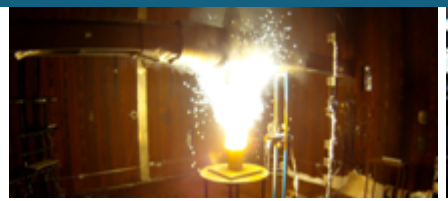
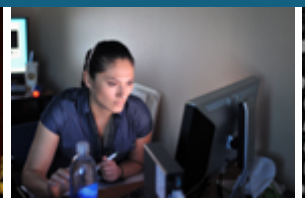
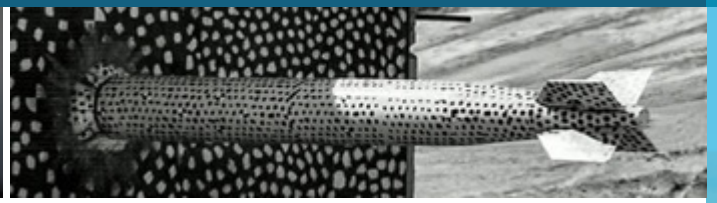
SAND2020-12646C

Sandia National Laboratories

ML for Trajectories: Bridging the Gap from Computer to Analyst with Tracktable







Presented by

Dr. Mark D. Rintoul

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Trajectory Analytics: Machine Learning + Shapes + Time

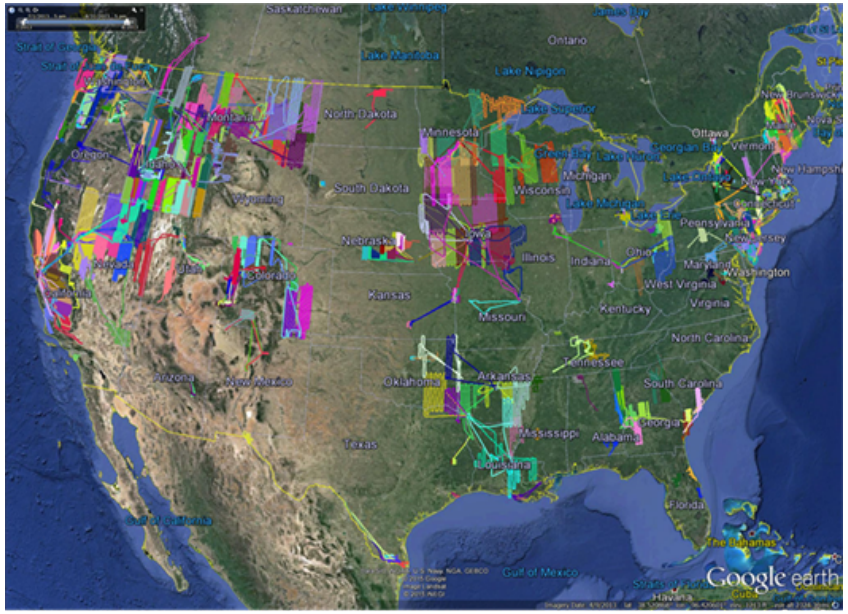
This is the golden age for trajectory analytics

- Machine Learning going through explosive growth
- Rapidly increasing data sets
- No shortage of applications

However, we are still doing many things by hand

The next generation of work in trajectory analytics will be focused on **using computer to bridge the gap between the analyst and the analytics**

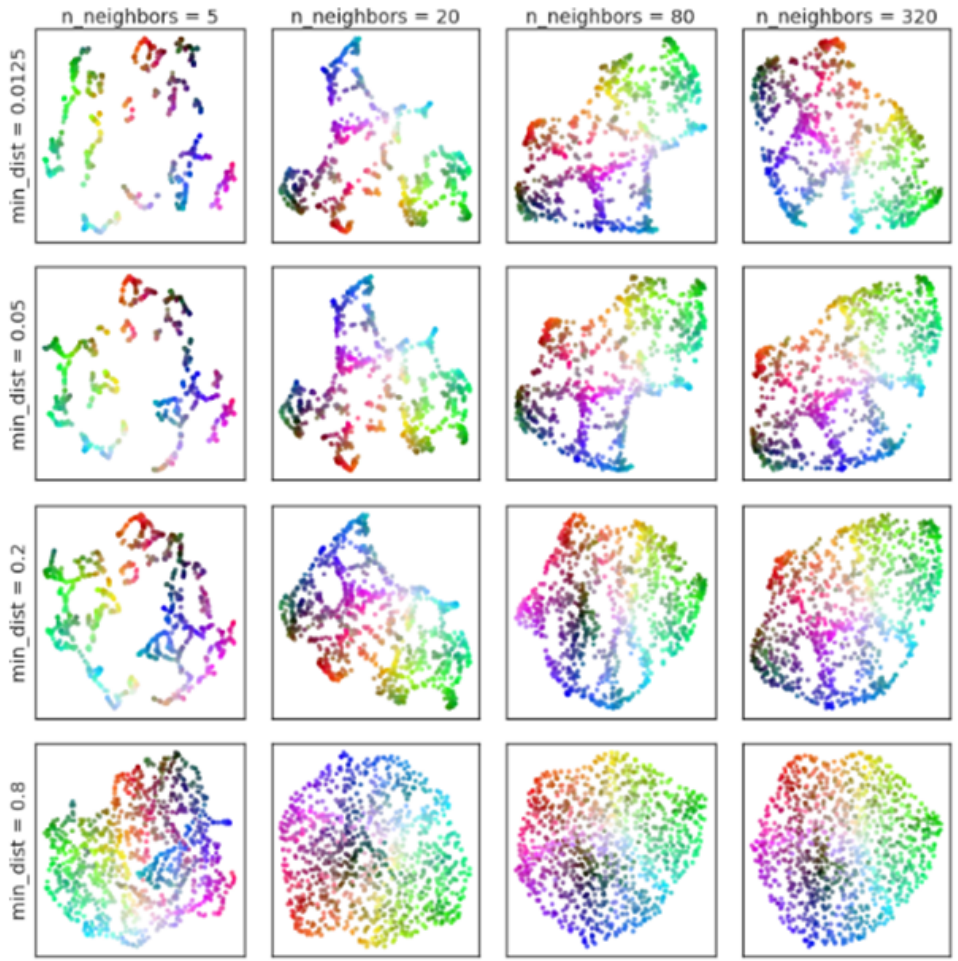
- We need to get to a place where the computer is doing more automated and complex analysis
- Analyst should not be required to have a graduate degree in data analytics



How Do We Get to the Future of Trajectory Analytics

Two broad categories of advances to be made:

- **Representation of trajectory information**
  - Should be concise
  - Should enable real-time analysis in most situations
  - Should be implementable on today's underlying trajectory information systems
- **Advances in applying advanced techniques from the ML literature**
  - Clustering
  - Classification
  - Anomaly detection



Representation of Trajectory Information: Feature Vector

Building up a list of descriptors about something is called a ***feature vector***

$$f_i = (x_{1i}, x_{2i}, \ldots, x_{ni})$$

Once you have a feature vector with *n* features in it, what you have is a set of points in an *n*-dimensional space.

Primary requirement: Nearness in feature space corresponds to "similarity"

Features can be as general (for machine discovery) or specific (user expertise) as you like!

- End-to-end distance traveled
- Total distance traveled
- Ratio of end-to-end distance traveled to total distance traveled
- Total curvature
- Total amount of turning
- Average heading change
- Area covered by flight (convex hull of points)
- Eccentricity of the convex hull
- Perimeter of convex hull
- Centroid of points
- Centroid of convex hull
- Start/Stop point
- Nearest distance to a given point
- "Distance" from a given specified track
- "Distance" from a given specified shape
- Start/Stop time
- Time nearest to a given point
- Average speed
- Range of speeds
- Max altitude
- Fluctuations of altitude/shape of altitude/time curve
- Difference from historical data
- Most common speed/altitude (cruise)
- Place, time and heading where first seen / last seen (might not be start/stop points)
- Other…

Generic Features for Machine Learning

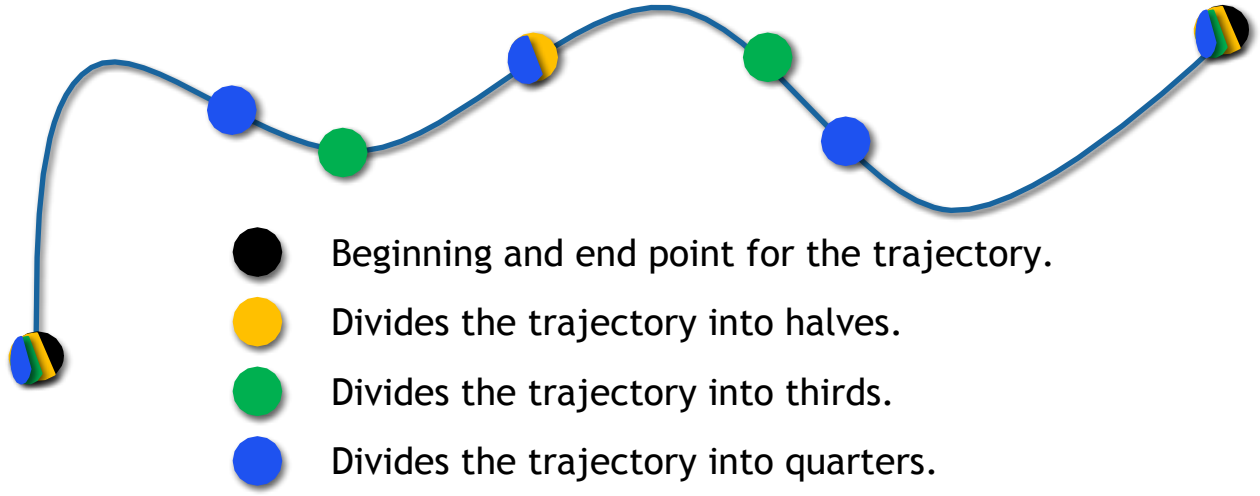
We have the need for a generic, “fingerprint” for flight shapes. This should:

- Be applicable to all flight shapes
- Have a hierarchical definition so that we can use it for high and low-resolution problems
- Be very fast to calculate
- Fit without our “feature vector” approach

Such a thing exists in the math community called, “distance geometry” that had previously been applied to problems in

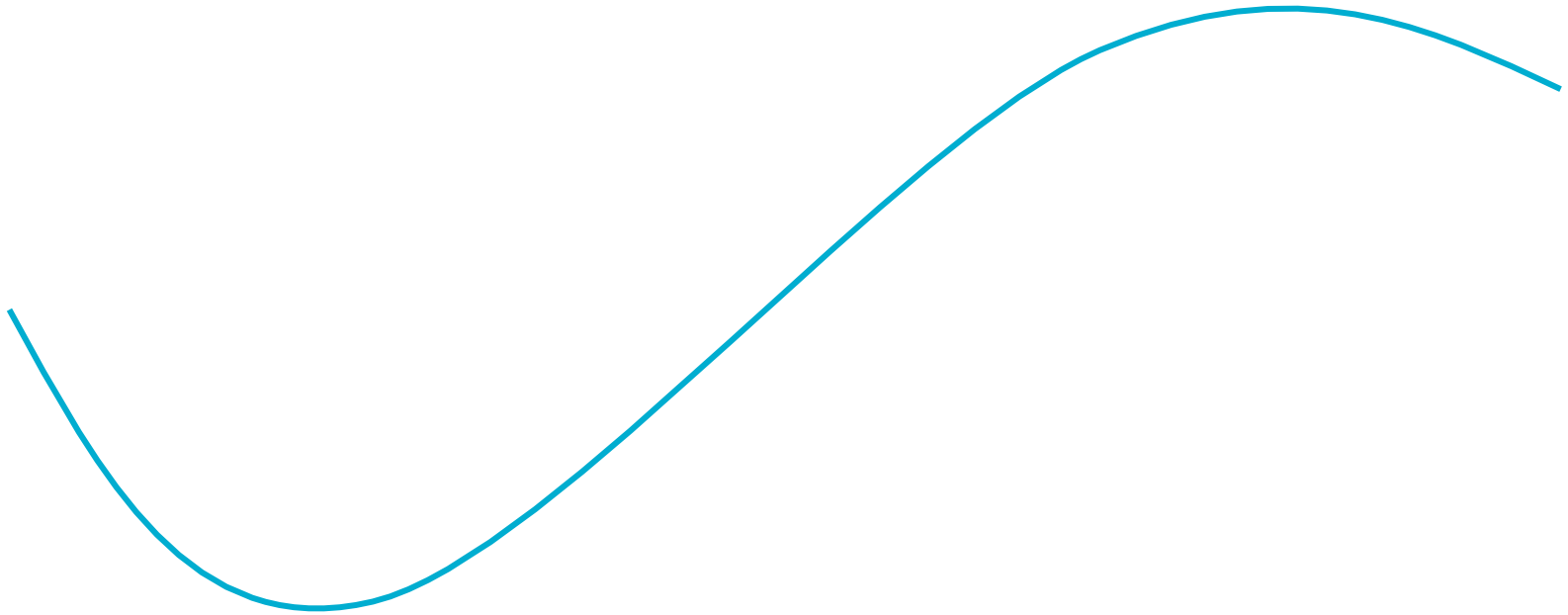
- NMR
- Protein Structure Reconstruction

But it is perfect for trajectory analytics!



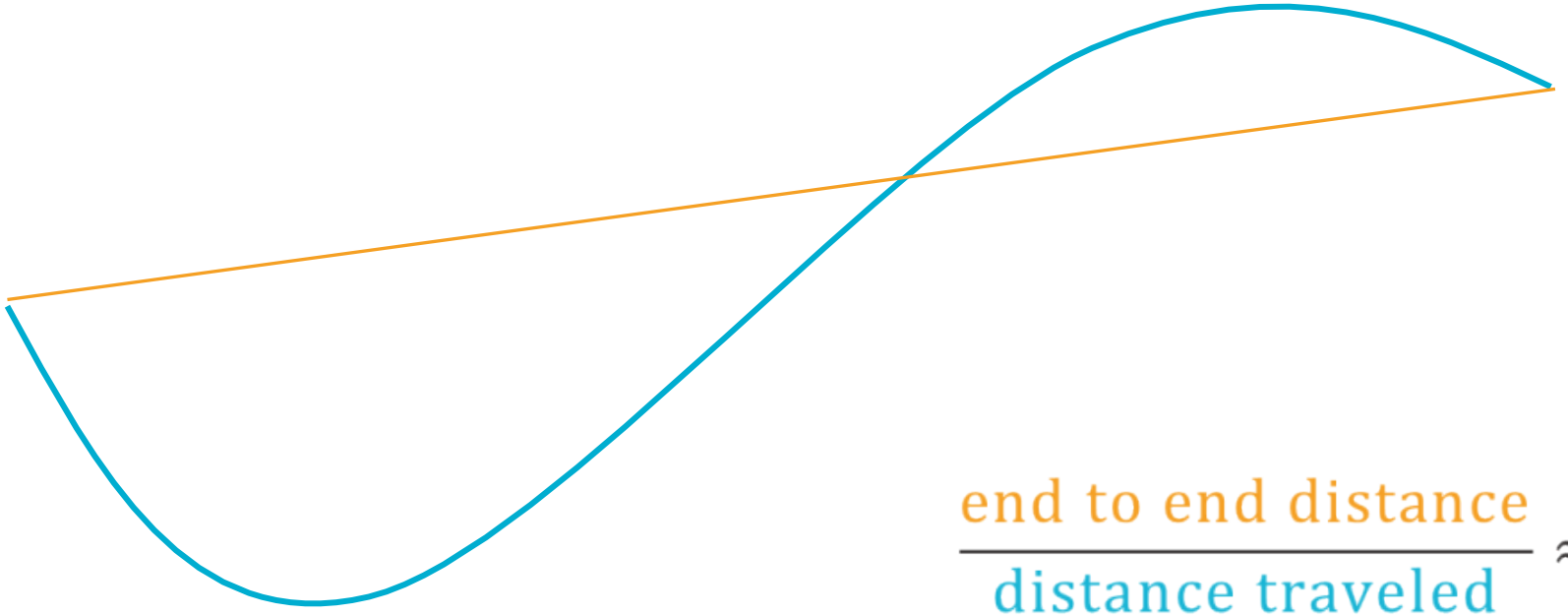
What is Distance Geometry?

Goal: Describe the shape of curves **concisely** and **comparably**.



What is Distance Geometry?

Goal: Describe the shape of curves **concisely** and **comparably**.

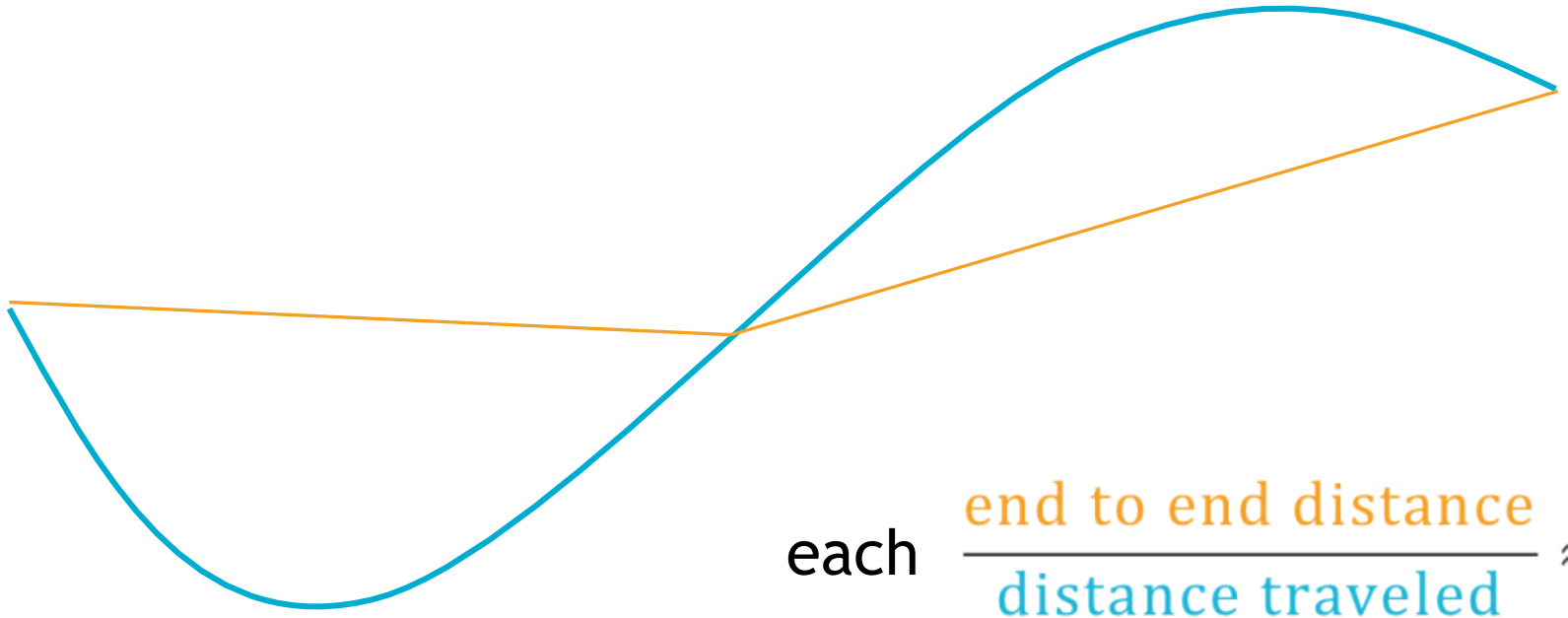


$$\frac{\text{end to end distance}}{\text{distance traveled}} \approx \text{straightness}$$

feature vector = (____, …)

What is Distance Geometry?

Goal: Describe the shape of curves **concisely** and **comparably**.

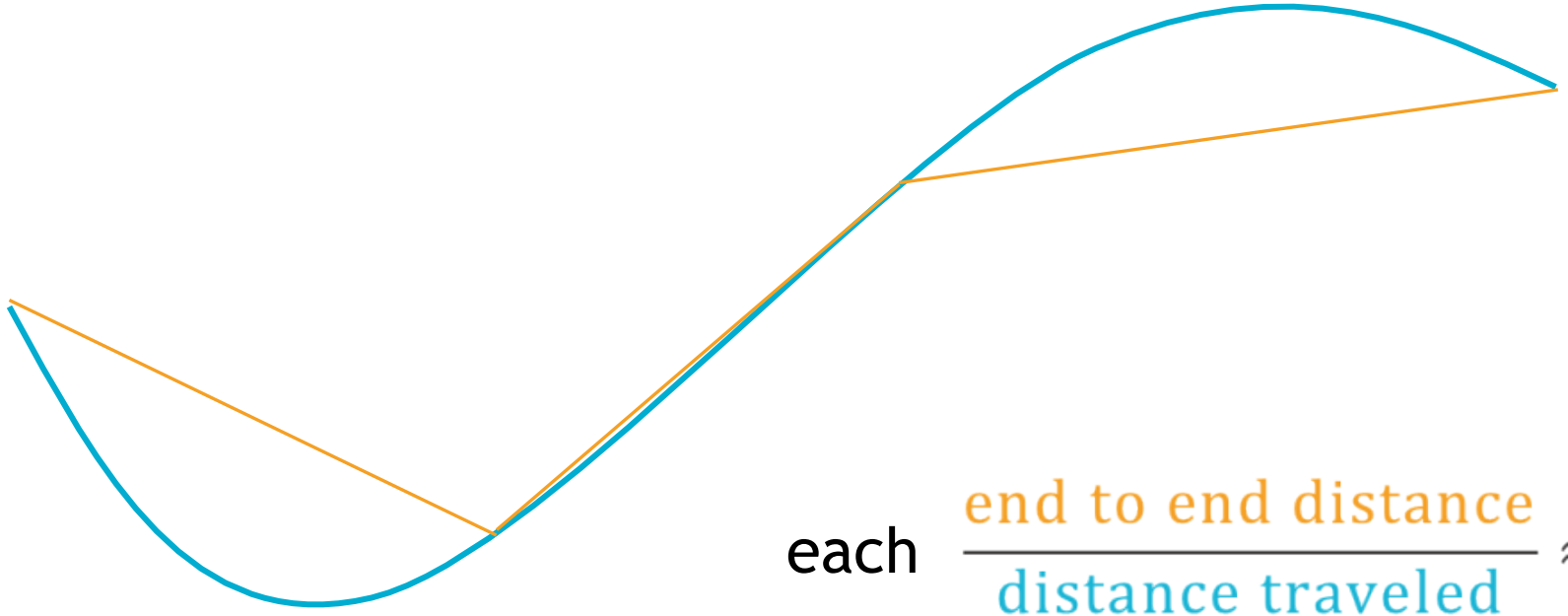


each $\frac{\text{end to end distance}}{\text{distance traveled}} \approx \text{straightness}$

feature vector = (___, ___, ___, …)

What is Distance Geometry?

Goal: Describe the shape of curves **concisely** and **comparably**.

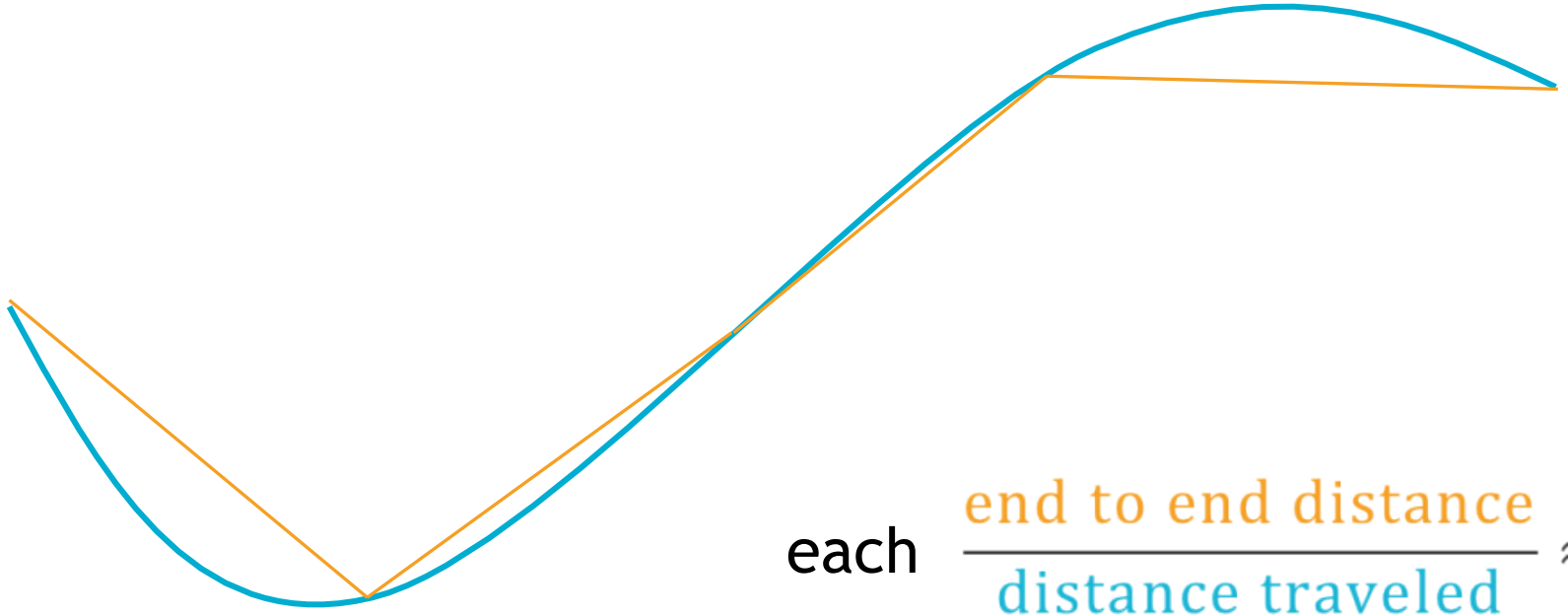


each $\frac{\text{end to end distance}}{\text{distance traveled}} \approx \text{straightness}$

feature vector = (___, ___, ___, ___, ___, ___, …)

What is Distance Geometry?

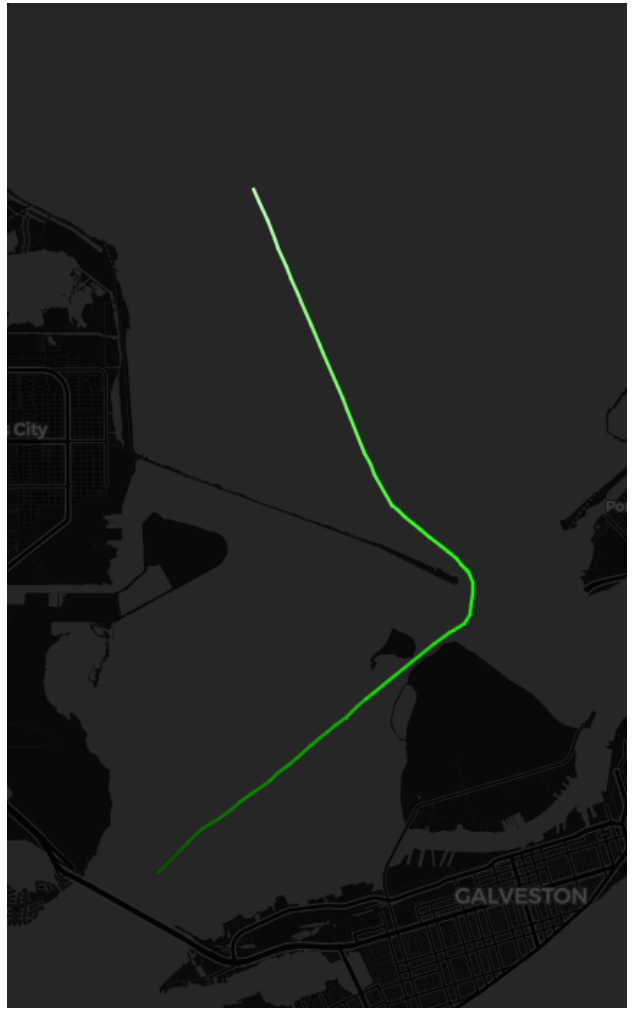
Goal: Describe the shape of curves **concisely** and **comparably**.



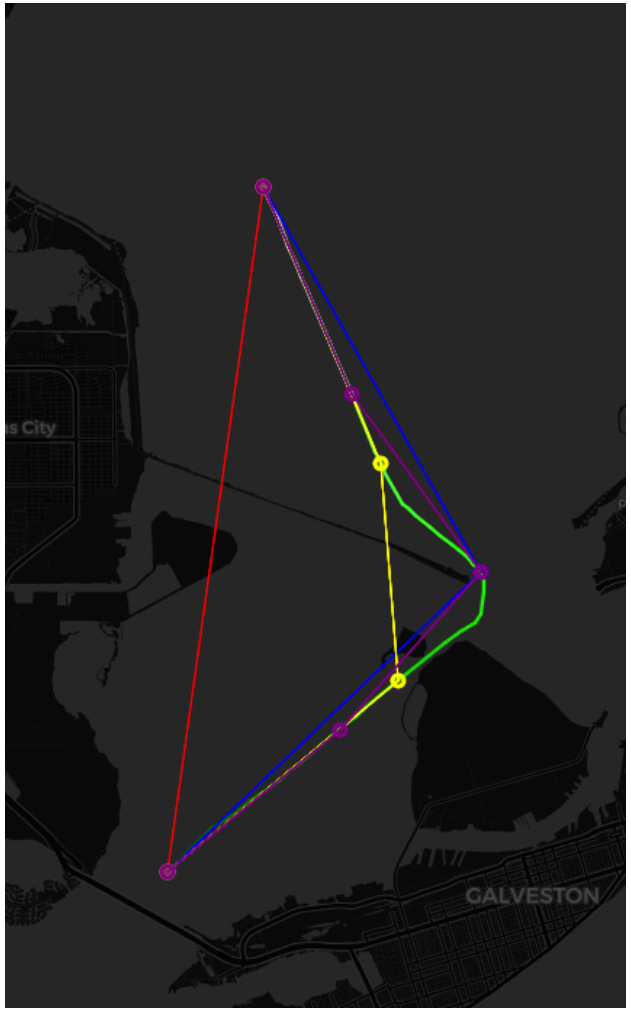
each $\frac{\text{end to end distance}}{\text{distance traveled}} \approx \text{straightness}$

feature vector = (___, ___, ___, ___, ___, ___, ___, ___, ___, ___)

Distance Geometry: Example



original trajectory



trajectory overlaid with distance geometry

```
feature vector
  = [0.76811,
     0.96196,
     0.98189,
     0.99745,
     0.72708,
     0.99993,
     0.99669,
     0.93286,
     0.97614,
     0.99992]
```

Clustering

Feature vectors are very powerful, and can be used for any generic trajectory comparison algorithm.

But their real power comes from unsupervised learning approaches, such as **clustering**.

What we ideally want, is the computer to automatically

- Finding natural neighbors of a flight
- Find unexpected patterns
- Find outliers

Once you have your flights described by feature vectors and have the capability of spatial indexing, these problems can all be solved by clustering.

But what kind of clustering?

Density-based Clustering

DBSCAN

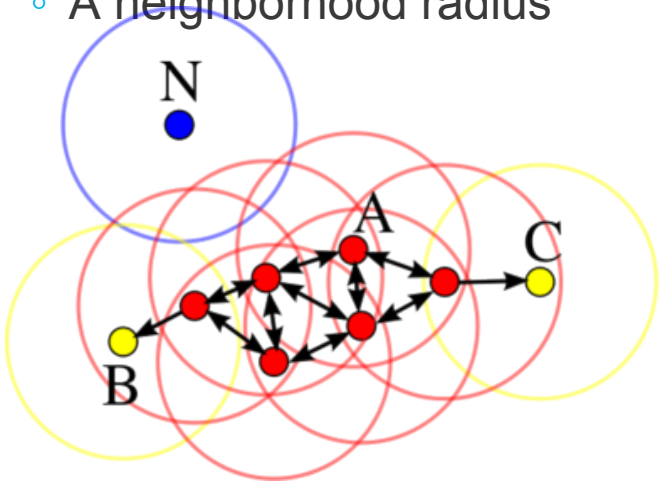
Simple but effective density-based clustering algorithm

DBSCAN has two nice properties

- Doesn't require a pre-defined number of clusters
- Has the notion of noise (outliers)

DBSCAN requires

- A neighborhood density
- A neighborhood radius



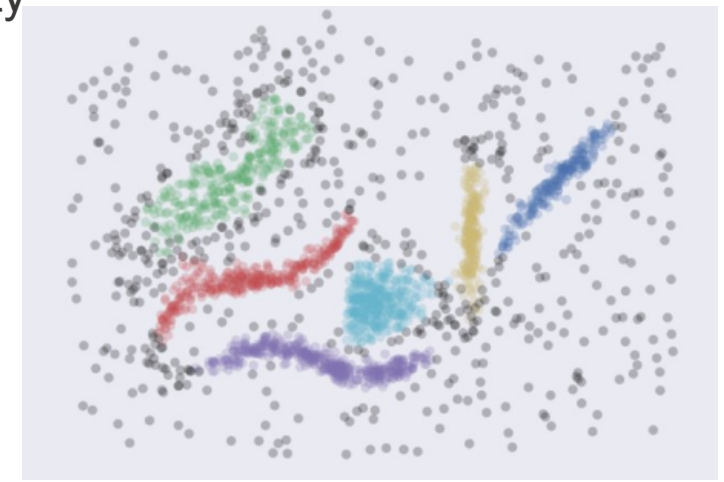
HDBSCAN

More complex than DBSCAN, focused on identifying clusters with **relatively** higher density than the points around them

Doesn't have notion of neighborhood radius (although that can be incorporated)

Essentially has two "soft" parameters

- Minimum cluster size
- Number of neighbors used to measure density

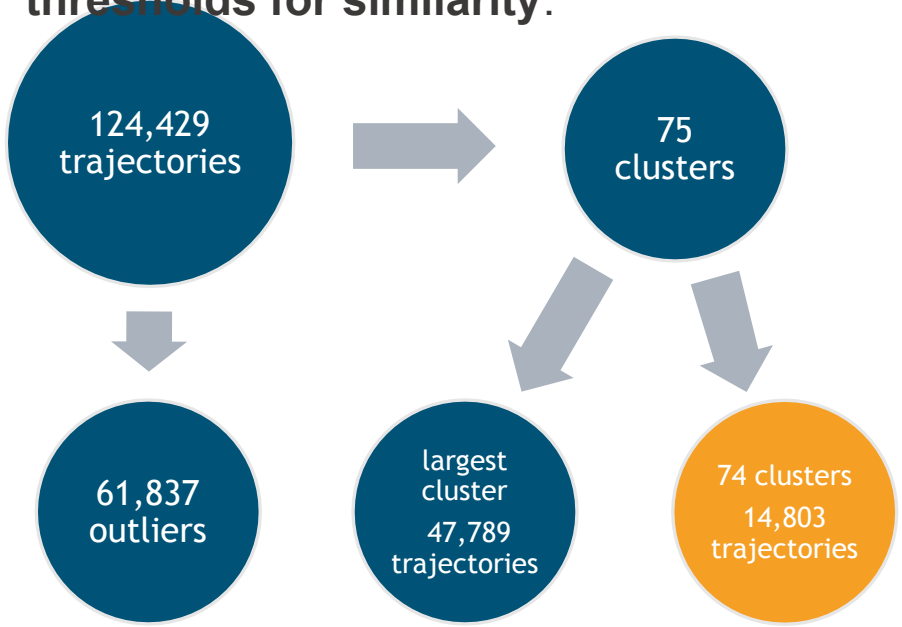


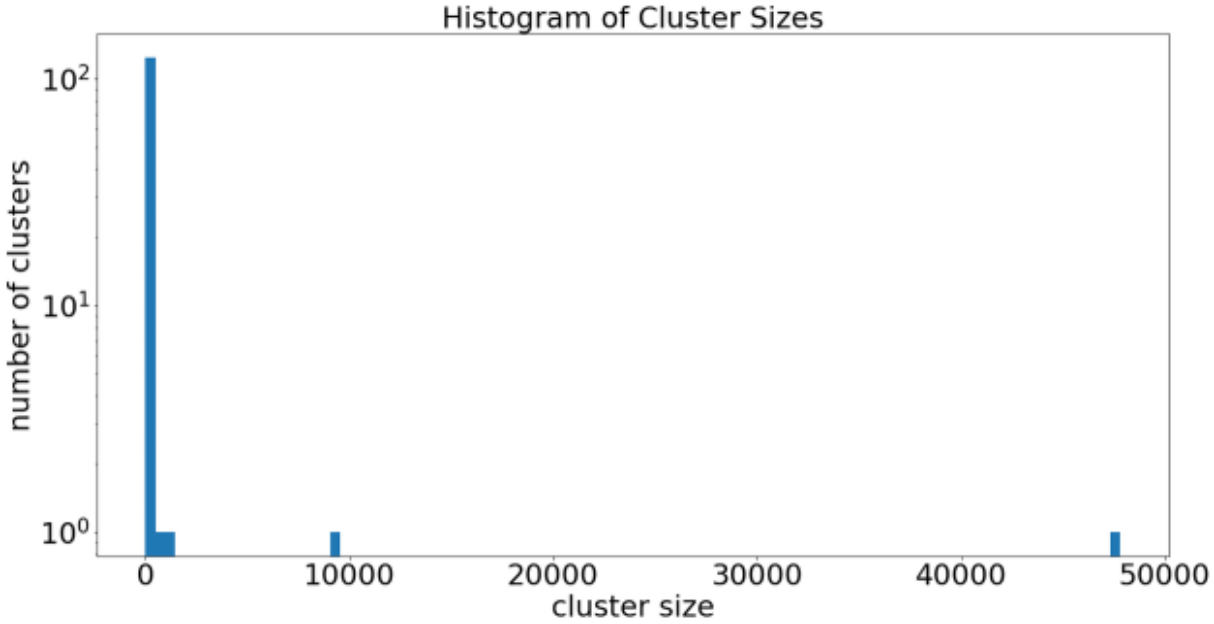
Distance Geometry Clustering Results

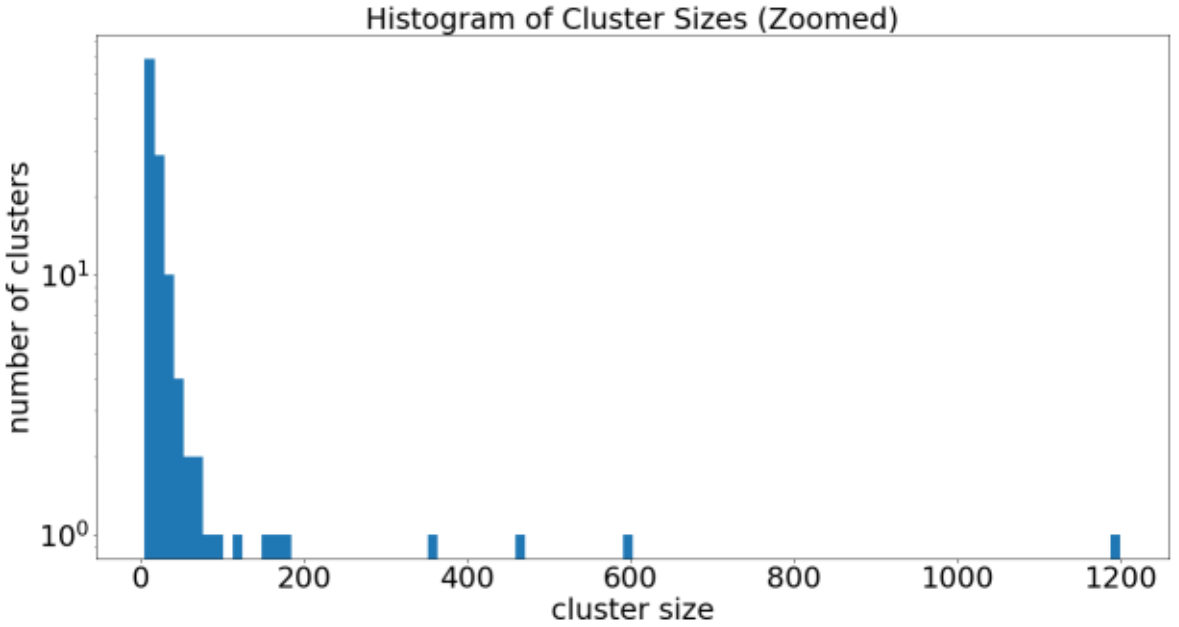
Pattern Detection

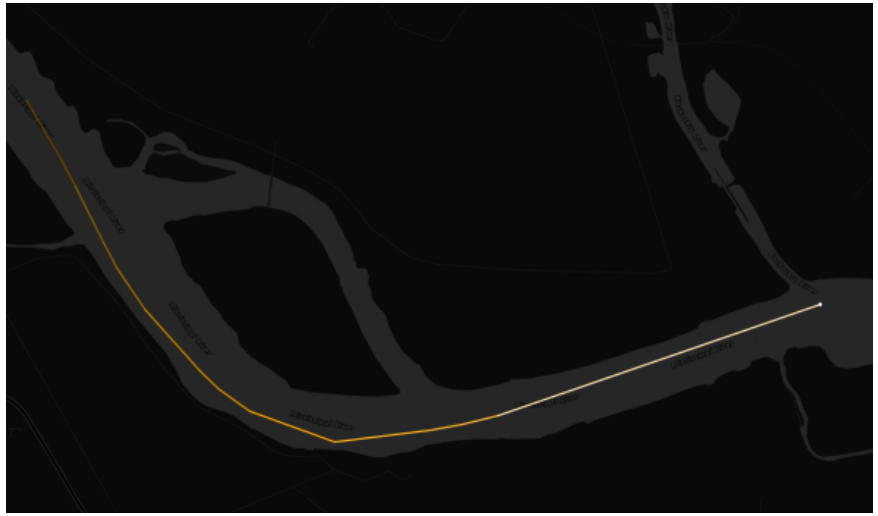
Using Tracktable's **density-based clustering** (box-DBSCAN), we cluster over the feature vectors that we created using distance geometry.

Parameters are chosen to cause trajectories to cluster together based on **high thresholds for similarity**.



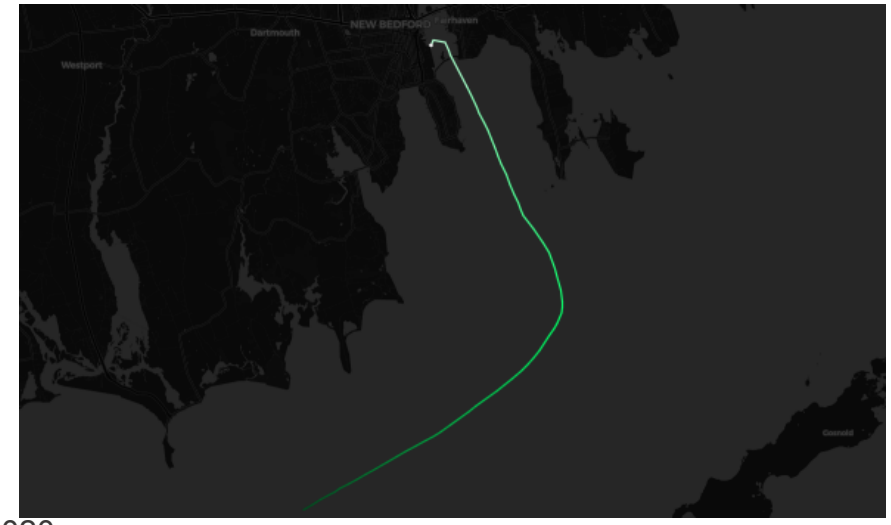


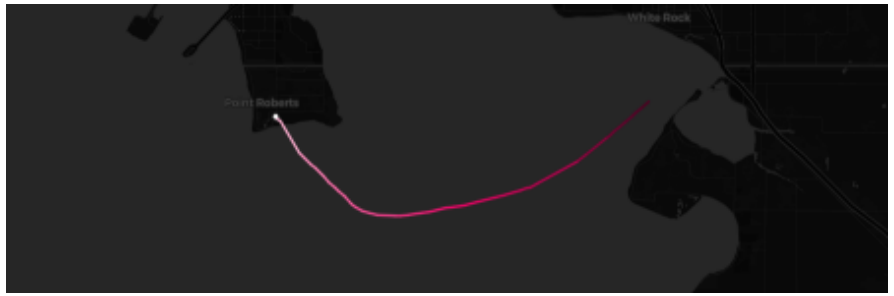


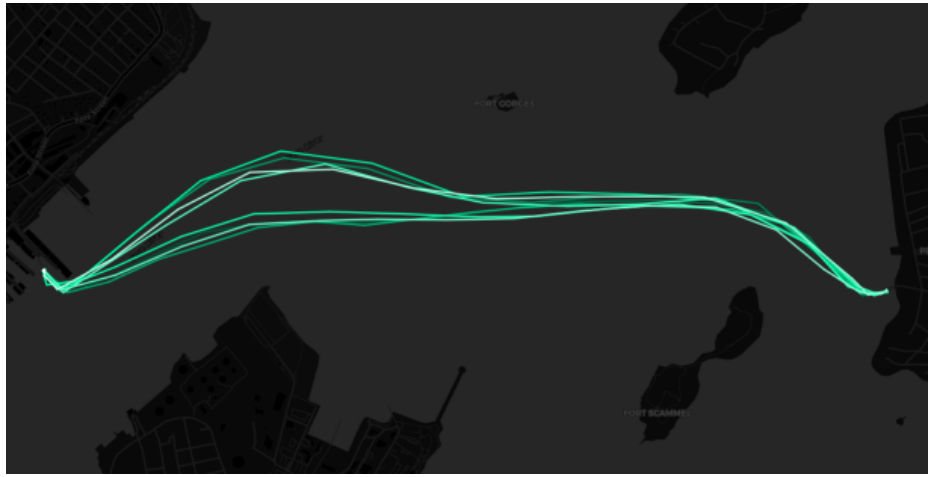


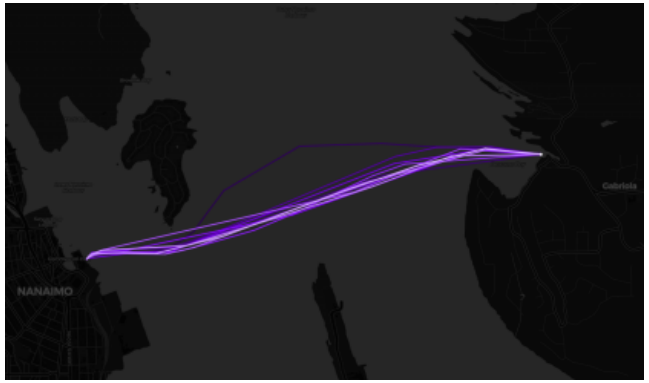
Largest Cluster

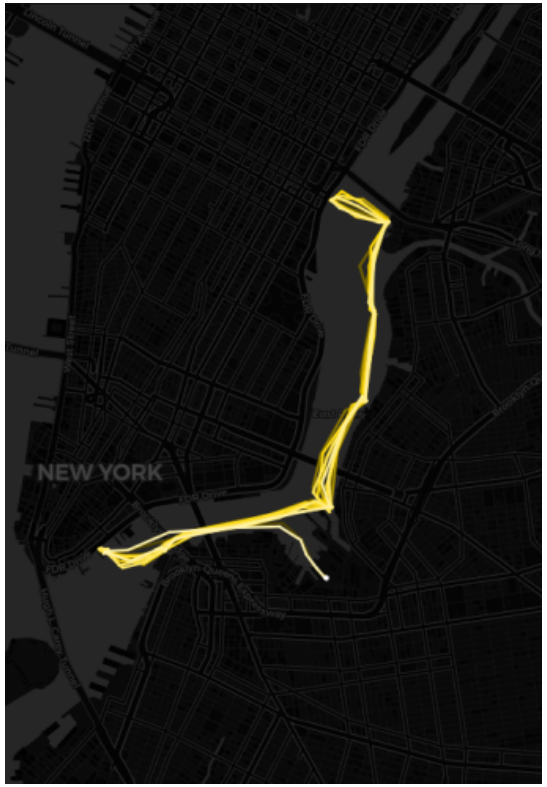
47,789
trajectories





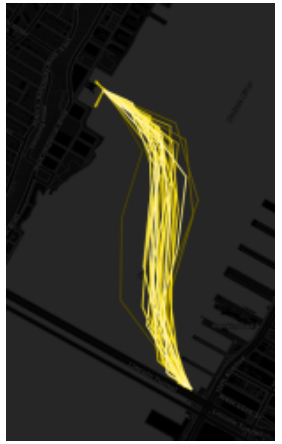


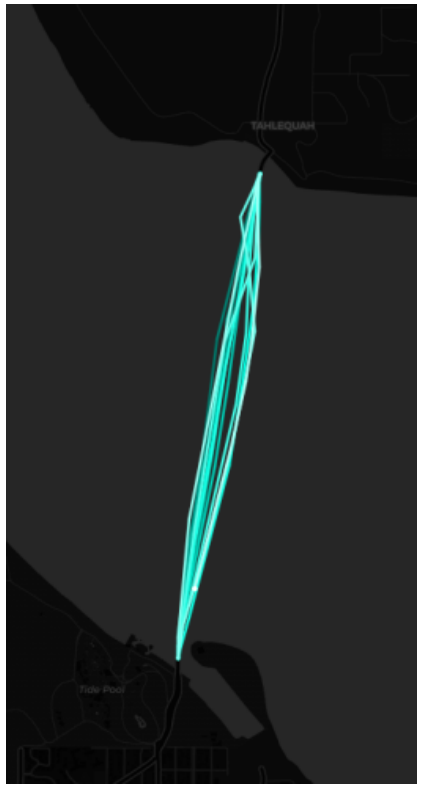


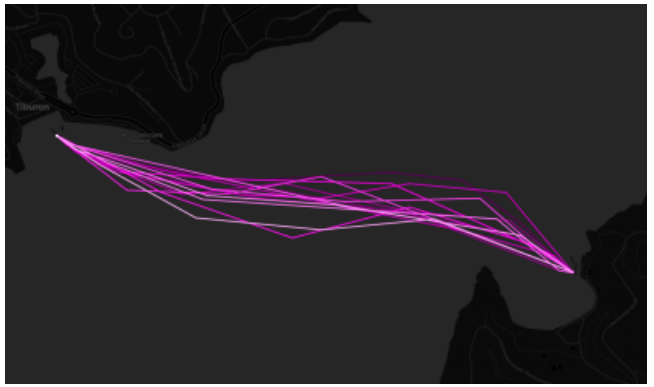


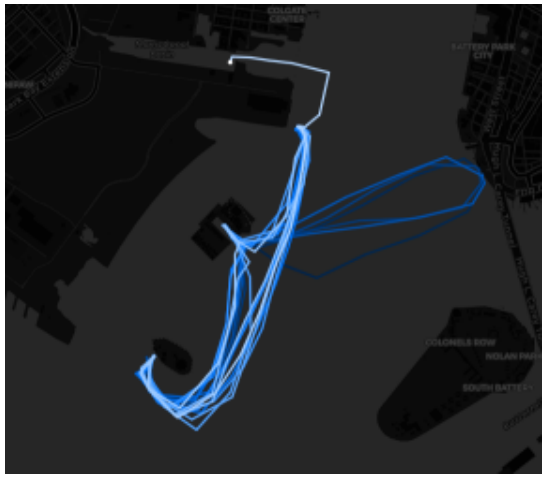
Ferry Cluster

9,234 trajectories

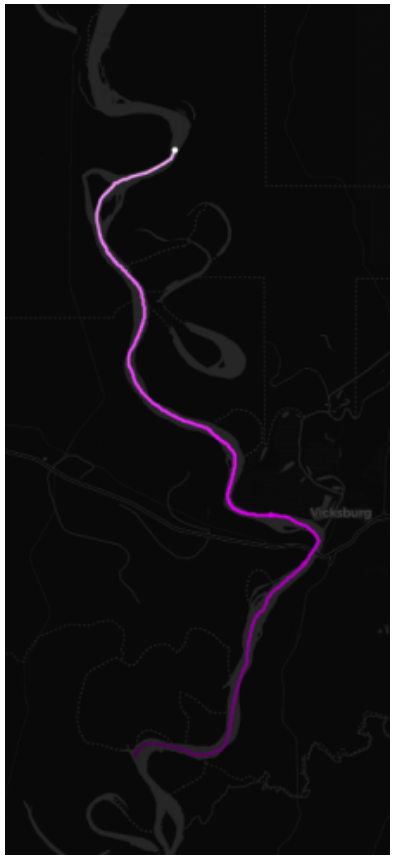








REX DOBSON
April 29-30, 2018



CITY OF VICKSBURG
April 2-3, 2018

MERRICK JONES
April 4, 2018

VIKING QUEEN
April 10-11, 2018

SUSAN L STALL
April 27-28, 2018

Mississippi River Cluster

Five different tug boats traversing the Mississippi River near Vicksburg, MS.

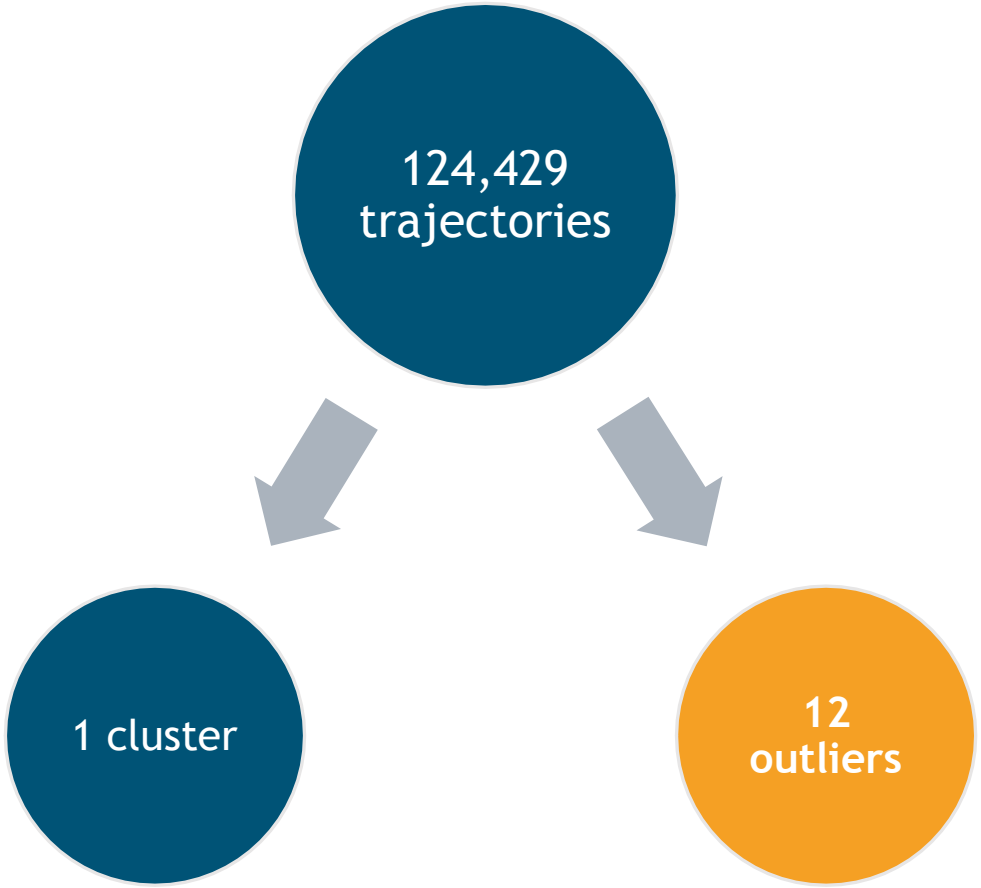
Distance Geometry Clustering Results

Anomaly Detection

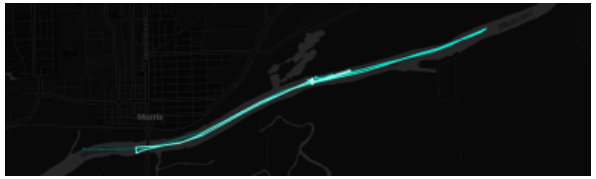
Using Tracktable's **density-based clustering** (box-DBSCAN), we cluster over the feature vectors that we created using distance geometry.

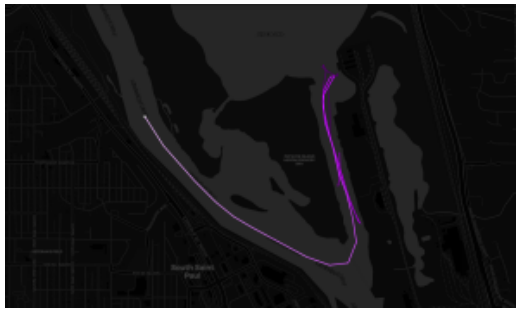
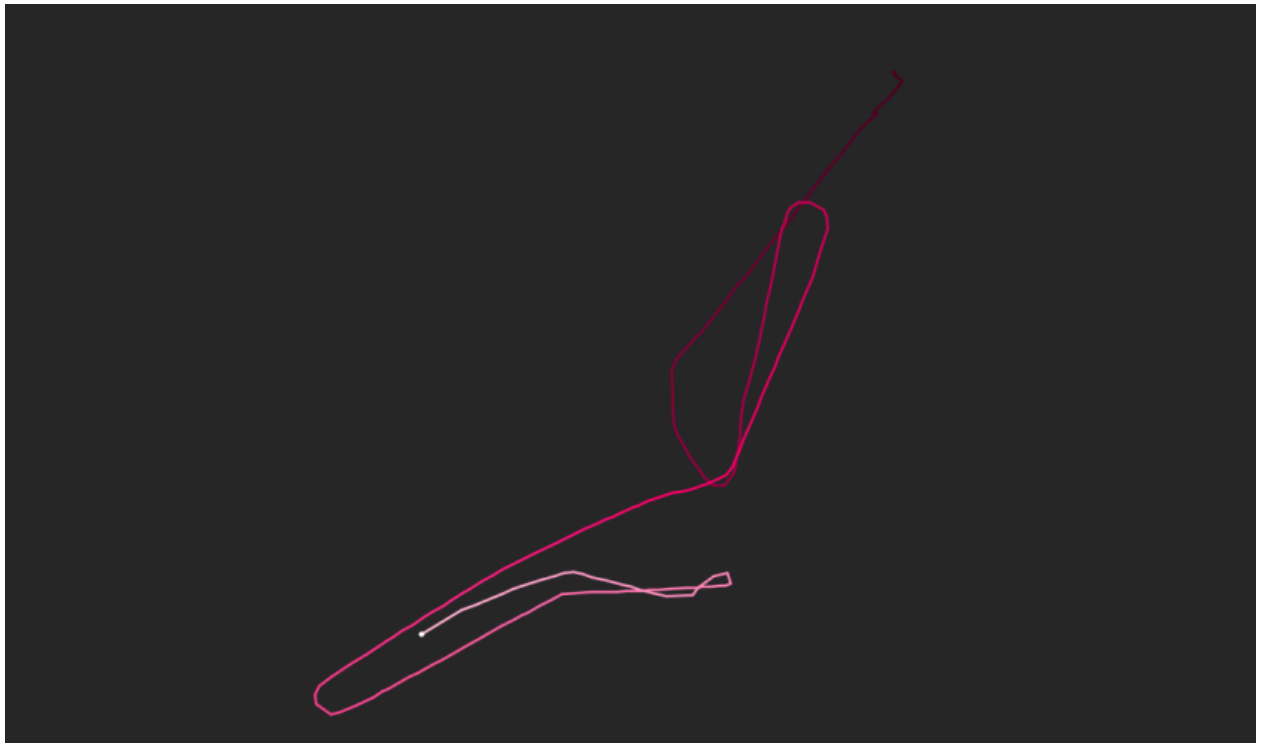
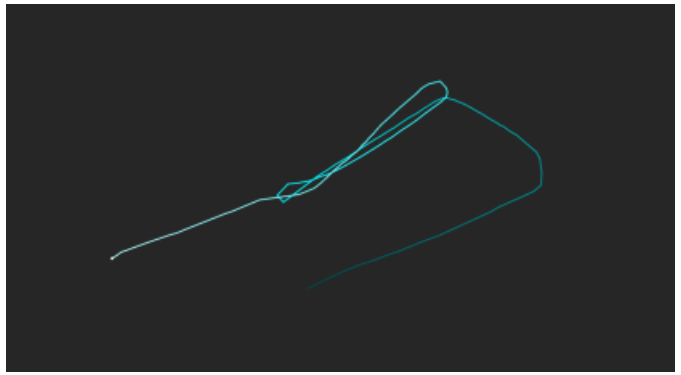
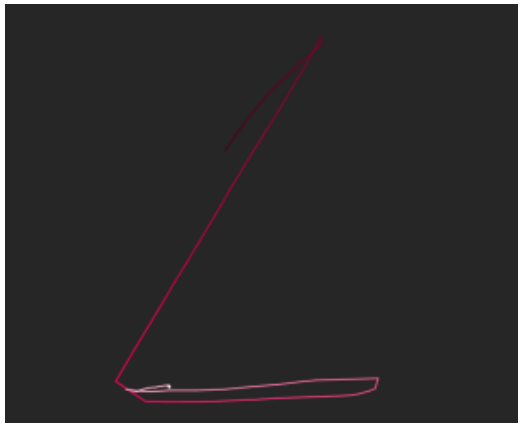
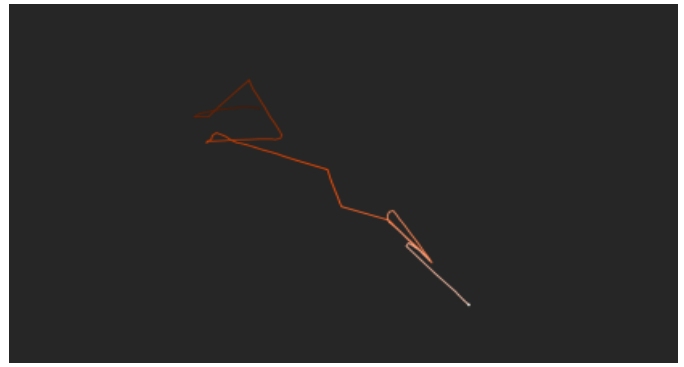
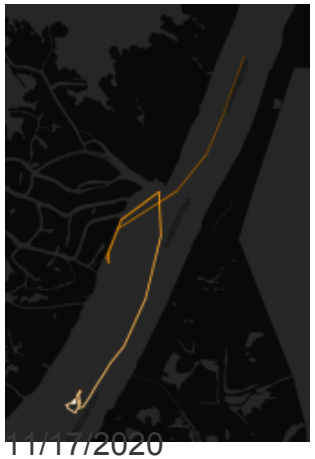
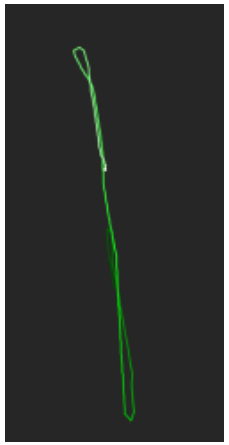
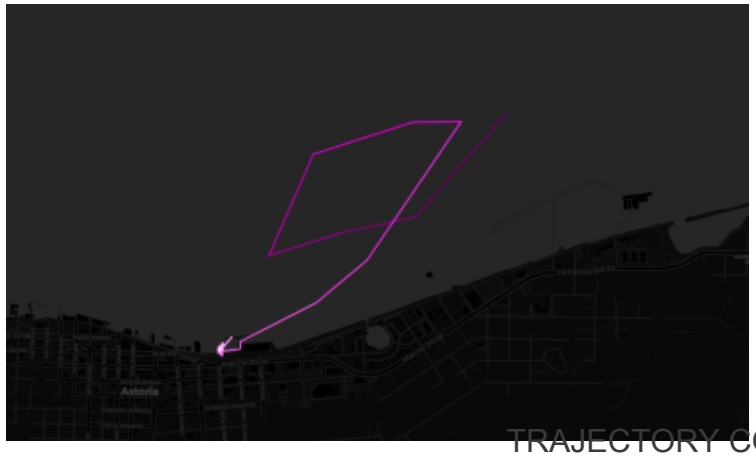
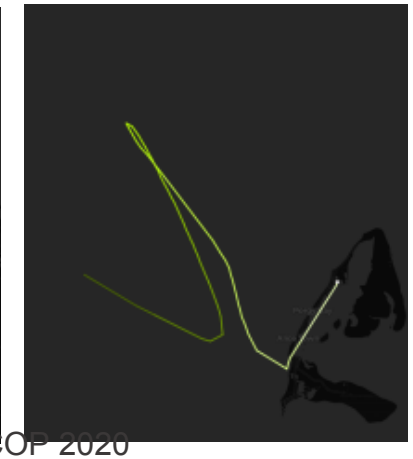
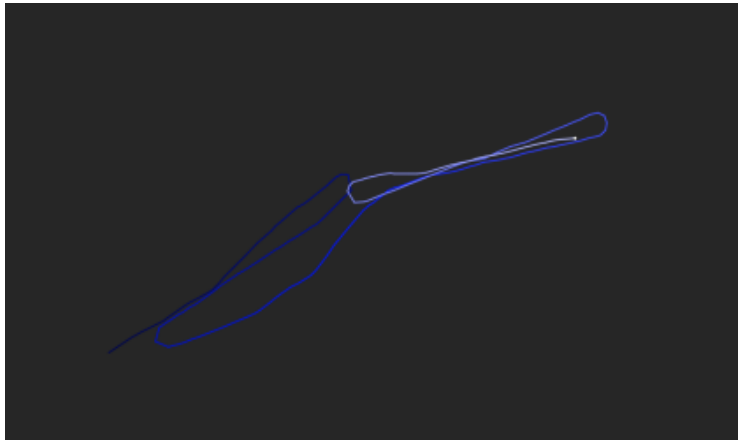
Parameters are chosen to cause trajectories to cluster together based on **low thresholds for similarity**.

Any trajectory that does not cluster is an **extreme outlier**.

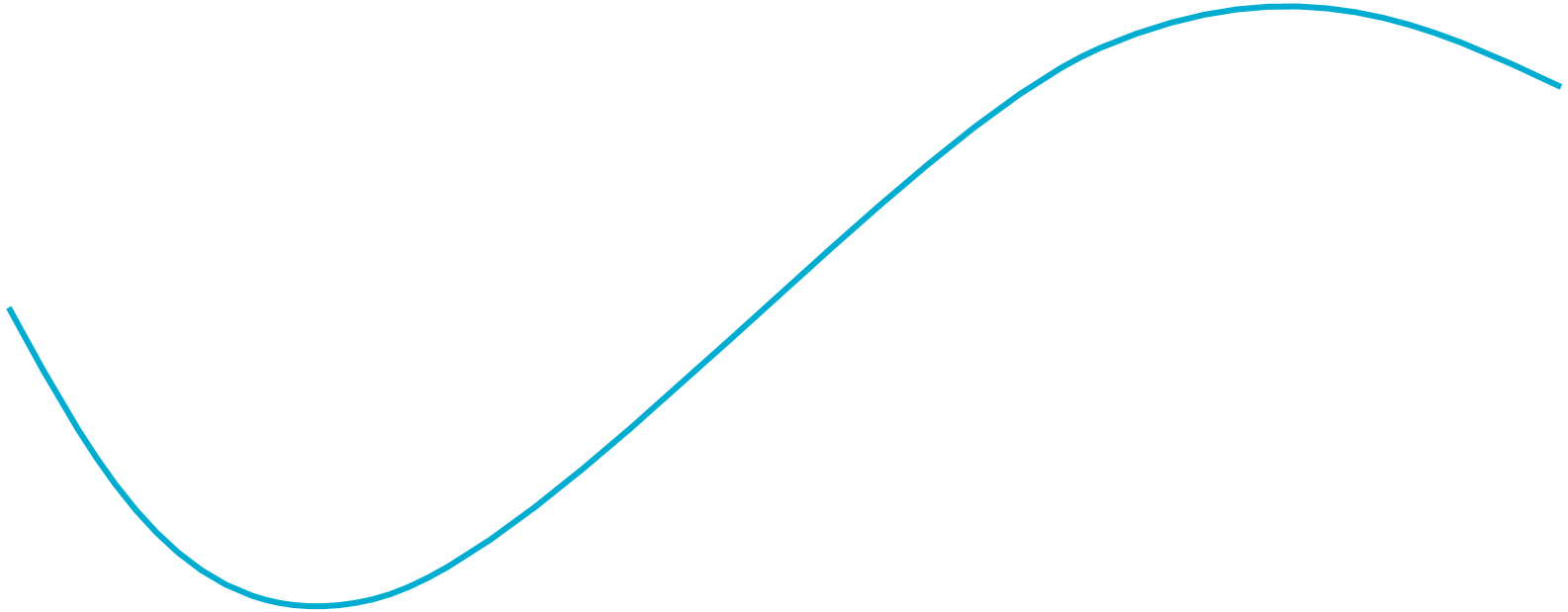


12 Outlier Trajectories



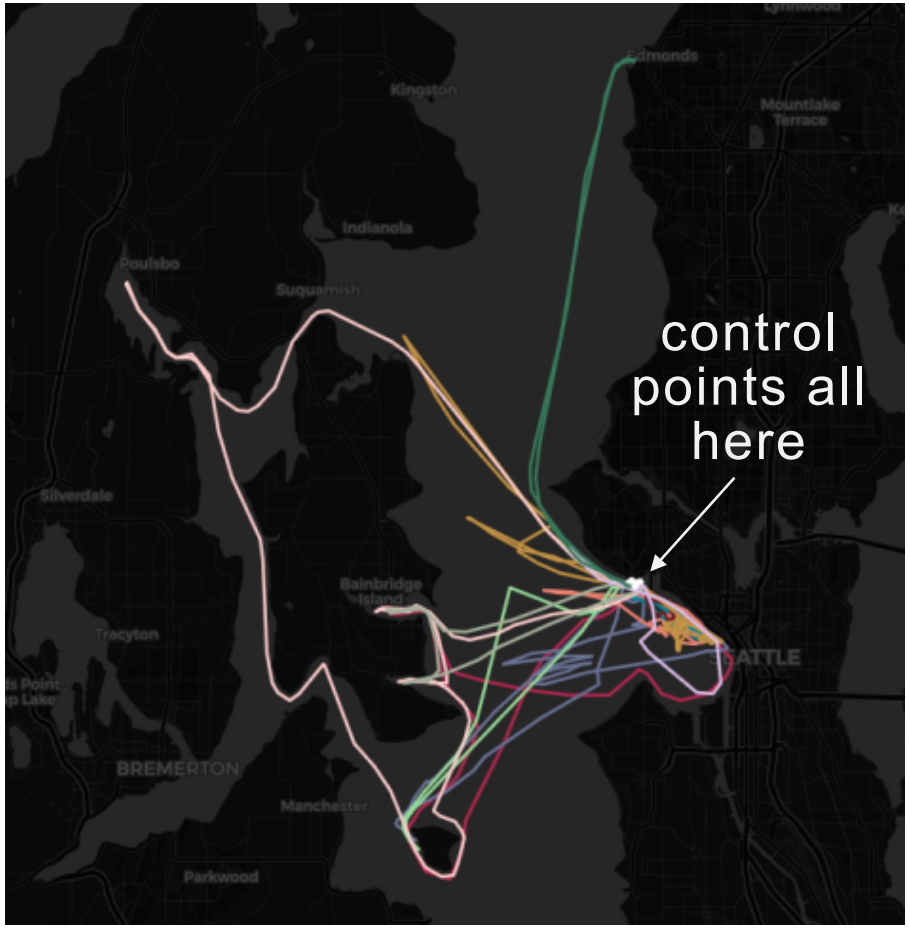
Co-Travel Geometry

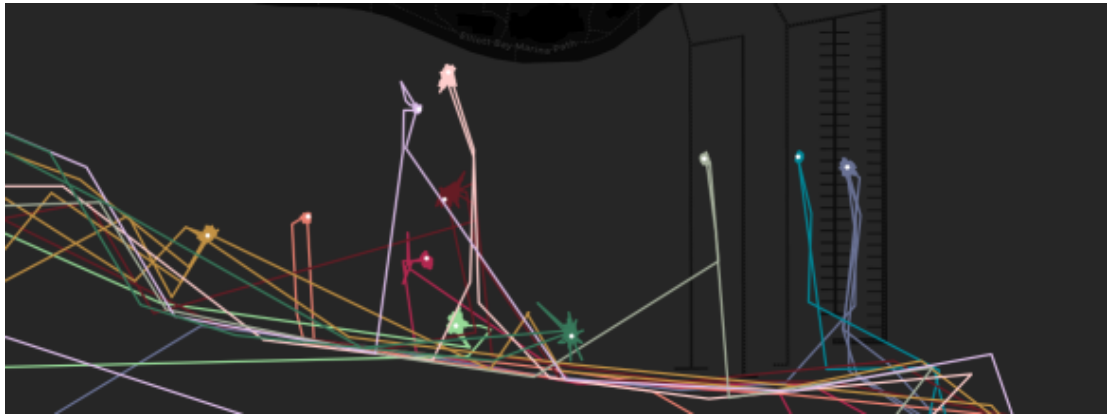


Using the latitude, longitude and timestamp at 10 points that are equally-spaced in TIME along the curve, we get a **30-dimensional temporospatial feature vector**.

Co-Traveler Clustering Results

Using the 30-dimensional temporospatial feature vector and clustering with box-DBSCAN, we initially get poor results.

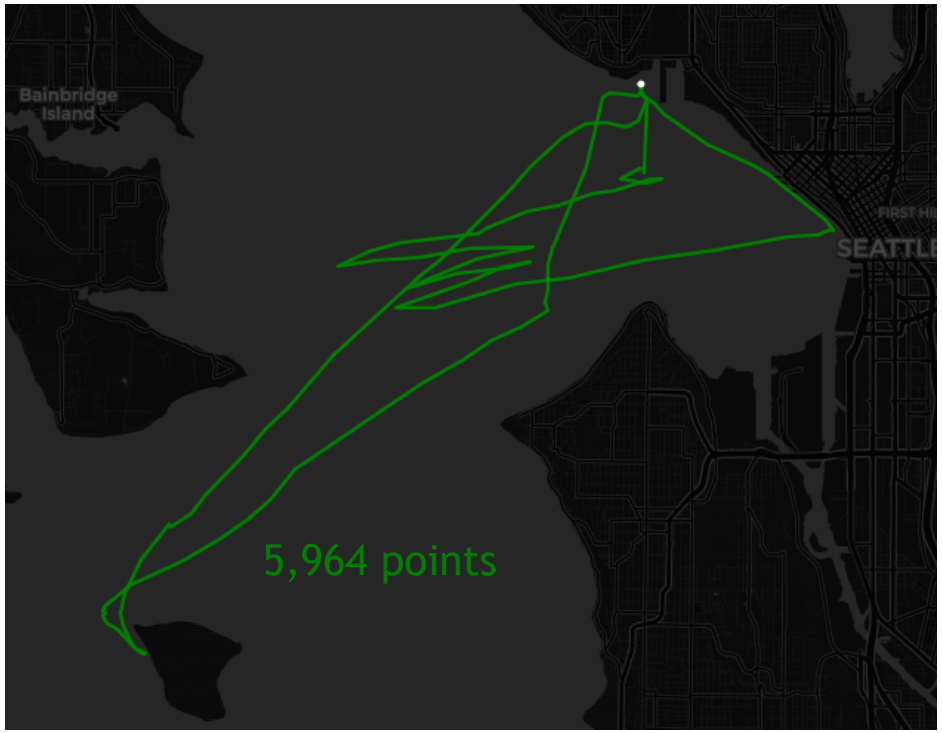




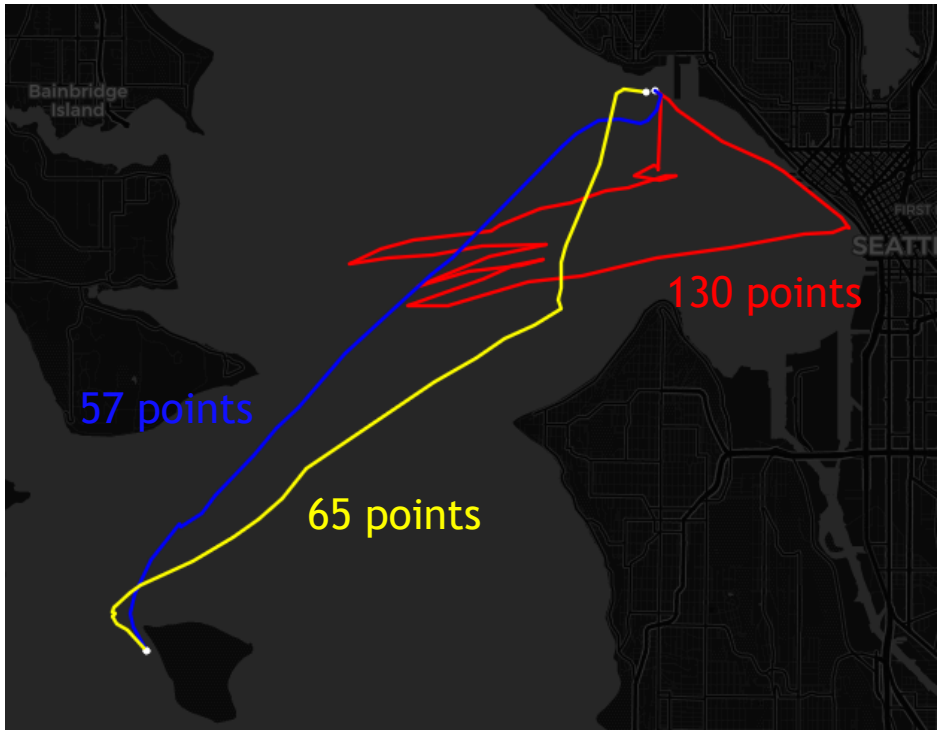
The 11 trajectories making up the largest co-traveler cluster are sitting in port in Seattle together for roughly 11 days before setting out for completely different outings, each lasting less than one day.

Solution: Idle Track Splitting

If the trajectory is “idle” (defined by a given radius and time threshold), split the trajectory and delete the idle points.



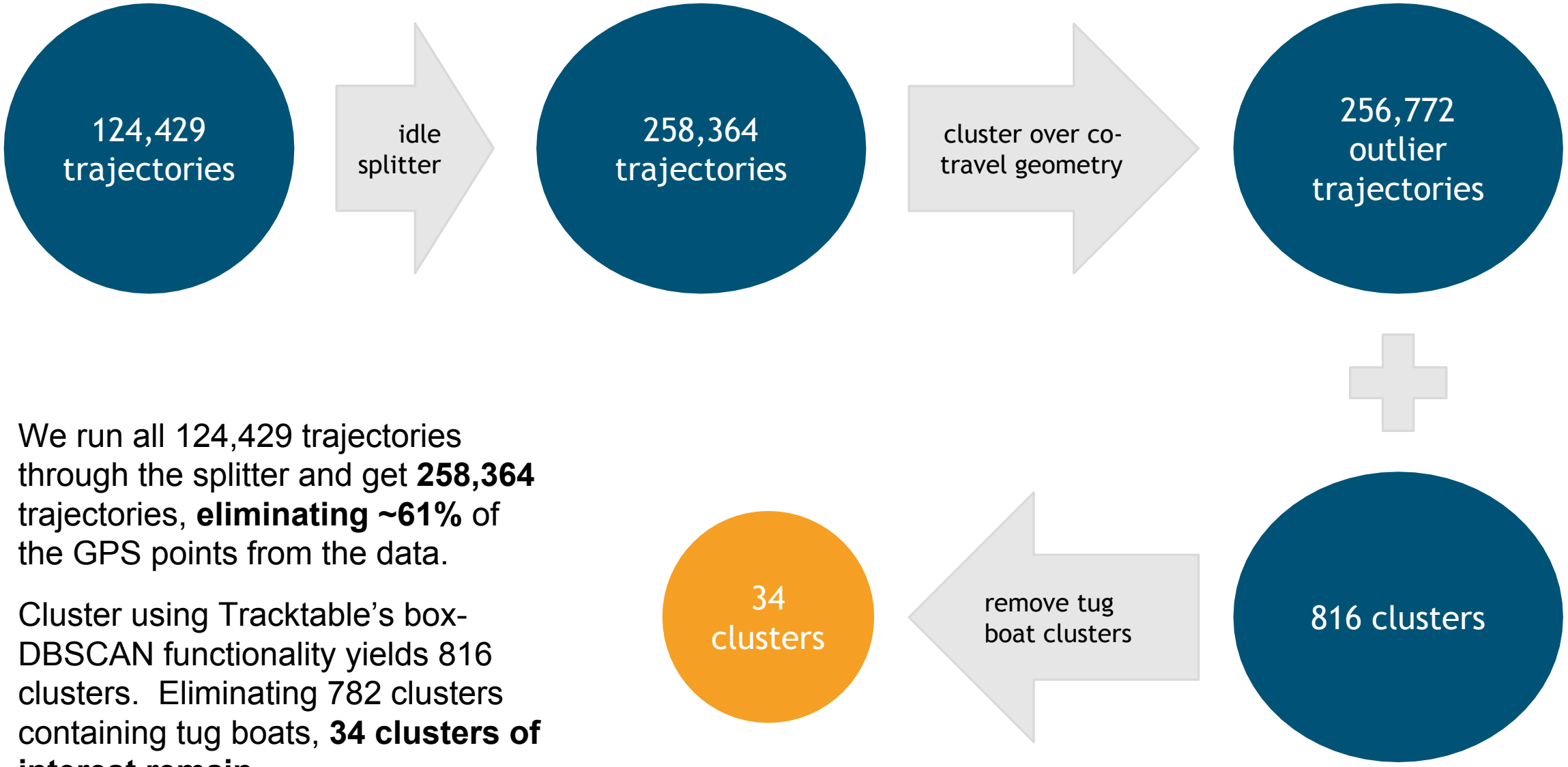
Original Trajectory



New Trajectories

5,712 idle points were deleted, and the remaining segments were formed into new trajectories.

Solution: Idle Track Splitting



We run all 124,429 trajectories through the splitter and get **258,364** trajectories, **eliminating ~61%** of the GPS points from the data.

Cluster using Tracktable's box-DBSCAN functionality yields 816 clusters. Eliminating 782 clusters containing tug boats, **34 clusters of interest remain**.

Co-Traveler Clustering Results (Post-Splitting)

FRANCES T.CARINHAS

VASCO DE GAMA

MARIA C

SEA WASP

LAUREN A

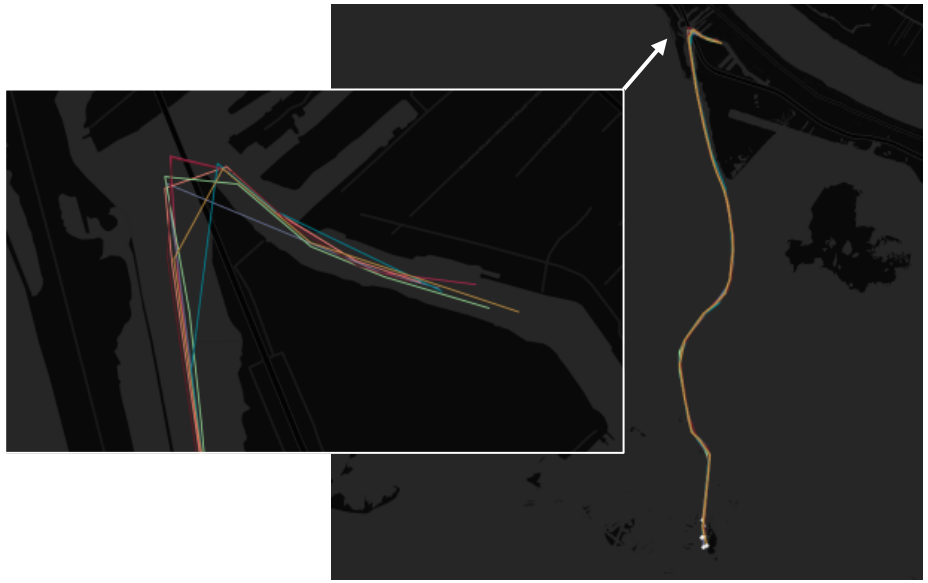
SEA FALCON

Cluster Size	Number of Clusters
2	27
3	5
4	1
7	1
outliers	256,772

Largest Co-Traveler Cluster:

- traveling on April 23, 2018
- **seven vessels**
- depart from a peninsula southeast of New Orleans, LA between **12:38 and 1:05 AM**
- arrive at an island due south from their departure point between **1:35 and 2:02 AM**

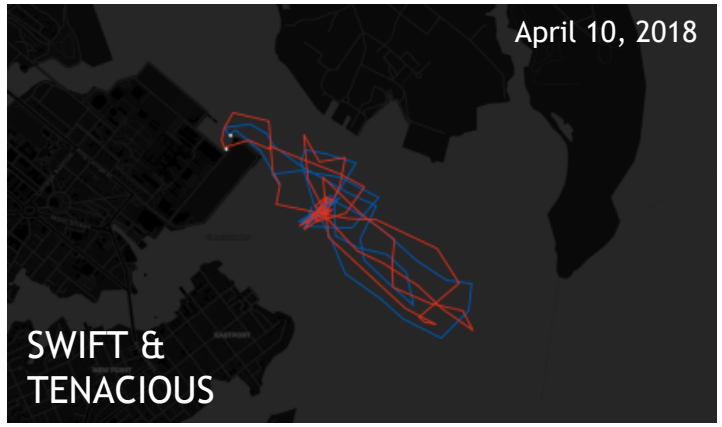
Largest Co-Traveler Cluster

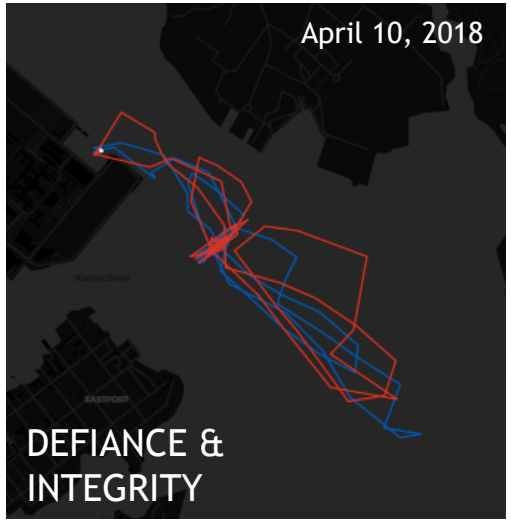


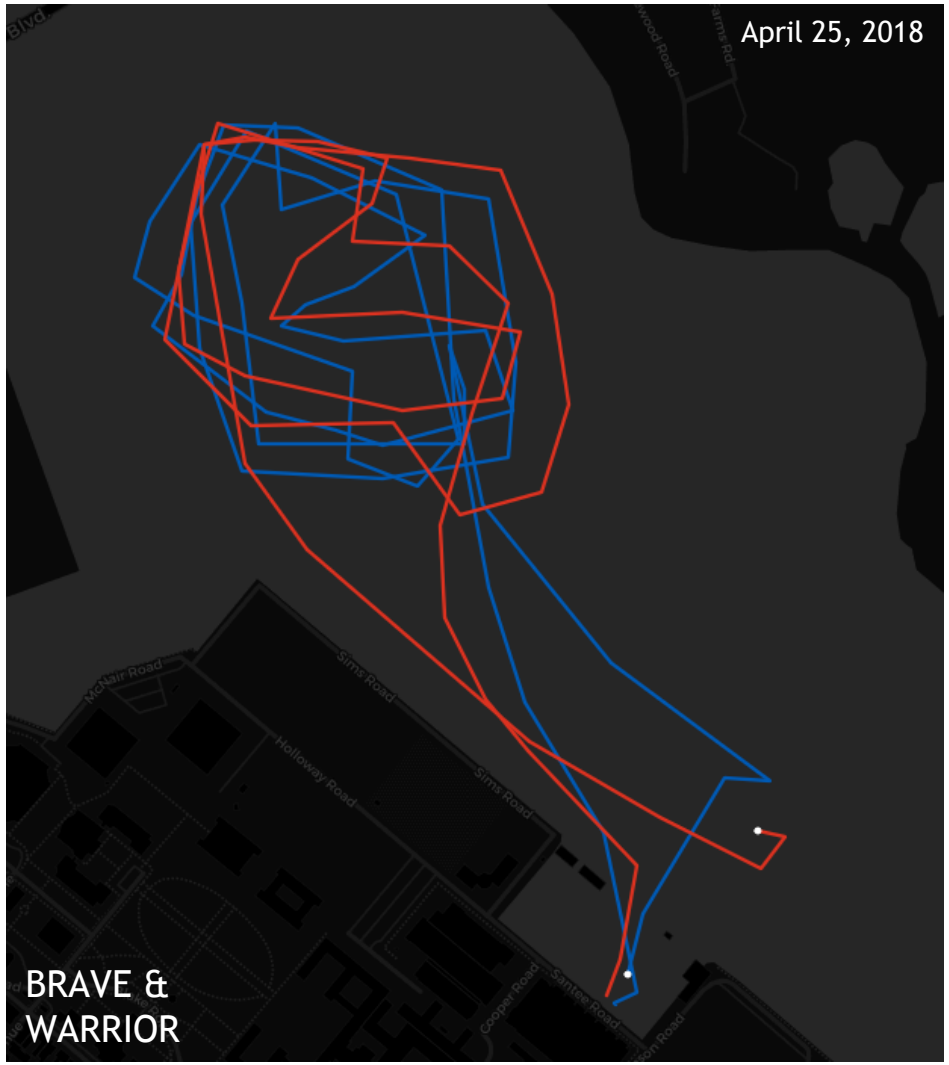
All seven vessels (SEA CHARGER photo not available) belong to Oceana Group (oceana.co.za), a **South African fishing company** that operates in the Gulf of Mexico via a subsidiary, Daybrook Fisheries. Daybrook Fisheries is the leading U.S. producer of sustainable Gulf Menhaden Fishmeal and Fish Oil.

DAYBROOK
AN OCEANA GROUP COMPANY

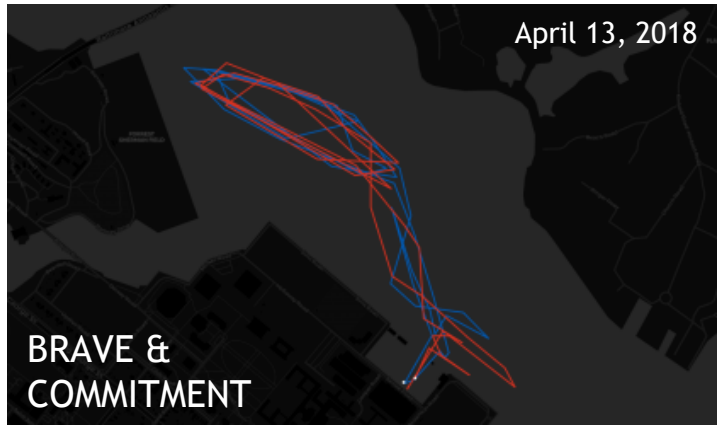
Co-Traveler Clustering Results (Post-Splitting)

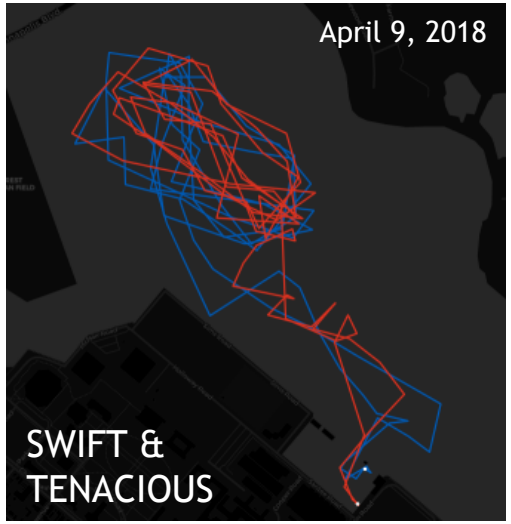






WARRIOR SAILING

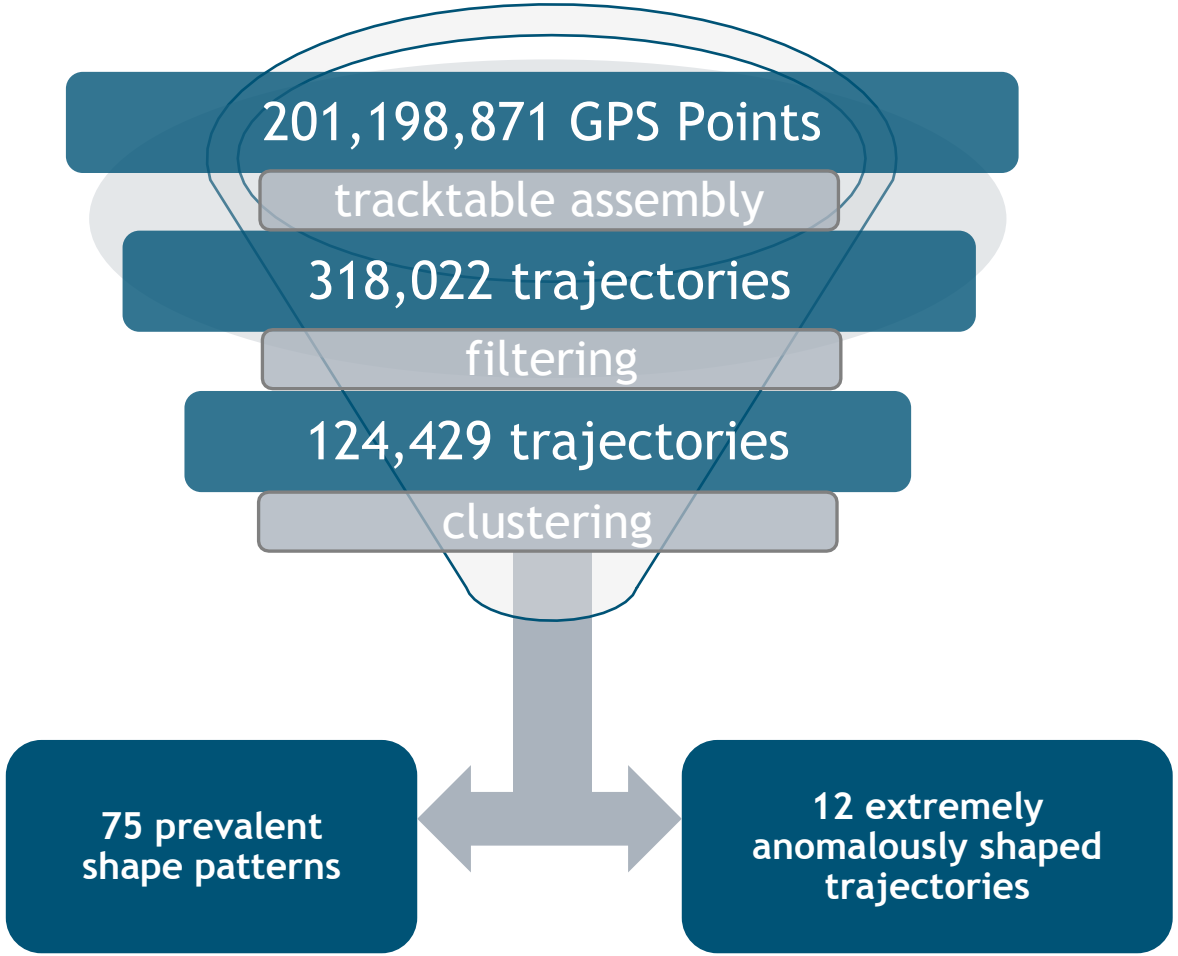




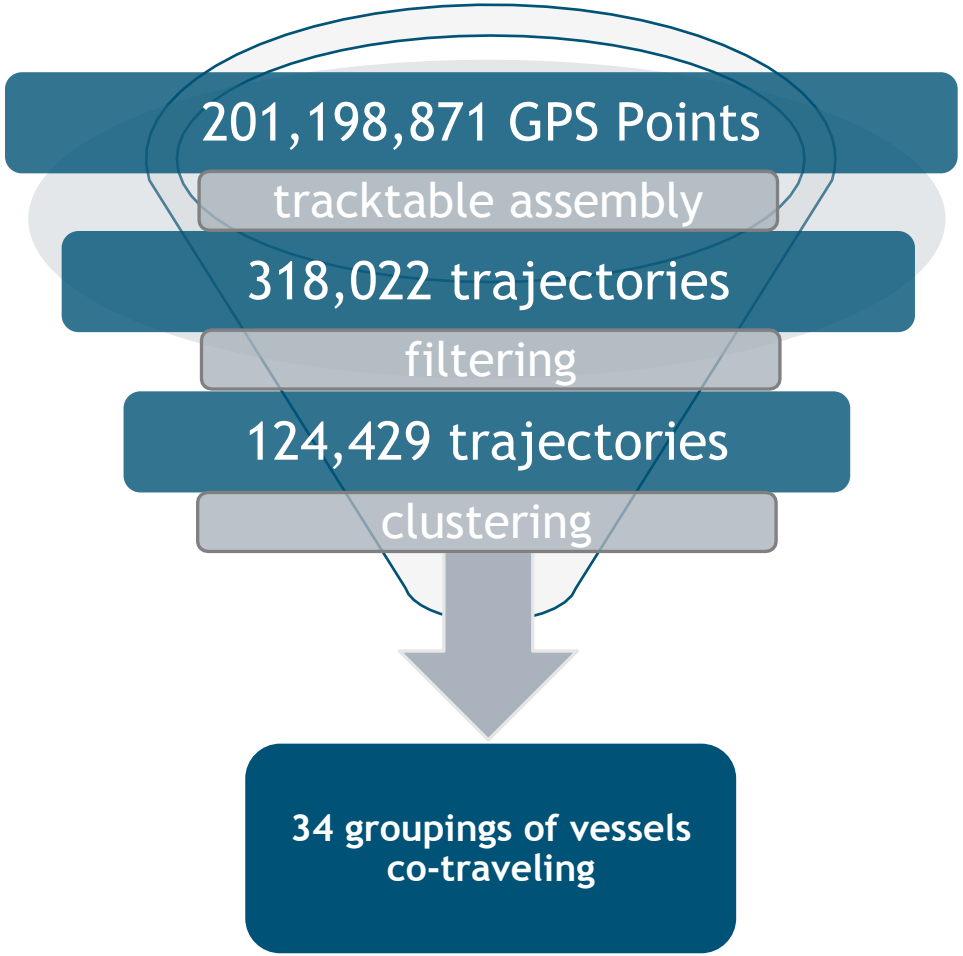
Five co-traveler pairs are sail boats operating in Annapolis, MD as part of Warrior Sailing. According to their website, “Warrior Sailing provides maritime education and outreach for wounded, ill and injured service members and veterans.”

Clustering Results Overview

Box-DBSCAN w/ **Distance** Geometry

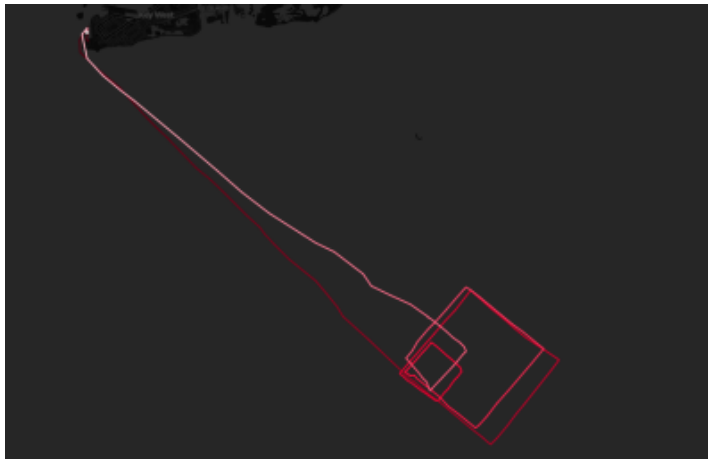


Box-DBSCAN w/ **Co-Travel** Geometry

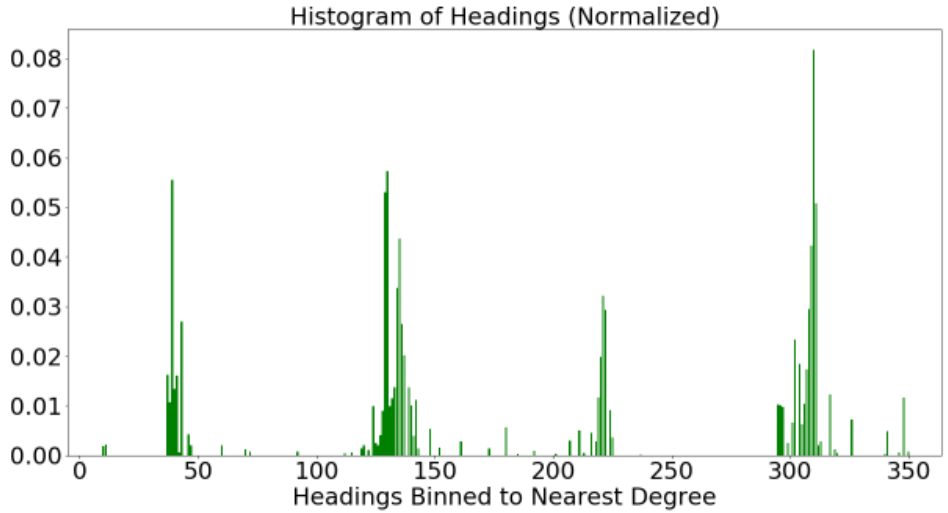


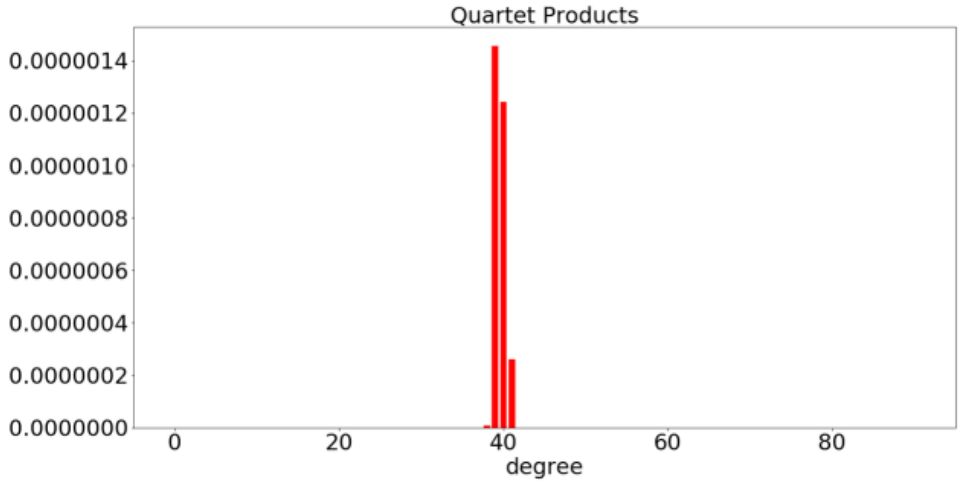
“Boxiness” Detection

Goal: Detect trajectories moving in nearly perfect box-like patterns.



headings histogram





Quartet Convolution

Multiple headings histogram values at 90° intervals.

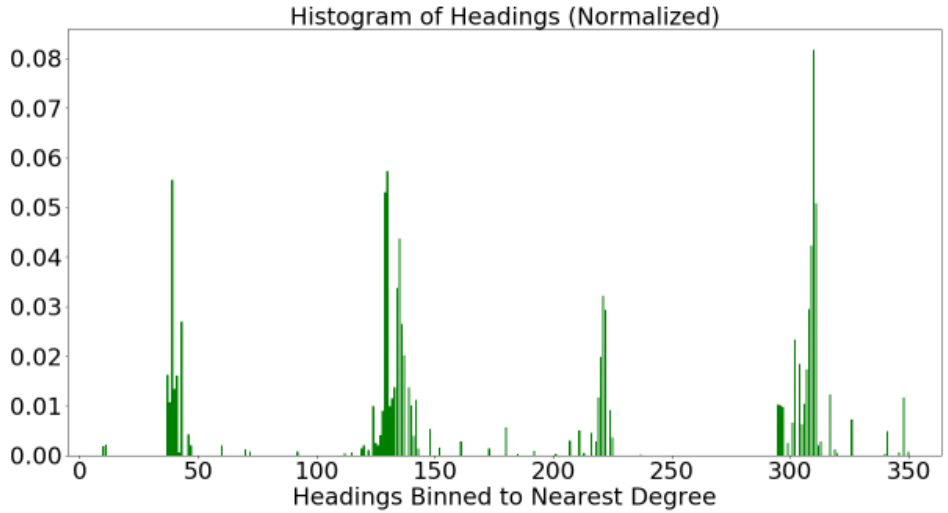
For example, value on plot to the left at 40° is the product of headings at 40°, 130°, 220° and 310°.

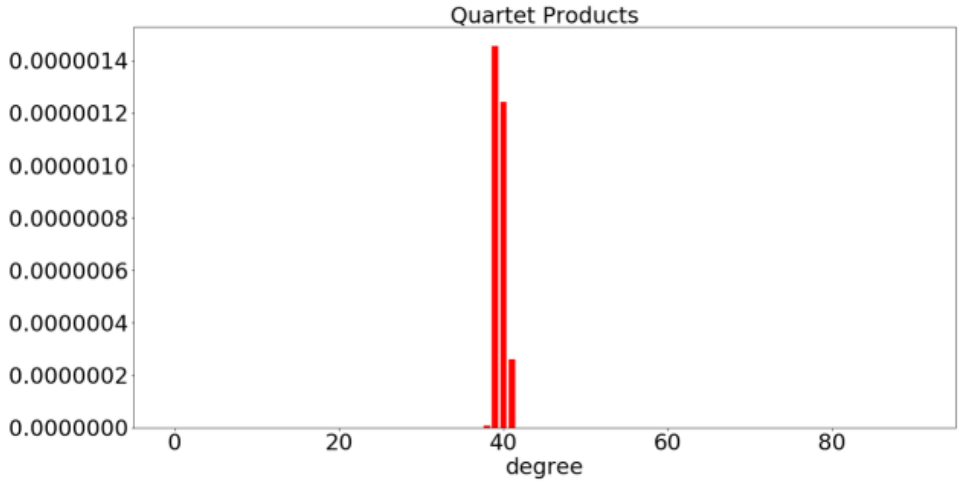
Boxiness = max value on the left plot, summed over a 5° interval (±2°)

“Boxiness” Detection

Goal: Detect trajectories moving in nearly perfect box-like patterns.

headings histogram





Quartet Convolution

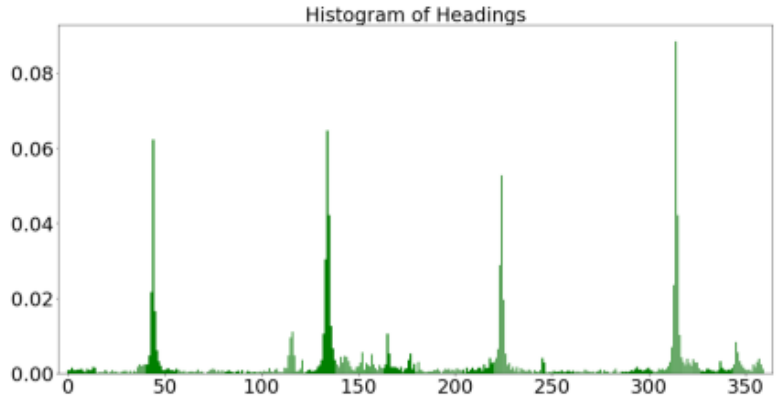
Multiple headings histogram values at 90° intervals.

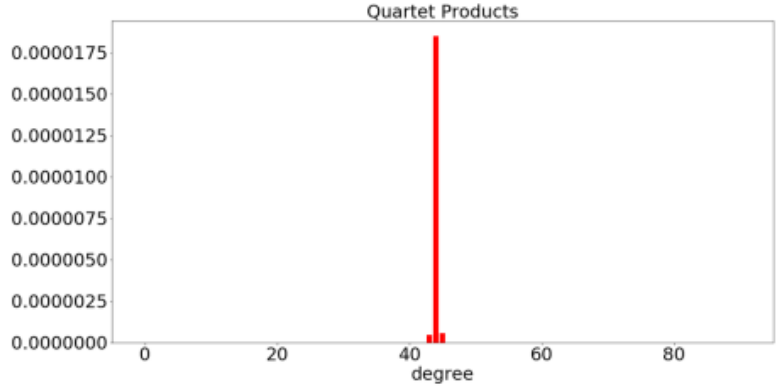
For example, value on plot to the left at 40° is the product of headings at 40°, 130°, 220° and 310°.

Boxiness = max value on the left plot, summed over a 5° interval (±2°)

Boxiest Trajectory

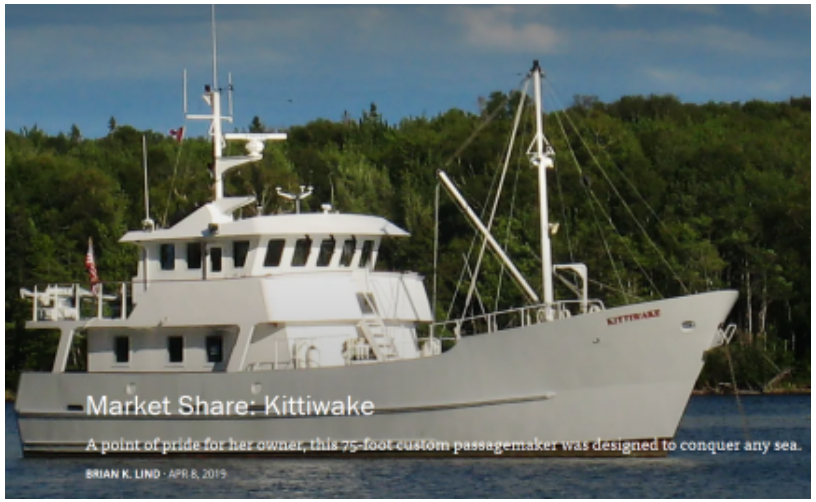
KOMMANDOR IONA is a **geophysical survey vessel** studying the ocean to the southeast of Rhode Island for the planned **Ocean Wind Project**.





The wind farm is expected to be operational and deliver 1,100 MW of energy by 2024.

Yacht for Sale



On April 8, 2018, the yacht KITTIWAKE was listed for sale.

On April 17, 19, and 23, KITTIWAKE traveled nearly identical trajectories in the Providence River (off the coast of Rhode Island) forming an **almost perfect box** each time.

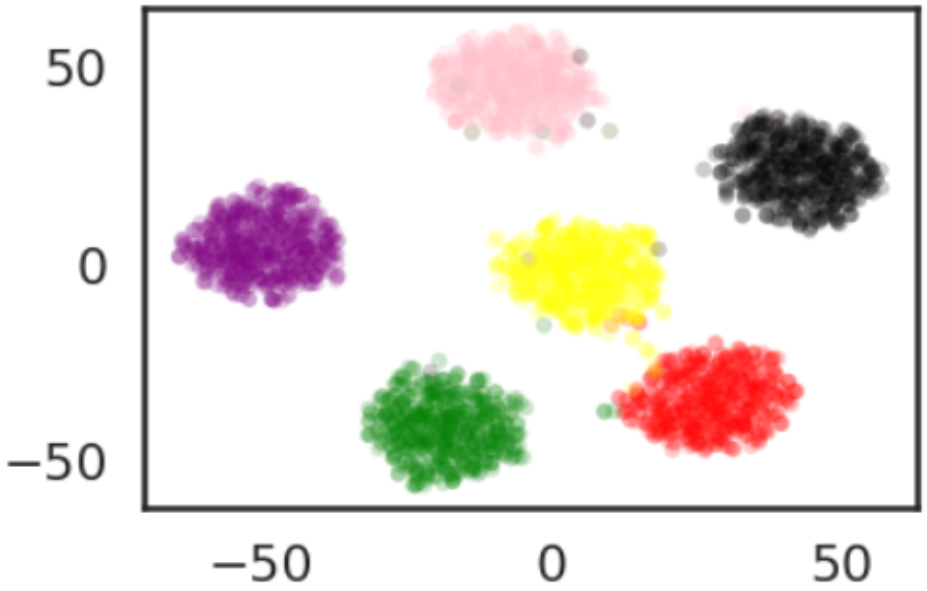
DBSCAN vs. HDBSCAN

Choice of clustering algorithm is important

DBSCAN tends to do better when clustering is easier, and some information is known about the cluster sizes and densities

HDBSCAN does much better when there is less known

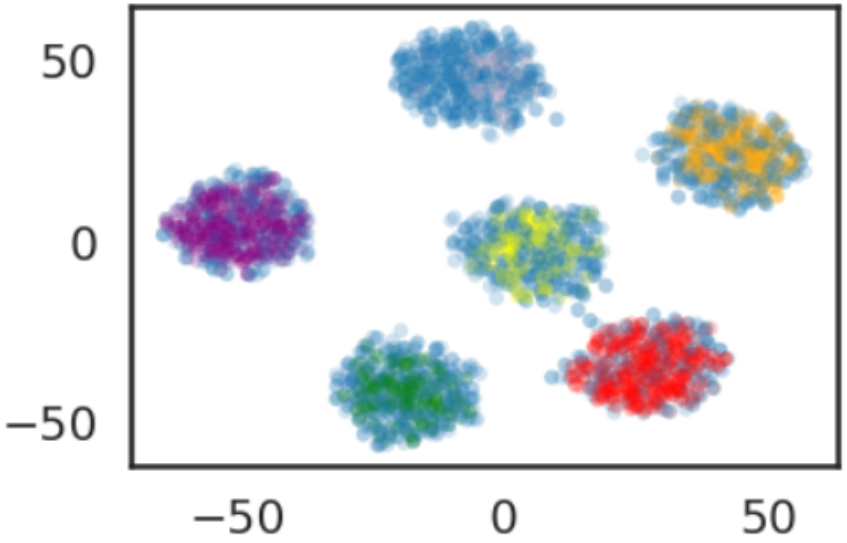
- Generally much more applicable in real-life scenarios



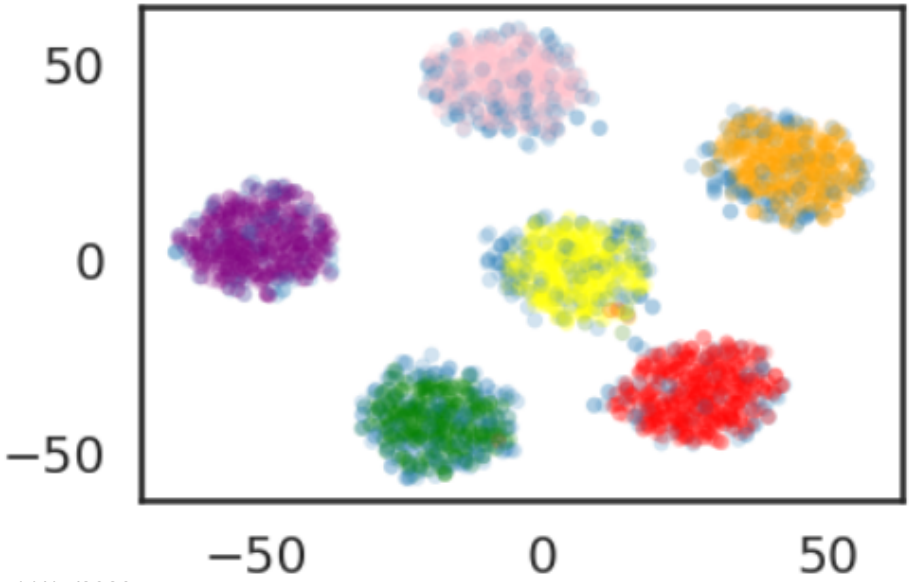
Ground Truth

- 10 dimensional data (projected to 2)
- 6 clusters

DBSCAN – Results

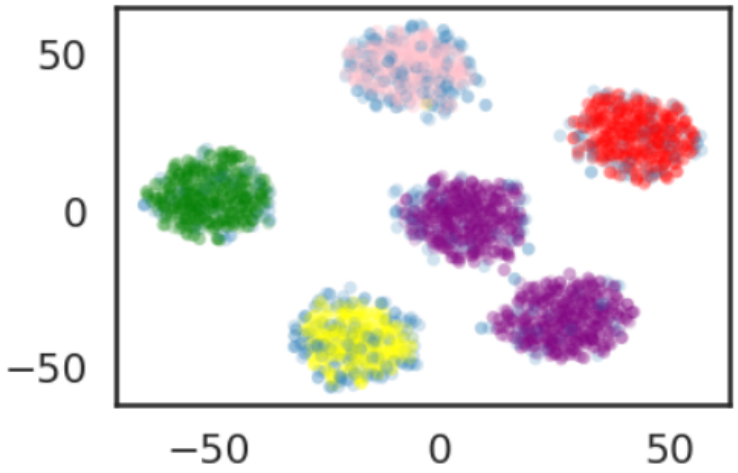


Right: Search_box = 0.2
Min_clust_size = 50
6 Clusters found
ARI = 0.213
AMI = 0.523



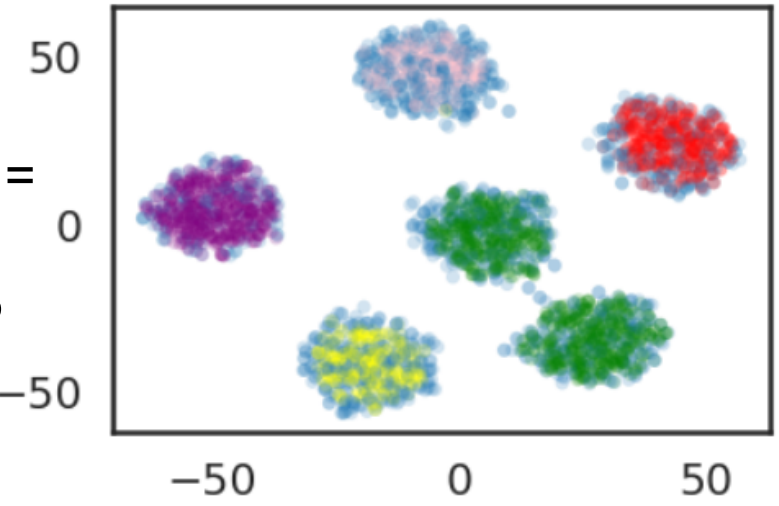
Left: Search_box = 0.2
Min_clust_size = 25
6 Clusters found
ARI = 0.661
AMI = 0.759

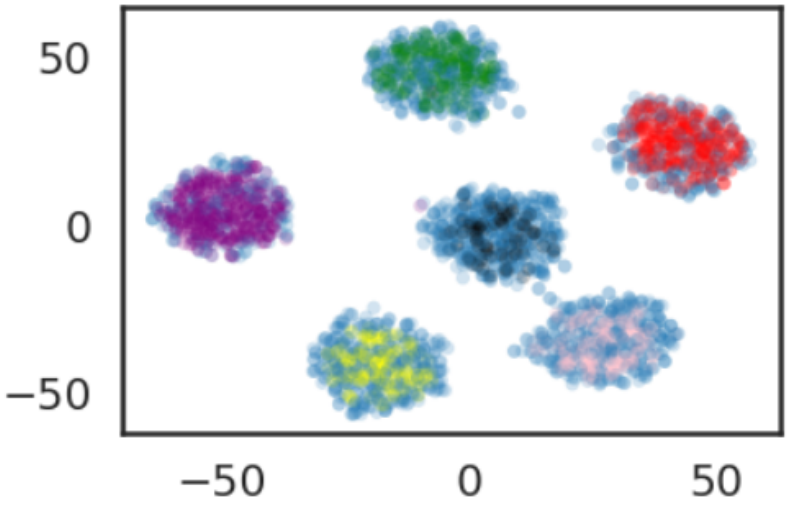
HDBSCAN – Results



Left: Min_Clust = 50
Min_samples = 1
5 Clusters found
ARI = 0.637
AMI = 0.755

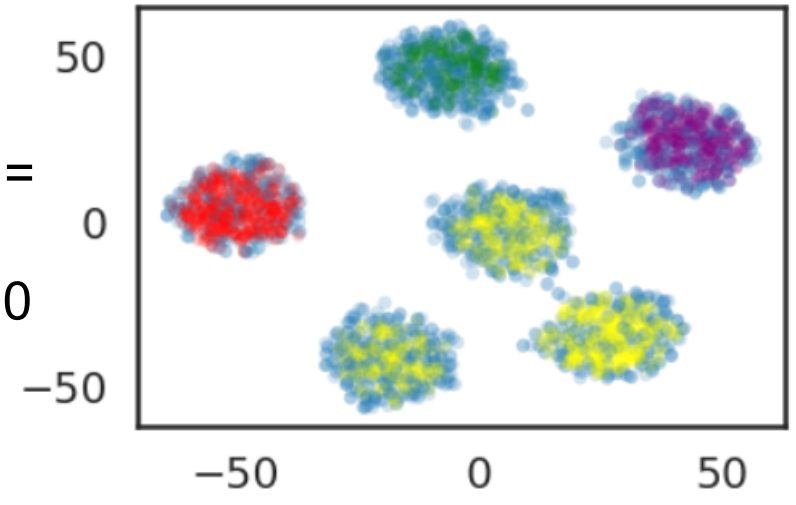
Right: Min_Clust = 50
Min_samples = 5
5 Clusters found
ARI = 0.347
AMI = 0.580





Left: Min_Clust = 50
Min_samples = 10
6 Clusters found
ARI = 0.216
AMI = 0.530

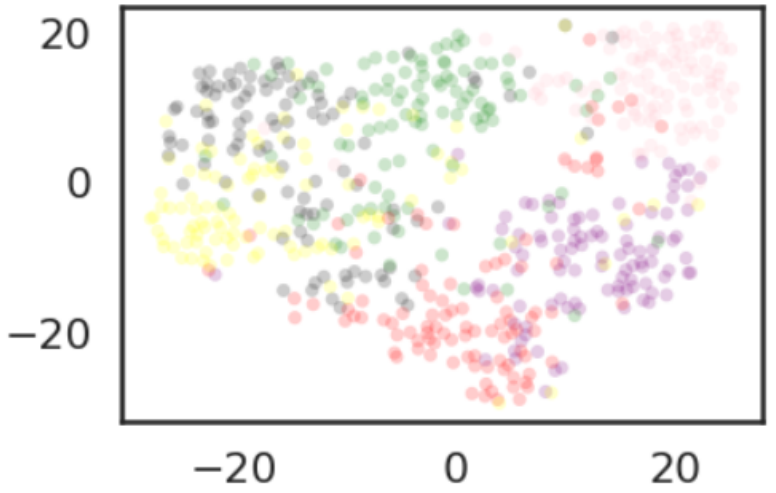
Right: Min_Clust = 50
Min_samples = 20
4 Clusters found
ARI = 0.173
AMI = 0.424



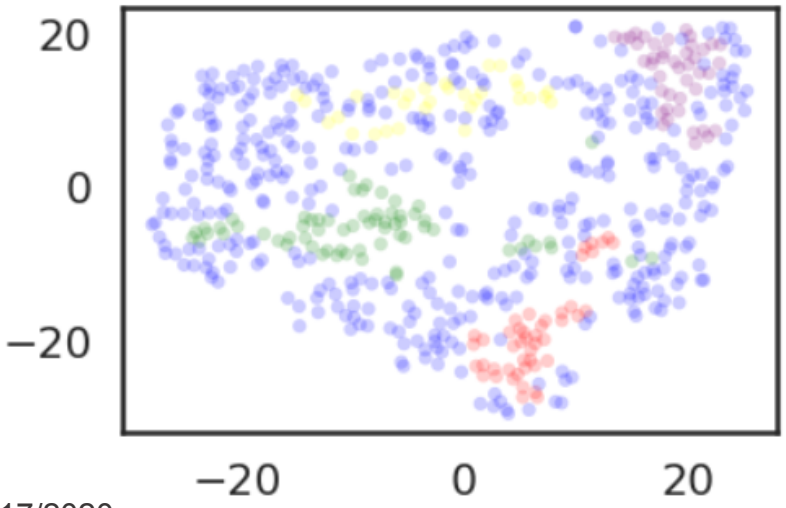
More Complex Data

With more realistic data, HDBSCAN generally performs better with much less human intervention

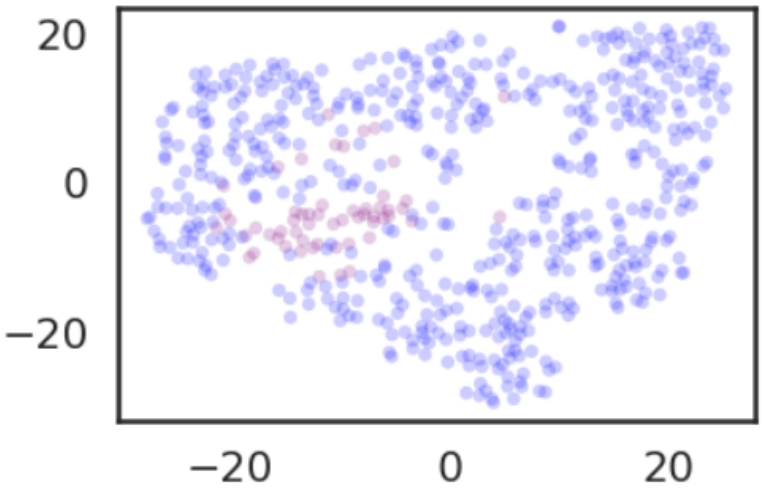
Ground Truth
- 4-d clustering
- 6 clusters, lots of overlap
- Much more similar to trajectories



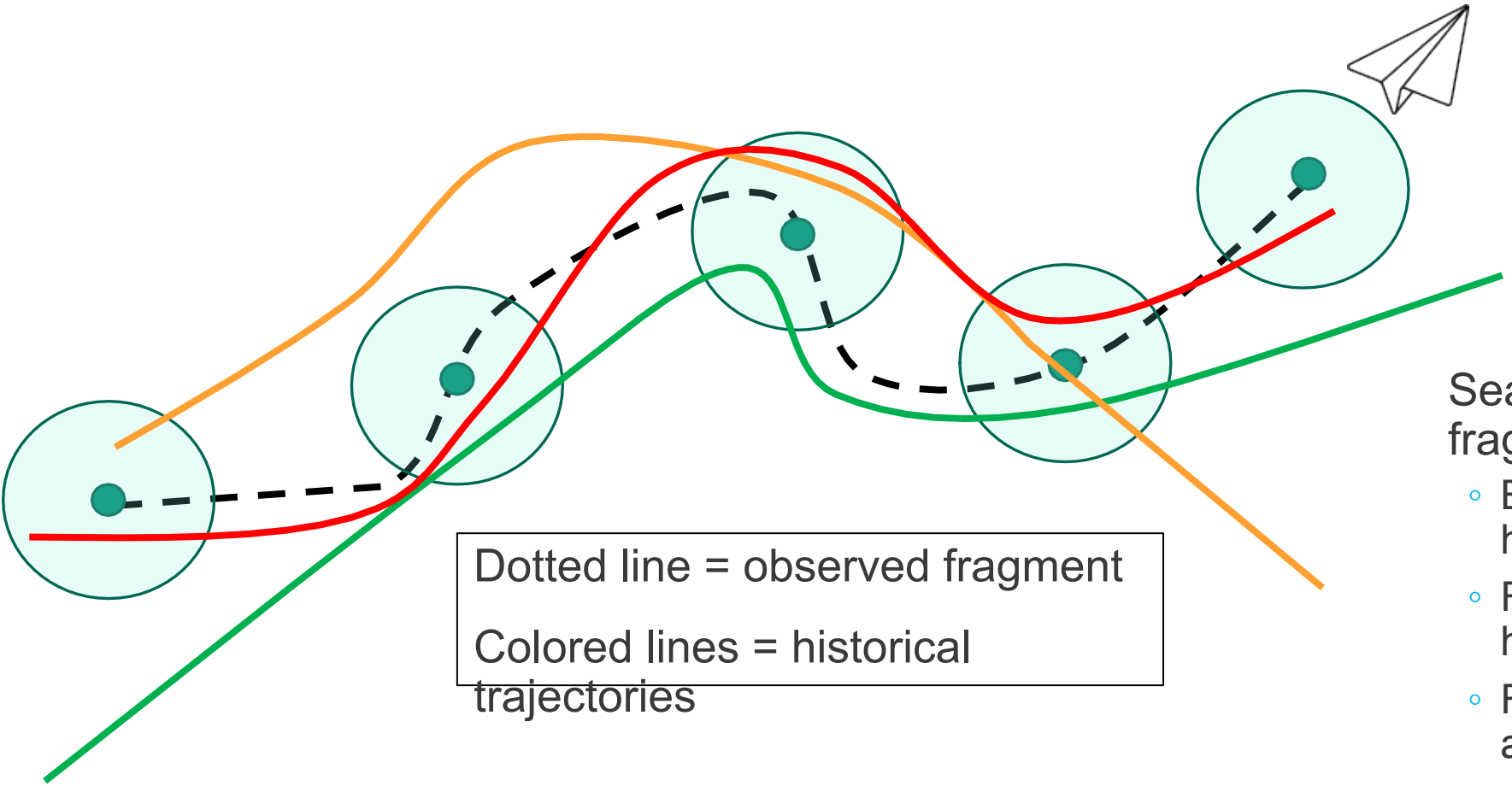
Left: HDBSCAN
min_clust = 25
min_samples = 1
ARI = 0.034, **AMI = 0.195 found 4**



Right: DBSCAN
min_clust = 25
serach_box = 0.25
ARI = 0.006, **AMI = 0.042 found 1**



Fragment Alignment for Prediction



Dotted line = observed fragment

Colored lines = historical trajectories

Green	Orange	Red
3	5	3

Searching historical data to match fragment

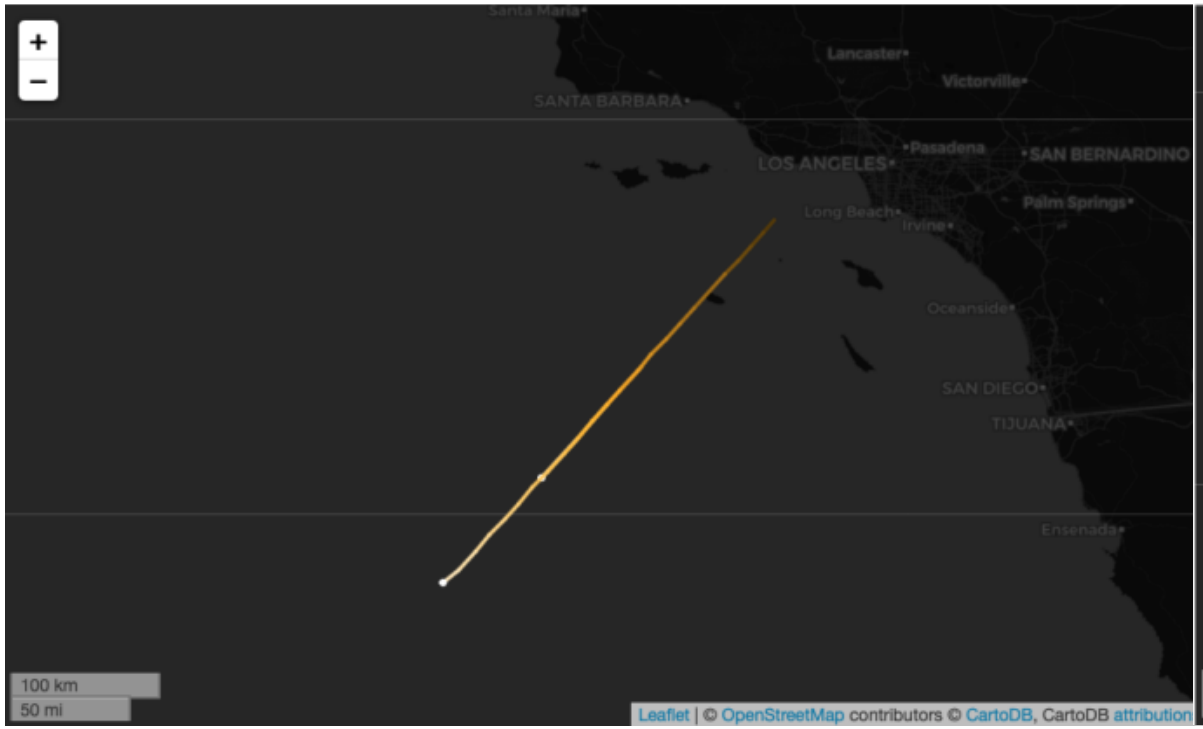
- Break fragment up even points (5 here)
- Find all historical trajectories that had points within radius *r*
- Find all trajectories that matched all *n* points (or *n*-1…)
- Rank them based on some metric (e.g. inverse of the largest point-trajectory distance)

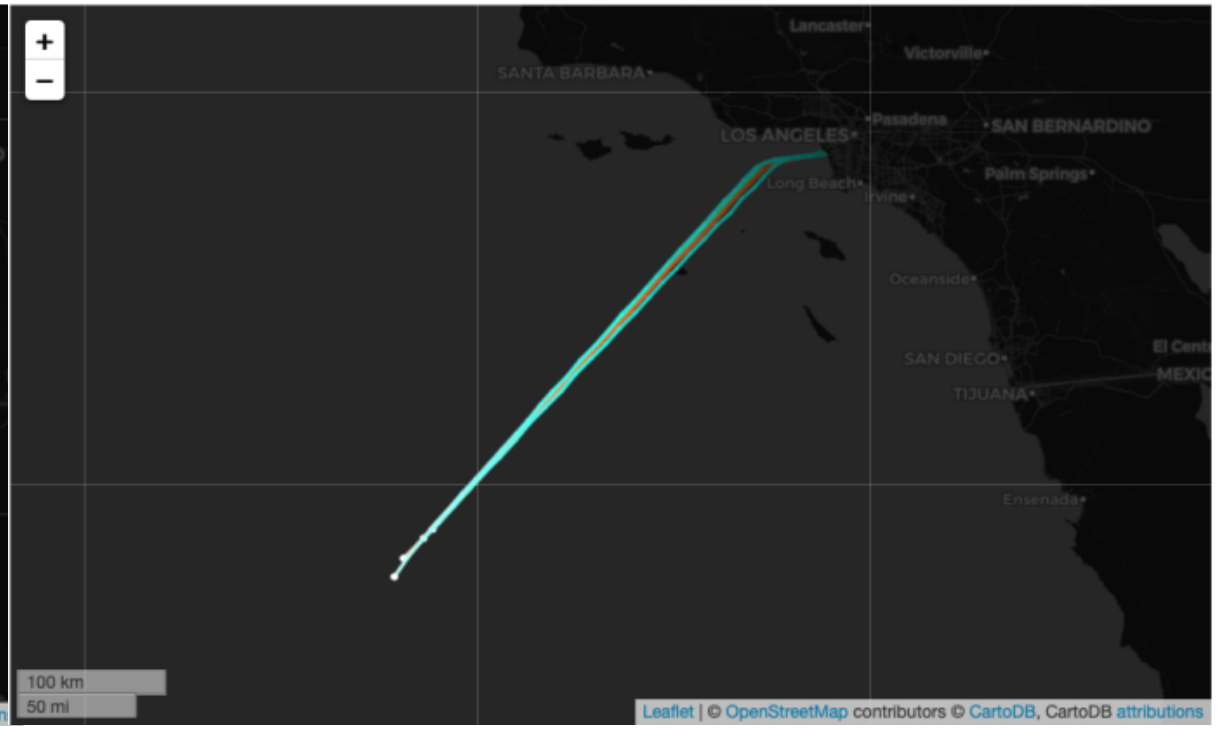
Surprisingly fast (~1s for 100k trajectories)

Prediction Examples

LA to HI is pretty straightforward

Unique for the area



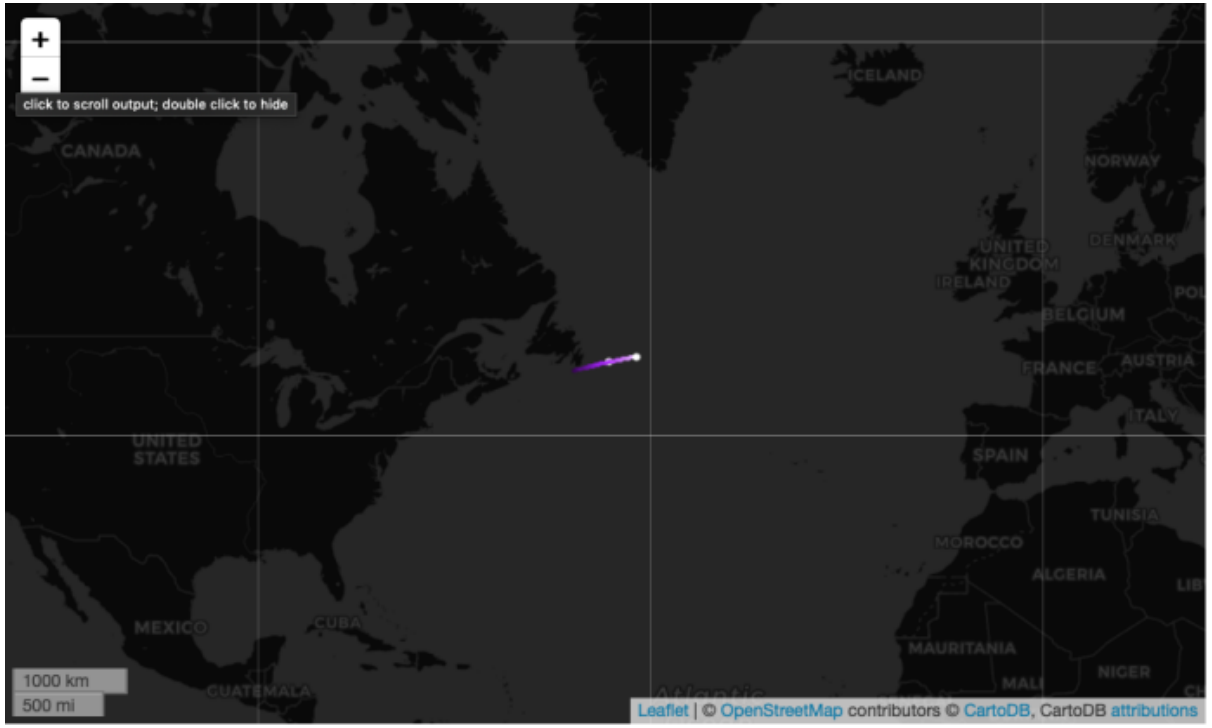


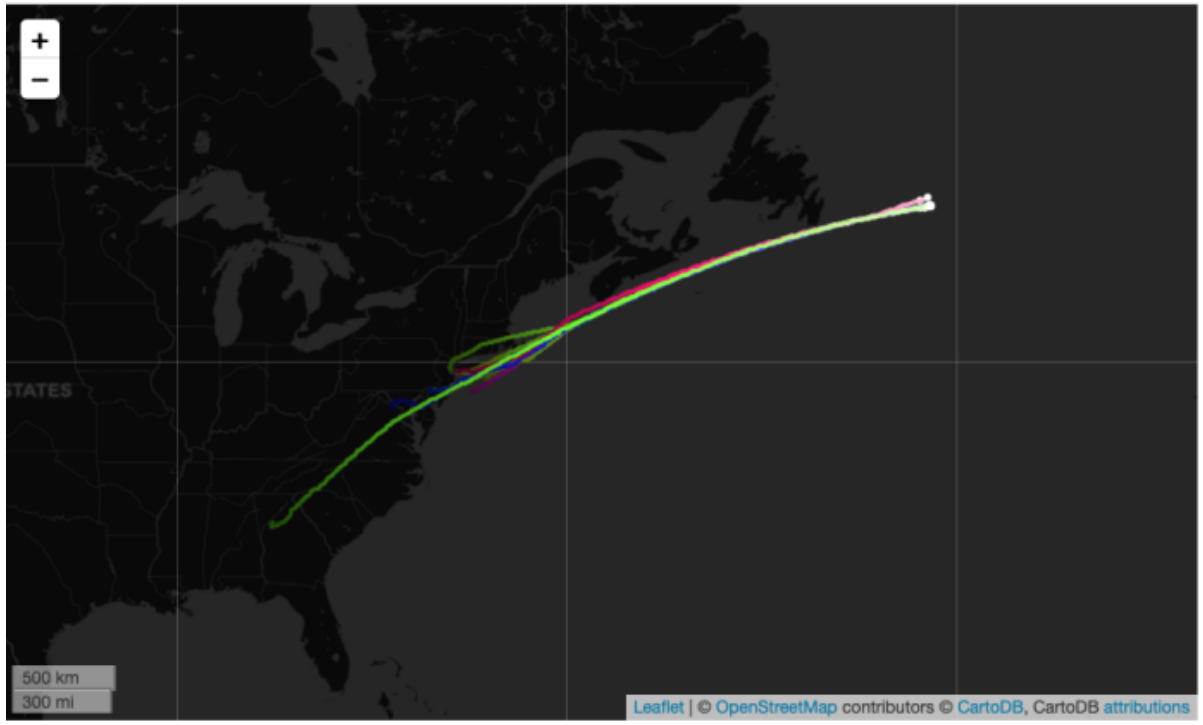
Prediction Examples

NY to Istanbul a little harder

Everything gathers in the same lane

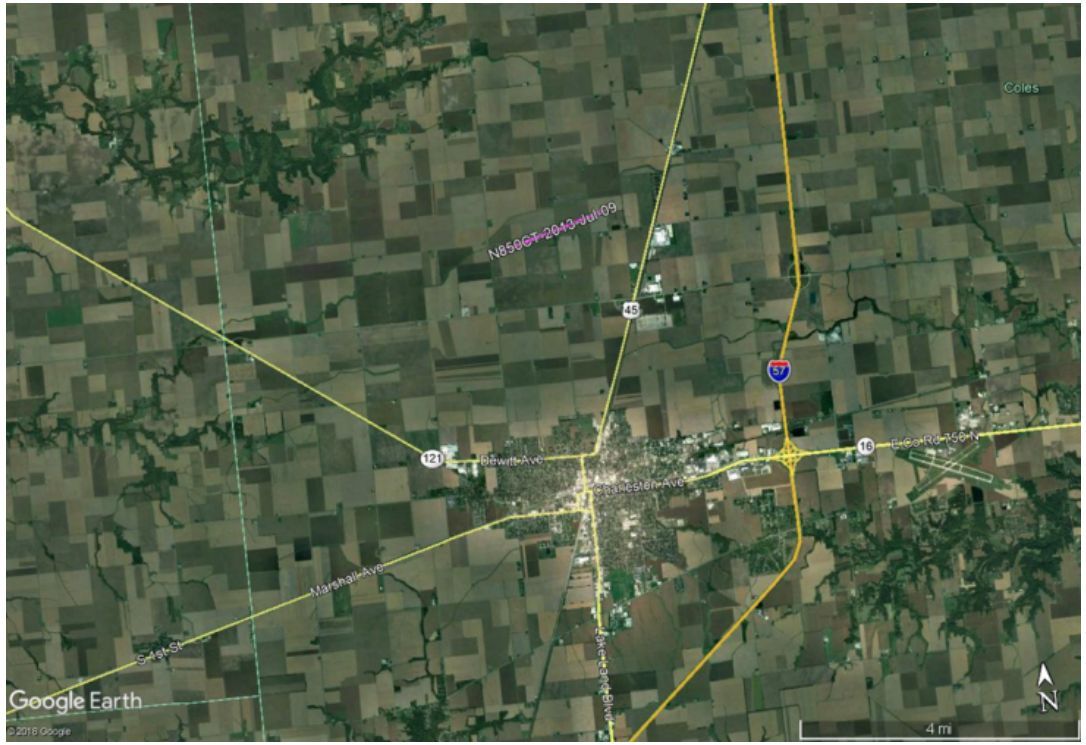
But, can offer probabilities based on historical data, **and**, suggest when you can make better guess

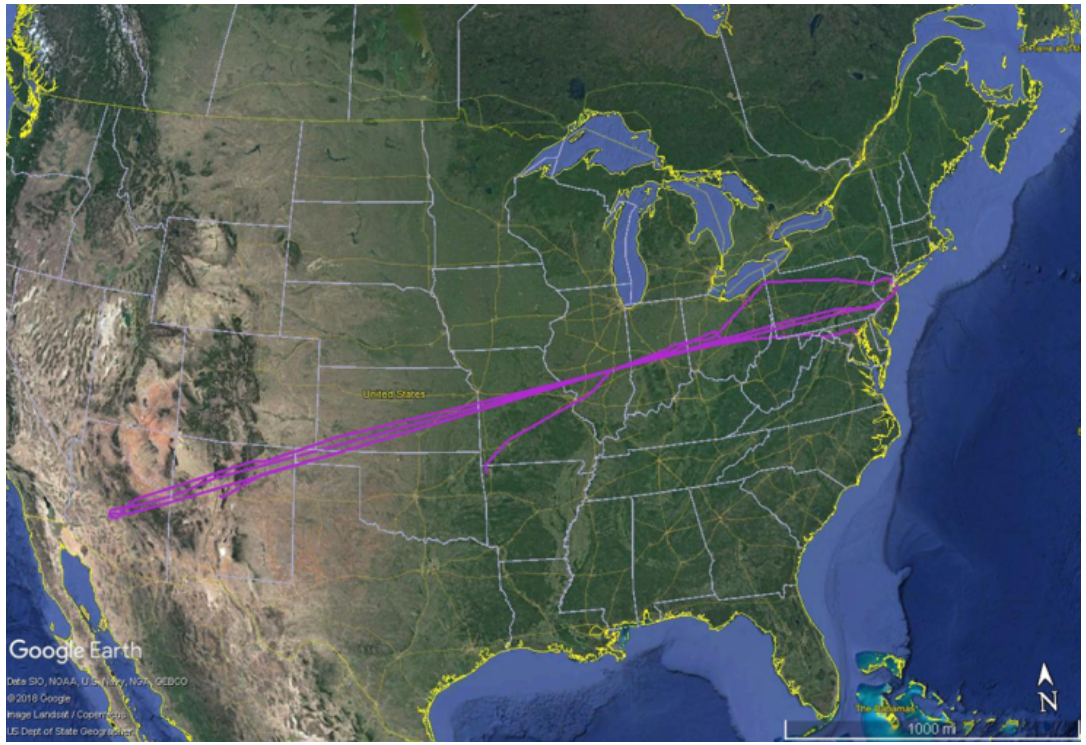




Work in Prediction Based on Historical Data

Even a 2.5 km fragment from Mattoon, IL can be used to know that this is a flight that is probably going to PHX





Tracktable: The Software

C++ for speed

- Core operations (math, clustering, geometry, trajectory manipulation, file parsing) written in C++11
  - Required language version is deliberately old to allow us to work in constrained environments
- C++ API is exposed and supported

Python for glue and flexibility

- I/O from diverse sources, including databases
- Render trajectories to images, movies, and interactive maps
- All capabilities in C++ components available in Python
- New algorithms prototyped in Python, then migrated to C++ if necessary
- Python interface lets us operate in Jupyter

Tracktable

Tracktable: Deployment

Option 1: Python Wheels (binary install packages)

- Package up Python code and C++ extension libraries
- Dependency libraries take careful management
- Build for multiple platforms, multiple Python versions
- ***These are a game-changer in their simplicity for users.***
- $ pip install tracktable
- import tracktable.examples (lots of Jupyter notebook examples made!)

Option 2: Build from Source

- Sometimes you just can't use a binary installer
- Relatively modern tools (CMake, Boost) with relatively lax version requirements
  - Constrained dev environments are often very slow to update
- CMake gives us cross-platform build capability

Documentation – https://tracktable.readthedocs.io

Summary

We need to get to a place where the computer is doing more of the work

- Need to find a way to better describe trajectories
- Need to leverage the fantastic work being done right now in ML

We need to find a way to deploy this to the current infrastructure in a way that it is easy to use

This requires a highly interdisciplinary team, with very broad backgrounds