Modeling the global shipping trade using a Python-based analysis stack

Shaun Walbridge, University of California, Santa Barbara; now Esri

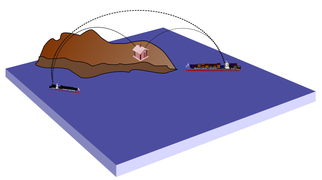
Abstract

Shipping accounts for 90% of global trade volume, and plays far-reaching economic and ecological roles. Despite this importance, little data is publicly available. This work uses `requests`, `BeautifulSoup` and `lxml` to download and parse billions of ship-based observations collected over radio frequencies using the Automatic Identification System (AIS), initially stored in KML files. These raw observations were filtered into a PostGIS database, which provides geographic primitives and spatial indexing. The data was then validated using probabilistic record linkage and fuzzy string matching using `jellyfish` to correlate between data sources. Finally, the data was geographically filtered by applying methods from `geographiclib`, `geopy` and GDAL in a parallel environment to create a validated model of global ship movement. A few specific ecological use cases of this data will be discussed, built using spatial Python tools, such as modeling the effect of ship strikes, or ship-whale collisions, based on computed vessel types and velocities, and examining noise pollution from vessel traffic. As streaming data platforms come online, vessel movement can be analyzed in real-time within a geospatial analysis workflow, opening the possibility for dynamic management of ocean resources, which requires integration between scientific computing and heterogeneous systems that map well onto Python's existing ecosystem, some implications of which will be discussed.

Background

Shipping, the global transportation of people and goods, moves most world trade, and understanding its effects is required to assess human usage of the oceans. This work examines the shipping trade by combining global observations of ship location with vessel identification records. These records are streamed from KML web services, parsed into a spatial database, validated using quality checking methods, routed on a global ship visibility graph, resulting in a high resolution data set which links individual vessels with their movement patterns and attributes.

Automated Information System (AIS)



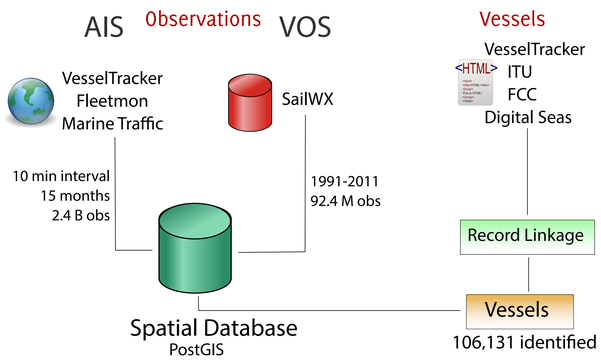
Attributes:
ship id
(x, y, z)
heading
type
speed

Ship: PURKI
Country: Malta
IMO: 9004217
MMSI: 248495000
Callsign: 9HA2386
Length: 147m
Width: 26m
Type: cargo ship
Obs: 24225

Analysis

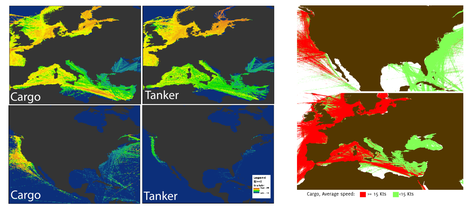
Many modern scientific analysis require spanning many domains and combining analytical tools from a variety of disciplines. Python is ideally suited to serve this role, with both a solid culture of software development with packages spanning the full breadth of problem domains, and a special focus on the needs of scientists, both through Python packages, and development of wrappers and extensions to the existing robust software written in the common C / C++ / Fortran stack which underpins CPython.

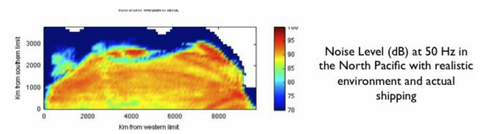
In this work, we used Python for every step of our analysis:
- Scraping, using urllib2 and requests.
- Numerous sources of vessel record data were merged using fuzzy string matching and approaches from the record linkage community. In particular, the `jellyfish` library proved useful in Python for its implementation of the Jaro-Winkler formula for string relatedness.



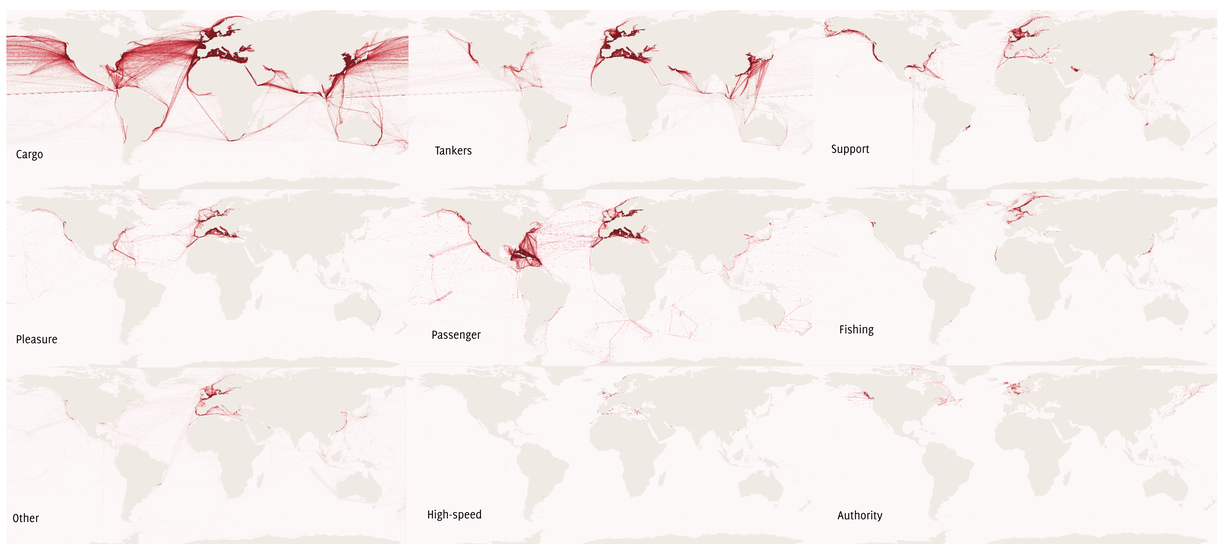
Type	Vessels *(AIS)*	Vessels *(VOS)*	fleet size	coverage (%)	observations (M)
Cargo	25214	5838	33392[1]	75.6	665.45
Tanker	9758	2375	14068[1]	69.4	264.42
Passenger	4007	777	6370[1]	62.9	142.16
Support	9954	735	25234[1]	39.4	298.02
High-speed	404	81	1178	34.3	2.52
Fishing	11186	349	51200[3]	21.8	6.87
Pleasure	20727	661	800,000[2]	2.59	267.48
Other	9507	1400	–	–	4.75
Authority	656	55	–	–	1.44

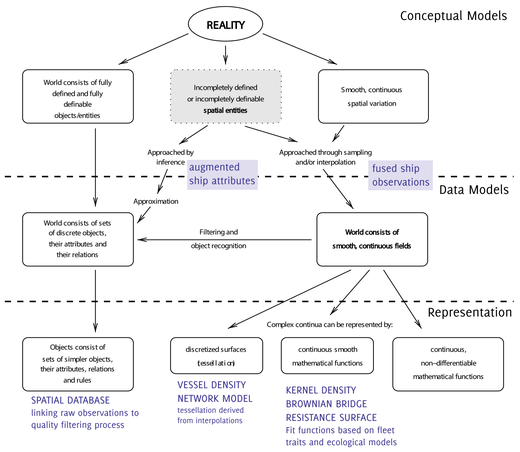
- Parsing was performed using lxml to parse XML documents. Here, this was used to parse KML documents representing ship locations at specific times, and collected over 14 months with a cron job. These parsed results were then sent on to the open source object-relational database, PostgreSQL, backed by PostGIS, using the roust psycopg2 library for database connectivity directly from Python. Similarly, records about individual vessels, contained in HTML documents were parsed using the BeautifulSoup library.





Sample uses of this model for mitigating impacts on marine mega-fauna

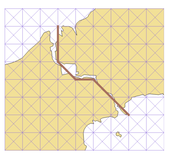




- There is no better environment for geospatial analysis than Python: it supports a rich set of core libraries for manipulating geospatial data and performing analysis, supplemented by the fact that most major GIS systems use Python as their language of choice (e.g. ArcGIS, QGIS, GRASS GIS). Here, we used GDAL, geographiclib, GeoJSON, GRASS GIS, Rtree, and Shapely to perform needed spatial analysis.
- Network analysis is also a domain well represented by Python: in addition to the pure-Python NetworkX library, numerous wrappers to memory and CPU efficient network representations are available. Here, the graph_tool binding to the Boost Graphing Library proved useful for its efficient representation of graph structures.

Routing Network

Both the AIS and VOS data have limited observation frequency, leading to gaps in data that when directly interpolated with geodesic paths, create invalid paths which cross land masses. Here, a routing model was used to create a visibility graph of the oceans, creating valid potential movement paths. Initially, the pure Python `NetworkX` package was used, but performance limitations made analysis infeasible. Here, I used the `graph_tools` package, a wrapper of the C++ Boost Graph Library. The graph used has 6.5M vertices (valid ship locations) and 17M edges (connections between locations). Further performance gains from using the `Rtree` package to store node locations, which can perform lookups on multidimensional indicies.



Panama Canal routing showing valid path on uncorrected visibility graph

Details at:
https://github.com/scw/thesis