flows between origins and destinations (OD flows). In contrast, modern data sources provide increasingly detailed episodic or quasi-continuous movement data (Andrienko et al., 2013). Movement data that goes beyond simple OD flows is commonly referred to as trajectory data. Demšar et al. (2015) define trajectory data as a discrete time series of measured locations. Trajectories can also stem from simulations (Loidl et al., 2016) or other movement data generators (Technitis et al., 2015). Since trajectories

appear in many different scientific domains (including physics, biology, ecology, chemistry, transport and logistics, astrophysics, remote sensing, and more), it is not surprising that the questions posed to trajectory data vary strongly across disciplines. Additionally, neither the terminology for writing about movement data analysis nor the core functionality of software tools for handling movement data has so far been well defined.

There are many libraries dealing with movement data, particularly in R (Klus & Pebesma, 2015). For example, Pebesma (2018) lists 27 R packages dealing with movement data. However, while Python is the scripting language of choice in both proprietary and open-source GIS environments, to the best of our knowledge there is currently no comparable Python library available that would enable researchers and GIS practitioners to efficiently interact with and analyse movement data. In this paper, we therefore introduce MovingPandas, an extension to the Python data analysis library Pandas (2018) and its spatial extension GeoPandas (2018), to add functionality for dealing with trajectory data.

The remainder of this paper is structured as follows: Section 2 describes the terminology and concepts currently used for describing trajectory data. We also analyse current conceptual frameworks and existing implementations in order to identify relevant core concepts and functionality for MovingPandas. Section 3 describes the current state of the MovingPandas implementation and demonstrates its usage. Finally, we discuss our plans for the further development of MovingPandas and how it fits into research agendas that focus on understanding and extracting knowledge from movement data.

2 State-of-the-art trajectory analysis

There is no consistent **terminology** in the field of movement data analysis. Depending on the research group and application domain, terms such as trajectory, track, path, moving point, move, travel and segment are used to describe the same or different concepts related to movement. Similarly, intervals without movement are referred to using terms such as stops, stays, events or activities. Finally, the individual data points are known, for example, as nodes, (spatio-temporal) positions, or locations.

Trajectory definitions vary. For example, Alvares et al. (2007) and Baglioni et al. (2009) define a trajectory as a sequence of moves and stops. According to Andrienko et al. (2013), the trajectory of a moving object is a function that defines a sequence of spatial positions (and thematic attributes) for a certain time interval. In contrast, for Spaccapietra et al. (2008) a trajectory is a segment of the spatio-temporal path covered by a moving object that represents a semantically meaningful unit of movement for the application. In other words, a trajectory is a travel (for some application-related purpose) from an initial starting point to the final destination. Similarly, in the R package *trajectories* (Klus & Pebesma, 2015), a track is meant to represent a series of consecutive location/timestamps that are not interrupted by another activity. Segments are connections between consecutive locations.

In addition to the pure measurements, there are also **semantics**. Vouros et al. (2018) present a trajectory ontology focusing on semantic trajectories. A trajectory is a temporal sequence of semantic nodes or trajectory segments. A semantic node specifies the position of a moving

object in a time period or at a particular instant, or a specific set of spatio-temporal positions of a single moving object; the node can be associated with contextual information. The authors also describe the concept of open trajectories, where the last semantic (terminal) node has not yet been reached, and closed trajectories, in which the last node is specified. A trajectory can also be classified as an intended trajectory (i.e. a planned or predicted one). Finally, Vouros et al. (2018) use hasParent and hasSuccessive properties to relate a trajectory to its parent and successive trajectories, respectively.

2.1 Data models

As well as choices in terminology, there exist a variety of approaches to represent trajectory data. Most approaches deal only with moving point data, while work on two-dimensional moving objects, such as moving regions, is rarer (Siabato et al., 2018). Nonetheless, moving regions, with application areas including modelling of forest fires, oil spills or the spread of invasive species, are highly relevant in the respective fields. For this paper, however, we focus on moving point data.

Geometries with timestamps (Simple Features-based trajectories) are the most common approach in GIS. This data model uses points or lines with timestamp attributes to preserve both spatial and temporal information. For example, the CSV (comma-separated value) data model used by the Movebank Data Repository (Wikelski & Kays, 2017) stores points with timestamps, while the OGC® Moving Features standard (OGC, 2017) uses lines with start and end timestamps.

Like the Moving Features standard, the R package *trajectories* (Pebesma et al., 2018) also implements the **line-based** approach: a track is meant to represent a single track followed by a person, animal or object, i.e. a series of consecutive location/timestamps that are not interrupted by another activity. The connections slot stores all the elements of a track in a data frame with a line on each row, with x0, y0, x1, y1 (the first four values) followed by attributes.

The temporal support for trajectories in PostGIS (Graser, 2018) uses a variation of the Simple Features concept. It stores movement data in **LineStringM** objects, where the measure variable M contains the temporal information. The function ST_IsValidTrajectory ensures that the temporal component increases from one vertex to the next. Matching a numerical M value to datetime requires application logic, since the database does not enforce any specific rules. Graser (2018) uses unixtime, but any other numerical representation of time could be used.

In the context of moving object databases, Pelekis et al. (2015) use the concept of **sliced representation** introduced by Güting et al. (2000). The key idea in this concept is to decompose movement into fragments, called slices, such that the movement within a slice can be described by a function. For example, Hermes supports first degree polynomials, circular arcs and the constant function. A moving point object is a collection of time periods and corresponding movement functions.

Finally, trajectories can be modelled as time series of locations. For example, Chen et al. (2005), building on established methods for one-dimensional time series, use the concept of

two-dimensional (xy) or three-dimensional (xyz) time series data for trajectories to derive similarity measures.

2.2 Functions for trajectories

Trajectory analysis encompasses many different analysis types, such as segmentation, similarity analysis, clustering, outlier detection, classification, hotspot detection, pattern detection, and flock/group/herd detection. All these analysis types depend on measures defined on trajectories (Wiratma et al., 2017). Trajectory functions enable us to compute these measures. As with terminology and concepts, however, there is no consistent framework for the classification of trajectory functions. Recent work includes Dodge et al.'s (2008) framework of movement and classification of movement patterns, and Wiratma et al.'s (2017) measures for trajectories and groups of trajectories.

On the technological side, we will have a closer look at implemented functions for movement data in the spatial database PostGIS (Graser 2018), the moving object database Hermes (Pelekis et al., 2015), and the R package *trajectories* (Pebesma et al., 2018; Moradi et al., 2018 preprint). Since it is clear that each implementation was created for a different research or application focus, it is challenging to devise a systematic and objective means of comparison. It is furthermore worth noting that Hermes syntax and functionality differ between the Oracle implementation (Hermes@Oracle) described in Pelekis et al. (2015) and the PostGIS implementation (Hermes@PostGIS) documented in Hermes (2017). The Hermes functionality described in this paper is a superset of the functions documented in the two sources.

To characterize **movement measures or parameters for individual moving objects**, we use the framework devised by Dodge et al. (2008). This framework distinguishes three parameter groups (primitive, primary and secondary derivatives) in three dimensions (spatial, temporal and spatio-temporal). Wiratma et al. (2017) call these 'measures for a trajectory in isolation'. Furthermore, we distinguish between global measures that describe the trajectory as a whole and local measures that describe only parts of the trajectory. This is very much in line with Pelekis et al. (2015), who state that most Hermes operations come in two versions: the first is related to a user-defined point in time, while the second is time-independent. Trajectory implementations should support access to, or computation of, these parameters. Table 1 provides an overview of the coverage of these measures in PostGIS, Hermes and R *trajectories*.

Position (x,y) and instance or interval (t) are the spatial and temporal primitives. The PostGIS trajectory object provides direct access to the positions of the whole trajectory. In Hermes, the function `f_trajectory()` projects the moving point representation to a geometry on the Cartesian plane. R *trajectories* provide an `@sp` slot. To access the local position at a certain point in time, Hermes `at_instant(t)` and PostGIS `ST_LocateAlong(trajectory,t)` return the position of the moving object at time t. In R *trajectories*, `approxTrack()` is used to access interpolated positions along a trajectory. It supports different interpolators, including straight line and smooth or not smooth splines.

To access the local time, PostGIS `ST_InterpolatePoint(trajectory,point)` returns the m-value along the trajectory that is closest to the point provided. Similarly, Hermes

provides `atPoint(trajectory,point)`, as well as `Moving_Point.at_linestring-(linestring)`. R *trajectories* does not support accessing the time at a certain location.

Spatial primary derivatives are distance, direction and spatial extent. These derivatives can be measured in different ways. Distance, for example, can be measured for the whole trajectory (PostGIS `ST_Length(trajectory)`; Hermes `Length(trajectory)`), or between the trajectory start and a specific point along the trajectory (Hermes `f_length(tolerance,t)`). Similarly, direction can be measured as average direction (Hermes `anglexxavg(trajectory)`), or as local direction at a specific point along the trajectory (PostGIS `ST_Azimuth(pointA,pointB)`; Hermes `f_direction(Moving_Point,t)`). Finally, spatial extent can be interpreted in many ways, including a simple bounding box, or, for example, Wiratma et al.'s (2017) measure of area covered by a trajectory (using a disc of radius r). In PostGIS, this can be computed using `ST_Buffer(trajectory)` or `ST_Envelope(trajectory)`. R *trajectories* offers access to summary statistics, such as distance and average speed, as well as spatial (`bbox`) and spatio-temporal (`stbox`) extent.

Temporal primary derivatives are duration (period of time in which a movement is observed) and travel time. PostGIS does not provide a built-in function to directly access duration. Therefore, this measure must be computed from the timestamps of the trajectory start and end points. Hermes provides `duration(trajectory).` The R *trajectories* `summary` provides minimum and maximum times.

Spatio-temporal primary derivatives are speed (rate of change of the object's position) and velocity (rate of change of position and direction). Hermes provides `Moving_Point.f_speed(t)` and `averageSpeed(trajectory).` In PostGIS, there are no built-in functions to compute speed or velocity. In R trajectories, `Trajectory@connections` provides access to local speed information.

In the implementations investigated, there are almost no built-in functions to directly compute secondary derivatives as described by Dodge et al. (2008). Spatial secondary derivatives are spatial distribution, change of direction, and sinuosity (also known as detour or straightness index) (Wiratma et al., 2017). Temporal secondary derivatives are temporal distribution and change of duration. Spatio-temporal secondary derivatives are acceleration and approaching rate. The closest thing to the spatial distribution measure is the R *trajectories* statistical functions to detect trajectory patterns based on point patterns using, for example, `density` and `idw`.

Table 1: Comparison of built-in functions for direct access to measures of individual moving objects for the whole trajectory 'globally' (G) and locally (L)

			PostGIS	**HERMES**	**R trajectories**
Spatial primitives	Position (x,y)	G	trajectory	Moving_Point .f_trajectory()	trajectory@sp as(trajectory, "Spatial")
		L	ST_LocateAlong (trajectory,t) ST_LocateBetween (trajectory,t1,t2)	atInstant (trajectory,t) atPeriod (trajectory,perio d)	
Temporal primitives	Instance or interval (t)	G	trajectory		index(trajector y)
		L	ST_InterpolatePoint (trajectory,point)	atPoint (trajectory,point) Moving_Point .at_linestring (linestring)	
Spatial primary derivative s	Distance	G	ST_Length (trajectory)	length (trajectory)	summary(traject ory) trajectory @connections
		L		Moving_Point .f_length (tolerance,t)	
	Direction	G	ST_Azimuth (pointA,pointB)	anglexxavg (trajectory) anglexx(trajector y)	Summary (trajectory)
		L	ST_Azimuth (pointA,pointB)	f_direction (Moving_Point,t) anglexx(segment)	Trajectory @connections
	Spatial extent	G	ST_Buffer (trajectory) ST_Envelope (trajectory)		trajectory.bbox or trajectory.stbo x
Temporal primary derivates	Duration	G		duration (trajectory)	Summary (trajectory)
Spatio-temporal primary derivates	Speed	G		averageSpeed (trajectory)	Summary (trajectory)
		L		Moving_Point .f_speed(t)	
	Velocity				

Measures for multiple moving objects can be divided into measures for one trajectory among other trajectories, measures for a single group of trajectories, and measures between groups of trajectories (Wiratma et al., 2017).

Pairwise similarity or distance has been studied extensively. Well-known geometric similarity measures are the Fréchet distance and the Hausdorff distance, which are implemented in R *trajectories* (`dists, frechetDist`) and PostGIS (`ST_HausdorffDistance`), respectively. Time-aware trajectory similarity measures include time-focused distance, dynamic time-warping distance, and the edit distance (Wiratma et al., 2017). The moving objects database Hermes provides a series of distance measures for trajectories, including Generalized SpatioTemporal Locality Inbetween Polylines (`GenSTLIP`), Generalized Speed-Pattern STLIP (`GenSPSTLIP`), Generalized Acceleration-Pattern STLIP (`GenACSTLIP`), Directional Distance (`DDIST`), and Temporal DDIST (`TDDIST`). In PostGIS, time-aware distance measures are limited to closest points of approach (CPA) measures (`ST_ClosestPointOfApproach`, `ST_DistanceCPA`, `ST_CPAWithin`).

Measures for a single group of trajectories include area covered, size (number of trajectories), density, and formation stability. Finally, measures between groups of trajectories include group similarity, closeness and centrality (Wiratma et al., 2017).

Working with trajectories also requires **trajectory data manipulation functions**, for example to clip trajectories to an area of interest, or to annotate trajectories with context information. Hermes supports intersection overlays (`intersection`) to extract the portion of the moving point inside a given region. PostGIS's intersection currently drops m-values and therefore makes it harder to create annotated trajectories (Westermeier 2018).

Range queries to extract trajectories that are fully contained within a given spatio-temporal window are supported by Hermes (`TB_MP_In_SpatioTemporal_Window`) and PostGIS (`&&& operator`). Similarly, Hermes also directly supports topological queries to extract trajectories that enter and/or leave a certain area (`TB_Topological_Query`). In PostGIS, the same can be achieved by checking whether the start and/or end points fall within those areas. Convenient spatio-temporal nearest-neighbour queries are supported by Hermes (`IncPointNNSearch`, `IncTrajectoryNNsearch`).

Other trajectory data manipulation functions are segmentation functions to split the raw location stream into meaningful trajectories and stops, functions for downsampling, and functions for generalizing trajectory data. R *trajectories* implements `downsample` (temporal) and `generalize` (spatio-temporal).

An important topic, particularly from a geospatial perspective, is the handling of geographic coordinates. Hermes is designed to work with data in Euclidean space. Vodas (2013) describes how Hermes implements transformation of Geographic to/from Topocentric conversion (EPSG 9837). PostGIS, on the other hand, supports a wide variety of coordinate reference systems and can handle planar, spherical and ellipsoidal computations. R *trajectories* also supports both Cartesian and geodetic coordinates.

Finally, R *trajectories* also implements **visualization** functions (`stcube`, `stplot`), while PostGIS trajectories can be visualized in Desktop GIS. Hermes can export its trajectories for visualization purposes.

This analysis of existing conceptual frameworks and implementations provides an overview of core functionality for movement data analysis libraries. The presentation of possible implementations in PostGIS, Hermes and R *trajectories* is not meant to be exhaustive. Nonetheless, the overview presents a solid base to determine the necessary core functionality for the development of our own movement data analysis library.

3 MovingPandas

Common Simple Features-based data models where trajectories consist of geometries with timestamps can be readily implemented in GIS environments, but they suffer from a lack of support for the temporal dimension, such as functions for duration and speed.

In stark contrast, the Pandas (2018) data analysis library has been developed with a strong focus on time series. By choosing Pandas data structures (1D series and 2D DataFrames) as a base for MovingPandas, we gain access to the library's built-in functionality, including: flexible indexing on timestamps and other column types; memory-efficient sparse data structures for data that is mostly missing or mostly constant; an integrated 'group by' engine for aggregating and transforming datasets, and moving window statistics (rolling mean, rolling standard deviation, etc.).

GeoPandas (2018) extends the data types that can be used in Pandas DataFrames, thus creating GeoDataFrames. Geometric operations on these spatial data types are performed by Shapely (2018). Geopandas further depends on Fiona (2018) for file access (which enables direct reading of GeoDataFrames from common spatial file formats, such as GeoPackage or Shapefile), and descartes and matplotlib for plotting.

The source code of MovingPandas is available on Github (https://github.com/anitagraser/movingpandas). MovingPandas uses the following terminology. A *trajectory* is, or more correctly has, a time-ordered series of geometries. These geometries and associated attributes are stored in a GeoDataFrame *df*, as shown in Figure 1. Furthermore, a trajectory can have a *parent* trajectory and can itself be the parent of successive trajectories. Raw unsegmented streams of movement data, as well as semantically meaningful subsections or other subsections, can therefore be represented as trajectories. Depending on the use case, the trajectory object can access a point-based or a line-based representation of its data.

Trajectory
id: string df: GeoDataFrame crs: string parent: Trajectory
__str__() set_crs(crs) has_parent(): boolean is_latlon(): boolean to_linestring(): shapely.geometry.LineString to_linestringm_wkt(): string get_start_location(): shapely.geometry.Point get_end_location(): shapely.geometry.Point get_bbox(): (minx, miny, maxx, maxy) tuple get_start_time(): datetime get_end_time(): datetime get_duration(): timedelta get_length(): float get_direction(): float get_row_at(timestamp, method='nearest'): pandas.Series get_position_at(timestamp, method='nearest'): shapely.geometry.Point interpolate_position_at(timestamp): shapely.geometry.Point get_linestring_between(timestamp1, timestamp2): shapely.geometry.LineString get_segment_between(timestamp1, timestamp2): Trajectory add_direction() add_speed() make_line(df): shapely.geometry.LineString clip(shapely.geometry.polygon): Trajectory intersection(fiona.feature): Trajectory

Figure 1: Trajectory class diagram

The functionality currently implemented in MovingPandas covers most of the primitive and primary derivative measures for individual moving objects defined by Dodge et al. (2008), as listed in Table 2. The functions for computing speed and direction (`add_speed()` and `add_direction()`) add a new column to the trajectory's GeoDataFrame. Individual local values can be accessed using `get_row_at(timestamp, method='nearest')`. Global or window statistics can be computed using appropriate aggregations.

Table 2: MovingPandas functionality

Spatial primitives	Position (x,y)	G	to_linestring(): shapely.geometry.LineString
		L	get_position_at(timestamp, method='nearest'): shapely.geometry.Point get_segment_between(timestamp1, timestamp2): Trajectory
Temporal primitives	Instance or interval (t)		< not implemented >
Spatial primary derivatives	Distance	G	get_length(): float
	Direction	G	get_direction(): float
		L	add_direction()
	Spatial extent	G	get_bbox(): (minx, miny, maxx, maxy) tuple
Temporal primary derivates	Duration	G	get_duration(): timedelta
Spatio-temporal primary derivates	Speed	L	add_speed()
	Velocity		< not implemented >

In addition, it is possible to clip trajectories with polygons, and to compute intersections with polygons. Like GeoDataFrames, trajectories come with coordinate reference system information. Geometric operations on geographic coordinates use spherical geometry, while operations on projected coordinates use planar geometry. All distances are computed in two dimensions, and no specific 3D functions have so far been implemented. To handle 3D trajectories properly, it will be necessary to provide appropriate ellipsoidal geometry functions.

3.1 Application examples

This section demonstrates how MovingPandas can be used to handle movement data in stand-alone Python scripts, as well as within the desktop GIS application QGIS. The two application examples in this section use the Geolife dataset published by Zheng, Li et al. (2008), Zhen, Zhang et al. (2009), and Zhen, Xie et al. (2010).

The first example, shown in the Listing, illustrates how the movement data of multiple moving objects can be read from a common spatial data file format (GeoPackage), converted to trajectory objects, and finally clipped by a polygon. First, the content of the GeoPackage is read into a GeoDataFrame and the index is set to the time attribute. Then, the GeoDataFrame is grouped by trajectory ID and each resulting grouping is converted to a trajectory. Finally, each trajectory is clipped by a polygon.

Listing: Reading trajectory data from GeoPackage and clipping by a polygon

```
>>> import os
>>> import sys
>>> import pandas as pd
>>> from geopandas import read_file
>>> from shapely.geometry import Polygon
>>> from trajectory import Trajectory

>>> xmin, xmax, ymin, ymax = 116.3685035,116.3702945,39.904675,39.907728
>>> polygon = Polygon([(xmin,ymin), (xmin,ymax), (xmax,ymax), (xmax,ymin), (xmin,ymin)])

>>> df = read_file(os.path.join(script_path,'demodata_geolife.gpkg'))
>>> df['t'] = pd.to_datetime(df['t'])
>>> df = df.set_index('t')
>>> print("Finished reading {} rows".format(len(df)))

Finished reading 5908 rows

>>> trajectories = []
>>> for key, values in df.groupby(['trajectory_id']):
>>>     trajectory = Trajectory(key, values)
>>>     print(trajectory)
>>>     trajectories.append(trajectory)
>>> print("Finished creating {} trajectories".format(len(trajectories)))

Trajectory 1 (2008-12-11 04:42:14 to 2008-12-11 05:15:46) | Size: 466 | Length: 6210.1m
LINESTRING (116.391305 39.898573, 116.391317 39.898617, 116.390928 39.898613, ...
Trajectory 2 (2009-06-29 07:02:25 to 2009-06-29 11:13:12) | Size: 897 | Length: 38728.7m
LINESTRING (116.590957 40.071961, 116.590905 40.072007, 116.590879 40.072027, ...
Trajectory 3 (2009-02-04 04:32:53 to 2009-02-04 11:20:12) | Size: 1810 | Length: 12739.2m
LINESTRING (116.385689 39.899773, 116.385654 39.899651, 116.385548 39.899699, ...
...

Finished creating 5 trajectories

>>> intersections = []
>>> for key, values in df.groupby(['trajectory_id']):
>>>     traj = Trajectory(key, values)
>>>     for intersection in traj.clip(polygon):
>>>         intersections.append(intersection)
>>> print("Found {} intersections".format(len(intersections)))

Found 3 intersections
```

To further demonstrate the usefulness of MovingPandas in the context of desktop GIS applications, we implemented a QGIS plugin called Trajectools (https://plugins.qgis.org/plugins/processing_trajectory/) building upon MovingPandas. Figure 2 shows a screenshot of this second application, with results of executing the tools to add direction and speed information to a layer containing Geolife data. Figure 3 shows the alternative line-based visualization using the LineStringM trajectory representation.

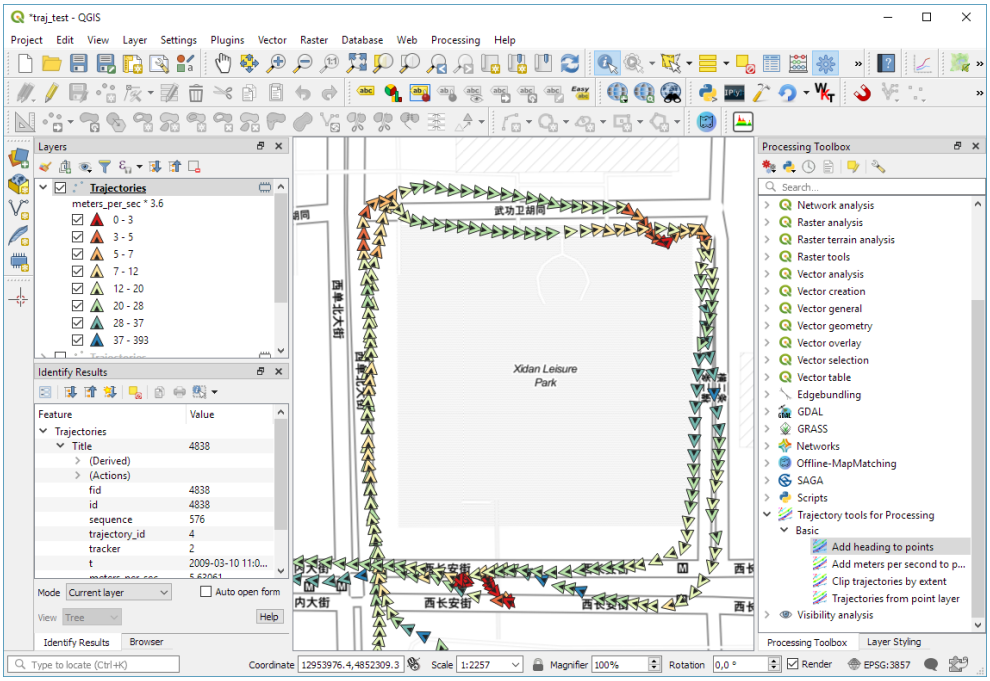

Figure 2: MovingPandas integration in QGIS, showing added direction and speed information

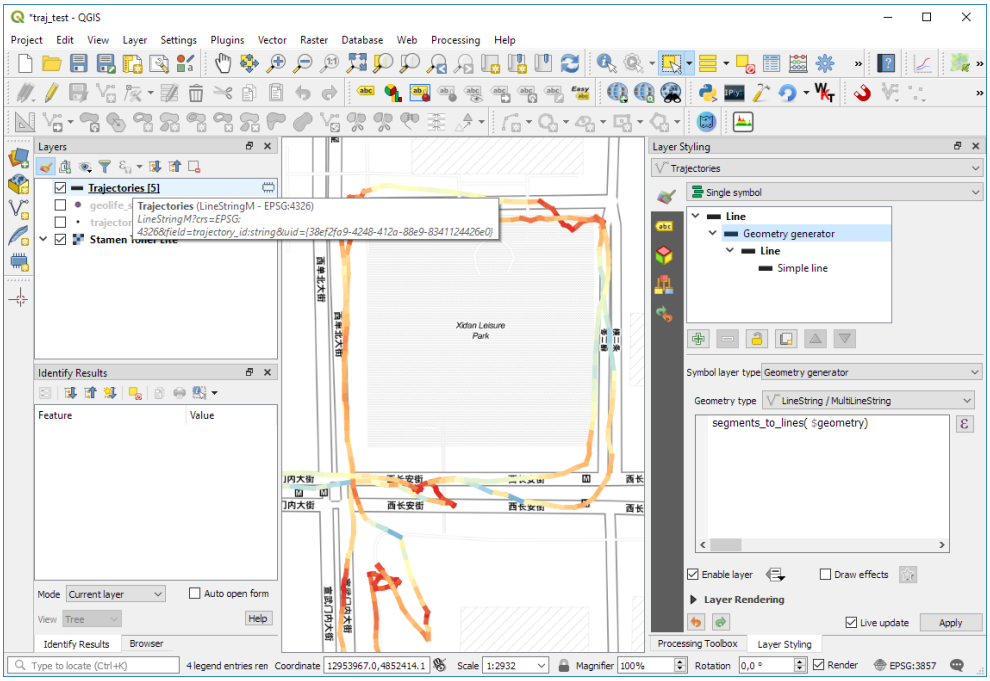

Figure 3: MovingPandas integration in QGIS showing the LineStringM trajectory representation styled by speed.

4 Conclusions and future work

In this paper, we presented MovingPandas, a new Python library for dealing with movement data based on the Pandas data analysis library and GeoPandas extension. We laid out how MovingPandas differs from other trajectory data modelling approaches commonly used in the GIS world. We also analysed existing conceptual frameworks and implementations (in the spatial database PostGIS, the moving objects database Hermes, and the R package *trajectories*) to determine the necessary core functionality for a movement data analysis library. Finally, we documented the current state of the MovingPandas implementation and illustrated its usefulness in stand-alone Python scripts, as well as within the context of the desktop GIS application QGIS. While MovingPandas currently supports only moving point objects, the underlying GeoDataFrame could also store polygons for moving area objects.

Current research agendas that focus on understanding and extracting knowledge from movement data emphasize challenges related to volume, velocity and variety (Georgiou et al., 2018). The challenge of volume deals with identifying effective methods for creating overviews (Robinson et al., 2017) and analysis of large, complex, movement data (Demšar et al., 2015; Mazimpaka & Timpf, 2016; Georgiou et al., 2018). The challenge of velocity deals with developing methods to deal with streams of movement data in which data about the same object can come from multiple sources (Georgiou et al., 2018). (In the big data literature, 'velocity' therefore has a different meaning from the one it has in the movement data analysis literature, where it refers to changes in distance and direction over time.) The challenge of variety deals with the integration of different data types from heterogeneous data sources (Mazimpaka & Timpf, 2016; Georgiou et al., 2018), the development of cross-scale trajectory data mining (Mazimpaka & Timpf, 2016), and the use of these datasets for movement prediction (Georgiou et al., 2018).

Ongoing MovingPandas development focuses on the implementation of trajectory sampling and prediction methods. Our plans for the further development of MovingPandas include pairwise distance measures, as well as measures for groups of trajectories. Existing implementations (Burq, 2018) demonstrate that GeoPandas can be used in the context of the Spark distributed processing environment. Future work will therefore investigate whether MovingPandas can be used efficiently with Spark to tackle the challenge of volume.

Acknowledgements

This work was supported by the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT) within the programme 'IKT der Zukunft' under Grant 861258 (project MARNG).

References

Alvares, L. O., Bogorny, V., Kuijpers, B., de Macedo, J. A. F., Moelans, B., & Vaisman, A. (2007). A model for enriching trajectories with semantic geographical information. In Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems (p. 22). ACM.

Andrienko, G., Andrienko, N., Bak, P., Keim, D., & Wrobel, S. (2013). Visual analytics of movement. Springer Science & Business Media.

Baglioni, M., de Macêdo, J. A. F., Renso, C., Trasarti, R., & Wachowicz, M. (2009). Towards semantic interpretation of movement behavior. In Advances in GIScience (pp. 271-288). Springer, Berlin, Heidelberg.

Burq, S. (2018). Github respository: sabman/PySparkGeoAnalysis. Retrieved from https://github.com/sabman/PySparkGeoAnalysis/blob/master/003-geopandas-and-spark.ipynb

Chen, L., Özsu, M. T., & Oria, V. (2005). Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data (pp. 491-502). ACM.

Demšar, U., Buchin, K., Cagnacci, F., Safi, K., Speckmann, B., Van de Weghe, N., Weiskopf, D., & Weibel, R. (2015). Analysis and visualisation of movement: an interdisciplinary review. *Movement ecology*, *3*(1), 5.

Dodge, S., Weibel, R., & Lautenschütz, A. K. (2008). Towards a taxonomy of movement patterns. *Information visualization*, *7*(3-4), 240-252.

Fiona developers (2018). Fiona documentation. Retrieved from https://fiona.readthedocs.io/en/latest/

GeoPandas developers (2018). GeoPandas documentation. Retrieved from https://geopandas.readthedocs.io/en/latest/

Georgiou, H., Karagiorgou, S., Kontoulis, Y., Pelekis, N., Petrou, P., Scarlatti, D., & Theodoridis, Y. (2018). Moving Objects Analytics: Survey on Future Location & Trajectory Prediction Methods. *arXiv preprint arXiv:1807.04639*.

Graser, A. (2018). Evaluating Spatio-temporal Data Models for Trajectories in PostGIS Databases. GI_Forum – Journal of Geographic Information Science, 1-2018, 16-33.

Güting, R.H., Bohlen, M.H., Erwig, M., Jensen, C.S., Lorentzos, N.A., Schneider, M., & Vazirgiannis, M. (2000). A foundation for representing and querying moving objects. ACM Transactions on Database Systems, 25(1), 1-42.

Hermes developers (2017) Hermes documentation. Retrieved from http://infolab.cs.unipi.gr/hermes/postgresql/doc/bin/html/functionality.html

Klus, B., & Pebesma, E. (2015). Analysing Trajectory Data in R. Vignette. Retrieved from http://www.et.bs.ehu.es/cran/web/packages/trajectories/vignettes/tracks.pdf

Loidl, M., Wallentin, G., Cyganski, R., Graser, A., Scholz, J., & Haslauer, E. (2016). GIS and transport modeling—Strengthening the spatial perspective. ISPRS Int. J. Geo-Inf. 2016, 5, 84.

Mazimpaka, J. D., & Timpf, S. (2016). Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, *2016*(13), 61-99.

Moradi, M.M., Pebesma, E., & Mateu, J. (2018 preprint). trajectories: Classes and Methods for Trajectory Data. Journal of Statistical Software. Retrieved from https://cran.r-project.org/web/packages/trajectories/vignettes/article.pdf

OGC Open Geospatial Consortium Inc. (2011). OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, Version: 1.2.1. Retrieved from http://www.opengeospatial.org/standards/sfa

OGC Open Geospatial Consortium Inc. (2017). OGC® Moving Features. Retrieved from http://www.opengeospatial.org/standards/movingfeatures

Pandas developers (2018) Pandas documentation. Retrieved from http://pandas.pydata.org/pandas-docs/stable/overview.html

Pebesma, E. (2012). spacetime: Spatio-Temporal Data in R. Journal of Statistical Software, 51(7):1-30.

Pebesma (2018). CRAN Task View: Handling and Analyzing Spatio-Temporal Data. Retrieved from https://cran.r-project.org/web/views/SpatioTemporal.html

Pebesma, E., Klus, B., Graeler, B., Gorte, N., & Moradi, M. (2018). Classes and Methods for Trajectory Data, Version 0.2-1, R package. Retrieved from https://cran.r-project.org/web/packages/trajectories/trajectories.pdf

Pelekis, N., Frentzos, E., Giatrakos, N., & Theodoridis, Y. (2015). HERMES: A trajectory DB engine for mobility-centric applications. International Journal of Knowledge-Based Organizations (IJKBO), 5(2), 19-41.

Robinson, A. C., Demšar, U., Moore, A. B., Buckley, A., Jiang, B., Field, K., Kraak, M.J., Cambolm S.P., & Sluter, C. R. (2017). Geospatial big data and cartography: research challenges and opportunities for making maps that matter. *International Journal of Cartography*, 1-29.

Shapely developers (2018) Shapely documentation. Retrieved from https://shapely.readthedocs.io/en/latest/

Siabato, W., Claramunt, C., Ilarri, S., & Manso-Callejo, M. Á. (2018). A Survey of Modelling Trends in Temporal GIS. *ACM Computing Surveys (CSUR)*, *51*(2), 30.

Spaccapietra, S., Parent, C., Damiani, M.L., de Macedo, J.A., Porto, F., & Vangenot, C. (2008). A Conceptual View on Trajectories. Data and Knowledge Engineering 65: 126–46.

Technitis, G., Othman, W., Safi, K., & Weibel, R. (2015). From A to B, randomly: a point-to-point random trajectory generator for animal movement. International Journal of Geographical Information Science, 29(6), 912-934.

Vodas, M. (2013). Hermes - Building an Efficient Moving Object Database Engine. Master Thesis, University of Piraeus. Retrieved from http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/8447/Vodas_Marios.pdf

Vouros, G. A., Doulkeridis, C., Santipantakis, G., & Vlachou, A. (2018). Taming Big Maritime Data to Support Analytics. In Information Fusion and Intelligent Geographic Information Systems (IF&IGIS'17) (pp. 15-27). Springer, Cham.

Westermeier, E.M. (2018). Contextual Trajectory Modeling and Analysis. Master Thesis, Interfaculty Department of Geoinformatics, University of Salzburg.

Wikelski, M., & Kays, R. (2017). Movebank: archive, analysis and sharing of animal movement data. Hosted by the Max Planck Institute for Ornithology. Retrieved from http://www.movebank.org/

Wiratma, L., van Kreveld, M., & Löffler, M. (2017). On Measures for Groups of Trajectories. In: Bregt A., Sarjakoski T., van Lammeren R., Rip F. (eds) Societal Geo-innovation. AGILE 2017. Lecture Notes in Geoinformation and Cartography. Springer, Cham.

Zheng, Y., Li, Q., Chen, Y., Xie, X., & Ma, W.-Y. (2008). Understanding mobility based on GPS data. In Proceedings of the 10th international conference on Ubiquitous computing, 312–321. ACM.

Zheng, Y., Xie, X., & Ma, W.-Y. (2010). GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. IEEE Data Eng. Bull., 33(2):32–39.

Zheng, Y., Zhang, L., Xie, X., & Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. In Proceedings of the 18th international conference on World wide web, 791–800. ACM.