Bird Migration Network

Exploring Effects on Migration Patterns and Hotspots

Anshul Choudhary - 21110028
Pratham Sagar - 21110165
Yashraj Deshmukh - 21110245

Preliminary Explorations

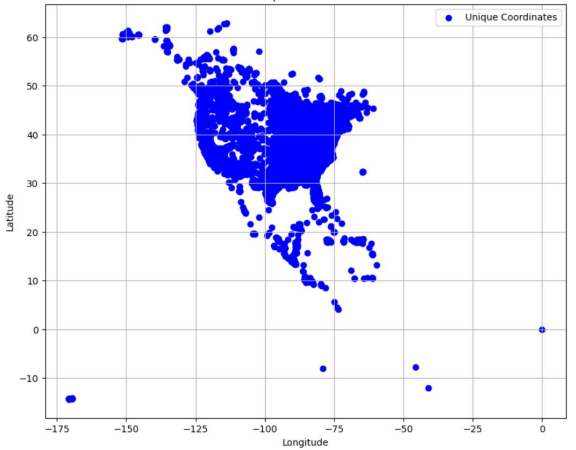
We worked on a csv file which had movement data on Cuckoos, Nightjars, Swifts, Hummingbirds in USA and tried out some testing on it. We did some preprocessing on the data to get rid of coordinate redundancy and nan values.

```
data.shape
```
✓ 0.0s
```
(1101311, 27)
```

```
coord = data[['LAT_DD', 'LON_DD']].apply(tuple, axis=1)
coord.dropna(inplace=True)
coord.nunique()
```
✓ 6.1s
```
10481
```

No. of clusters using DBSCAN algorithm (Max. dist. = 0.2 units)

```
Number of unique clusters/hubs: 1969
```



What we are the nodes of our network?

Nodes are important bird areas.

How many nodes are we expecting?

We are expecting around 10,000 nodes spread mostly over the American and Asian continents, analysing potential hotspots for few species of migratory birds.

What are the links/edges in the network?

Edges will represent bird movement from one area to another. It consists of two attributes, namely, bird species and number of birds.

Why to care?

Many migratory bird species are deeply affected by human activities and climate change, some even on the verge of extinction in recent years.
Through this project we wish to come up with a network which can be further developed to acquaint many more species and eventually save birds.

Do we have the data?

Need to find: Data used in Lee M, et al. (2007) Characterization of an H5N1 avian influenza virus from Taiwan. Vet 212 Microbiol 124(3):193-201.

Need to request: from datazone.birdlife.org (i) Species range map (ii) IBA boundary map

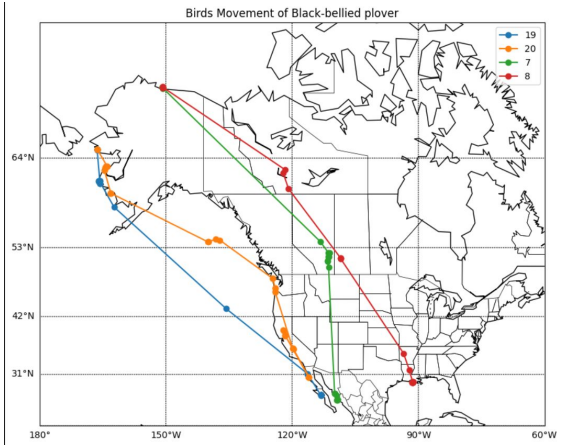
Currently Experimented on: Celis-Murillo, A., Malorodova, M., and Nakash, E., 2022, North American Bird Banding Program Dataset 1960-2022 retrieved 2022-07-14: U.S. Geological Survey data release, https://doi.org/10.5066/P9BSM38F.

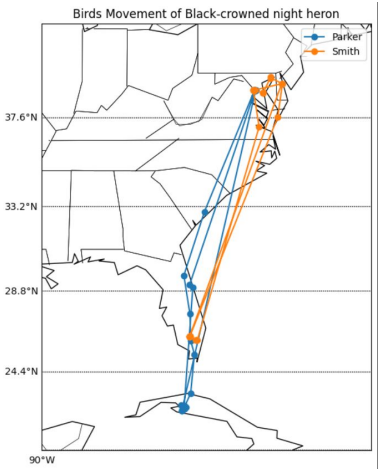
Data Lookout

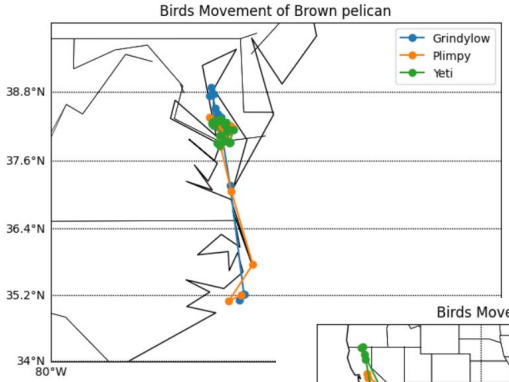
Smithsonian National Zoo data
BirdLife DataZone
MoveBank data

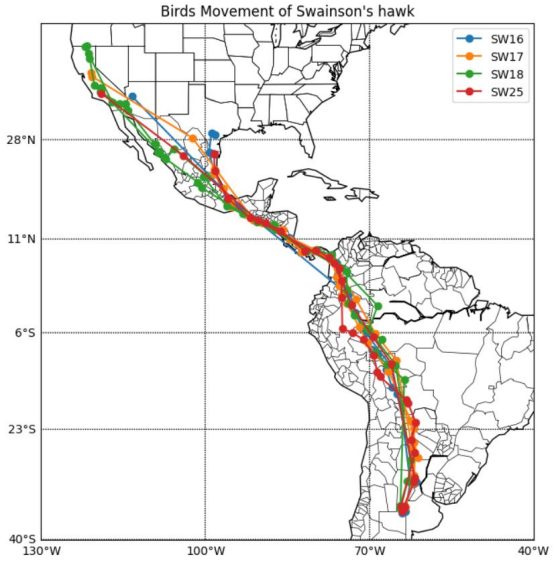
Smithsonian National Zoo data

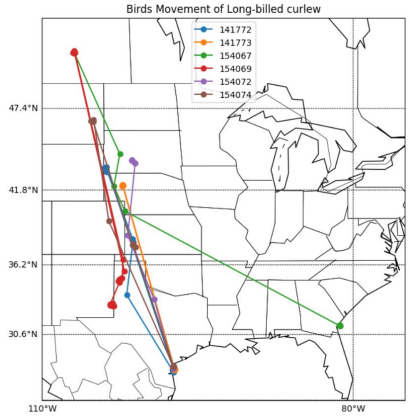
```
Species 'Black-bellied plover' has duplicate (birdID, date) pairs.
Species 'Black-crowned night heron' has duplicate (birdID, date) pairs.
Species 'Brown pelican' has duplicate (birdID, date) pairs.
Species 'Long-billed curlew' has duplicate (birdID, date) pairs.
Species 'Pacific loon' has duplicate (birdID, date) pairs.
Species 'Swainson's hawk' does not have duplicate (birdID, date) pairs.
```

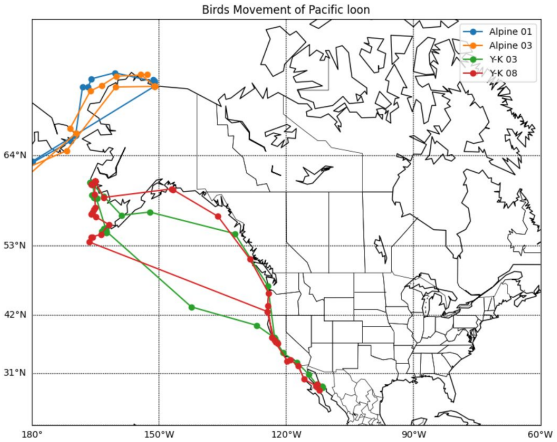












Taking a closer look at Swainson's hawk

SW16

SW17

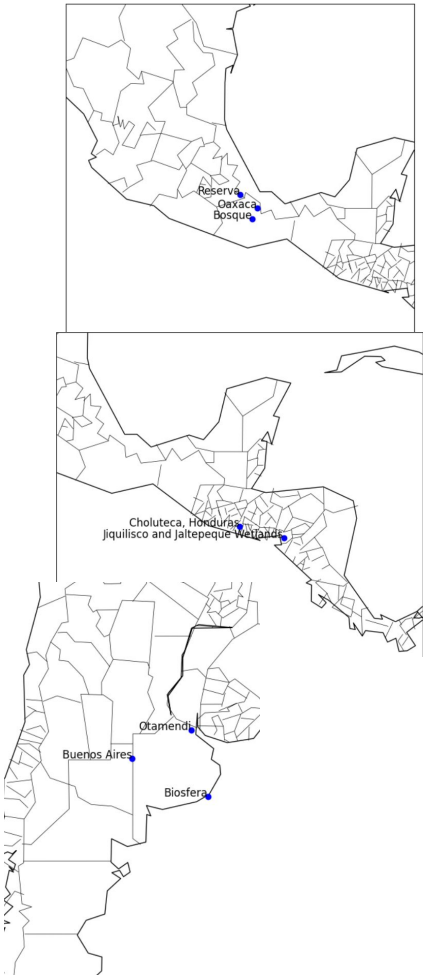
SW18

SW25

Attempt to create a common path

SW16						
Date	Long	Lat	Name	Potential IBA	Long	Lat
08-10-1996	-98.184	28.84	Laredo, Texas, USA	Lower Rio Grande Valley National Wildlife Refuge in Texas, USA	-97.852	26.187
09-10-1996	-98.877	29.12	San Antonio, Texas, USA			
10-10-1996	-99.299	25.879	Nuevo Laredo, Tamaulipas, Mexico			
11-10-1996	-98.326	22.77	Ciudad Victoria, Tamaulipas, Mexico			
12-10-1996	-96.021	17.831	Oaxaca City, Oaxaca, Mexico	Bosque Mesófilo de Montaña de la Sierra Madre de Oaxaca (Mexico)	-96.31	17.228
21-10-1996	-84.559	11.255	Bluefields, Nicaragua			
24-10-1996	-79.887	8.916	Panama City, Panama			
25-10-1996	-79.943	8.887	Panama City, Panama			
27-10-1996	-76.006	6.643	Quibdó, Colombia			
28-10-1996	-75.806	4.609	Medellín, Colombia			
31-10-1996	-73.646	1.248	Bogotá, Colombia			
01-11-1996	-72.574	-2.357	Leticia, Amazonas, Colombia			
03-11-1996	-71.38	-5.287	Iquitos, Loreto, Peru			
04-11-1996	-69.638	-8.95	Pucallpa, Ucayali, Peru			
05-11-1996	-68.592	-9.893	Porto Velho, Rondônia, Brazil			
07-11-1996	-66.924	-12.65	Cochabamba, Bolivia			
08-11-1996	-65.919	-15.806	Santa Cruz de la Sierra, Bolivia			
09-11-1996	-65.005	-16.888	Puerto Suárez, Santa Cruz, Bolivia			
11-11-1996	-64.539	-17.066	Corumbá, Mato Grosso do Sul, Brazil			
16-11-1996	-62.602	-31.517	Córdoba, Argentina			
03-12-1996	-63.442	-35.913	Buenos Aires, Argentina			
04-12-1996	-64.093	-36.022	Rosario, Santa Fe, Argentina			
09-12-1996	-64.143	-36.172	San Nicolás de los Arroyos, Buenos Aires, Argentina			
21-12-1996	-63.687	-35.86	Pergamino, Buenos Aires, Argentina			
24-01-1997	-62.197	-31.829	Villa María, Córdoba, Argentina			

Was not fully created as only few points fell under some IBA

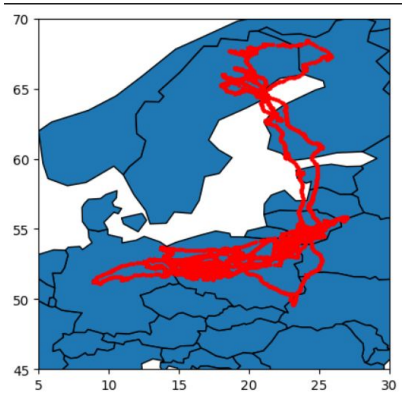


Checking proximity of collected data with IBAs

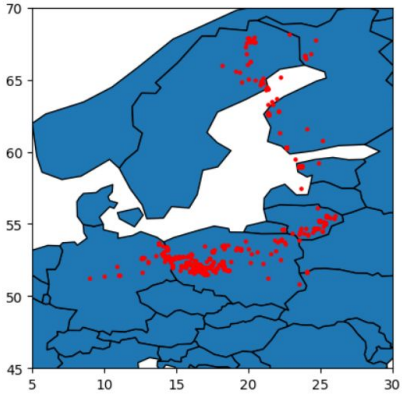
MoveBank data

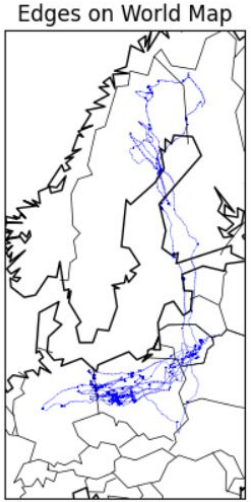
Data is location of multiple birds of same species (basically a scatter plot)

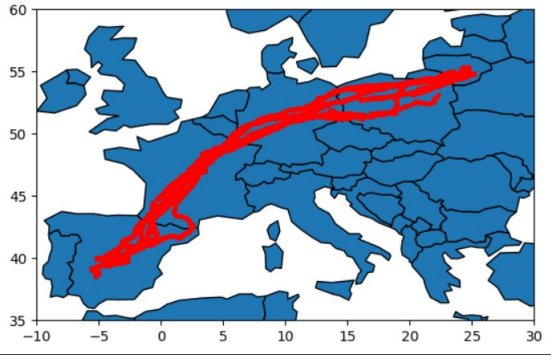
3 datasets on common crane over 3 different years

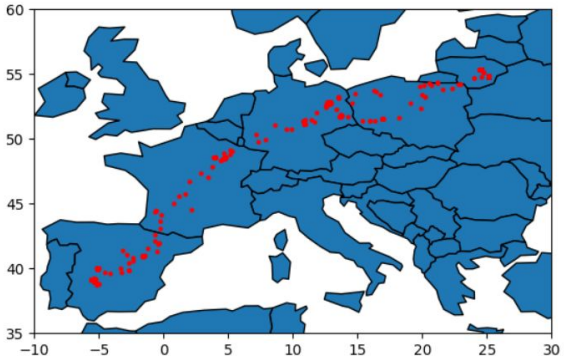


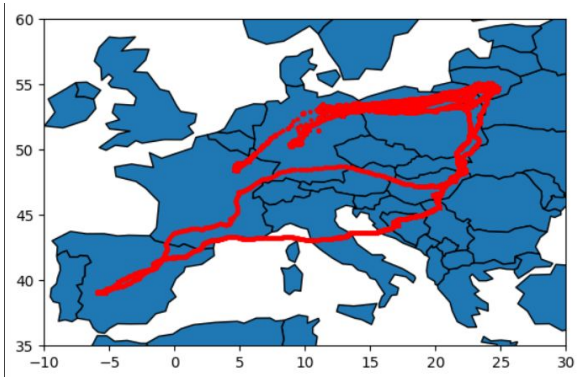
Daily-averaged locations

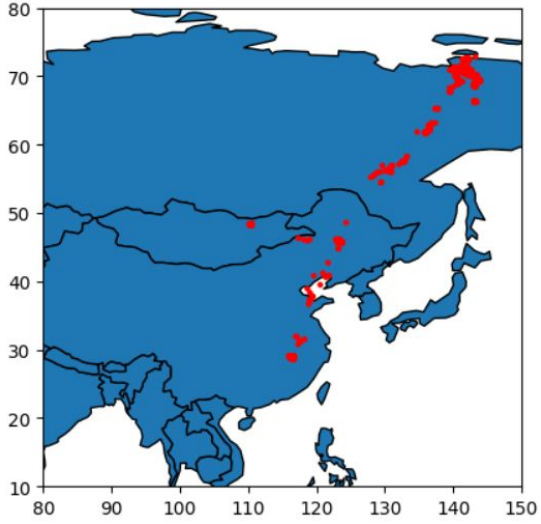




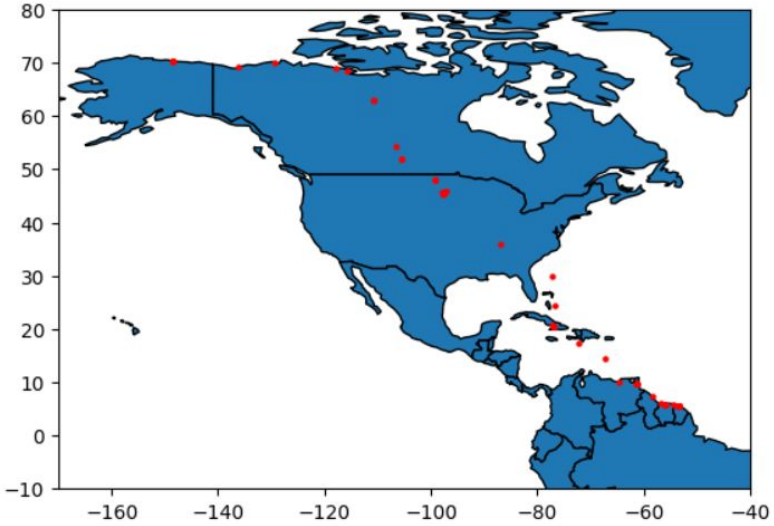




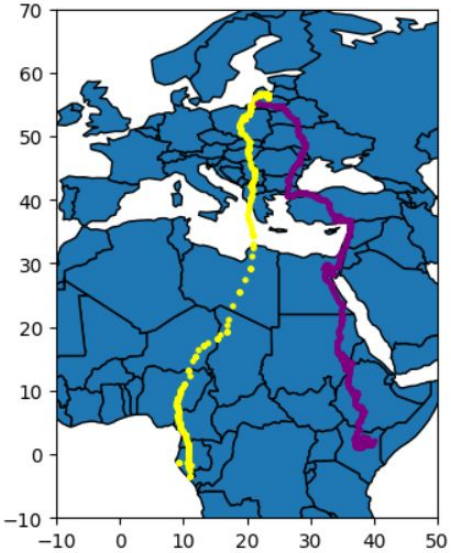




Siberian
Crane

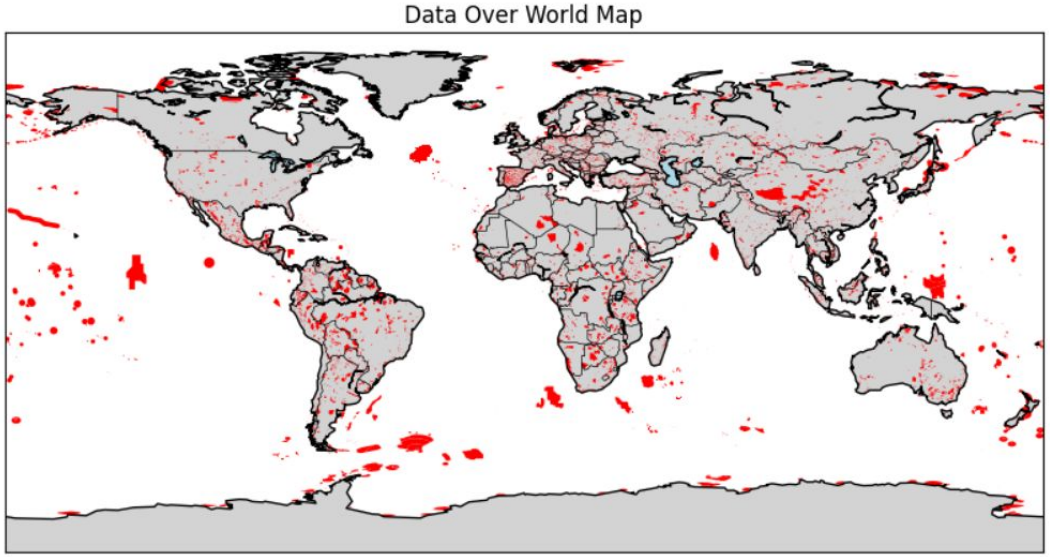


Arctic
Shoebird



Honey Buzzard
And
Spotted Eagle

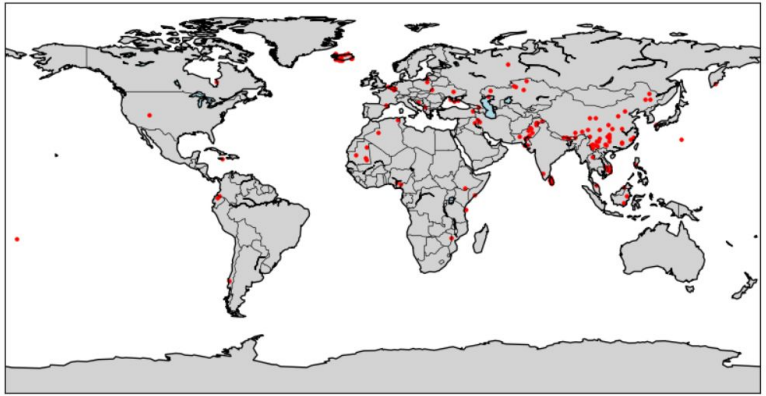
BirdLife Datazone IBA data



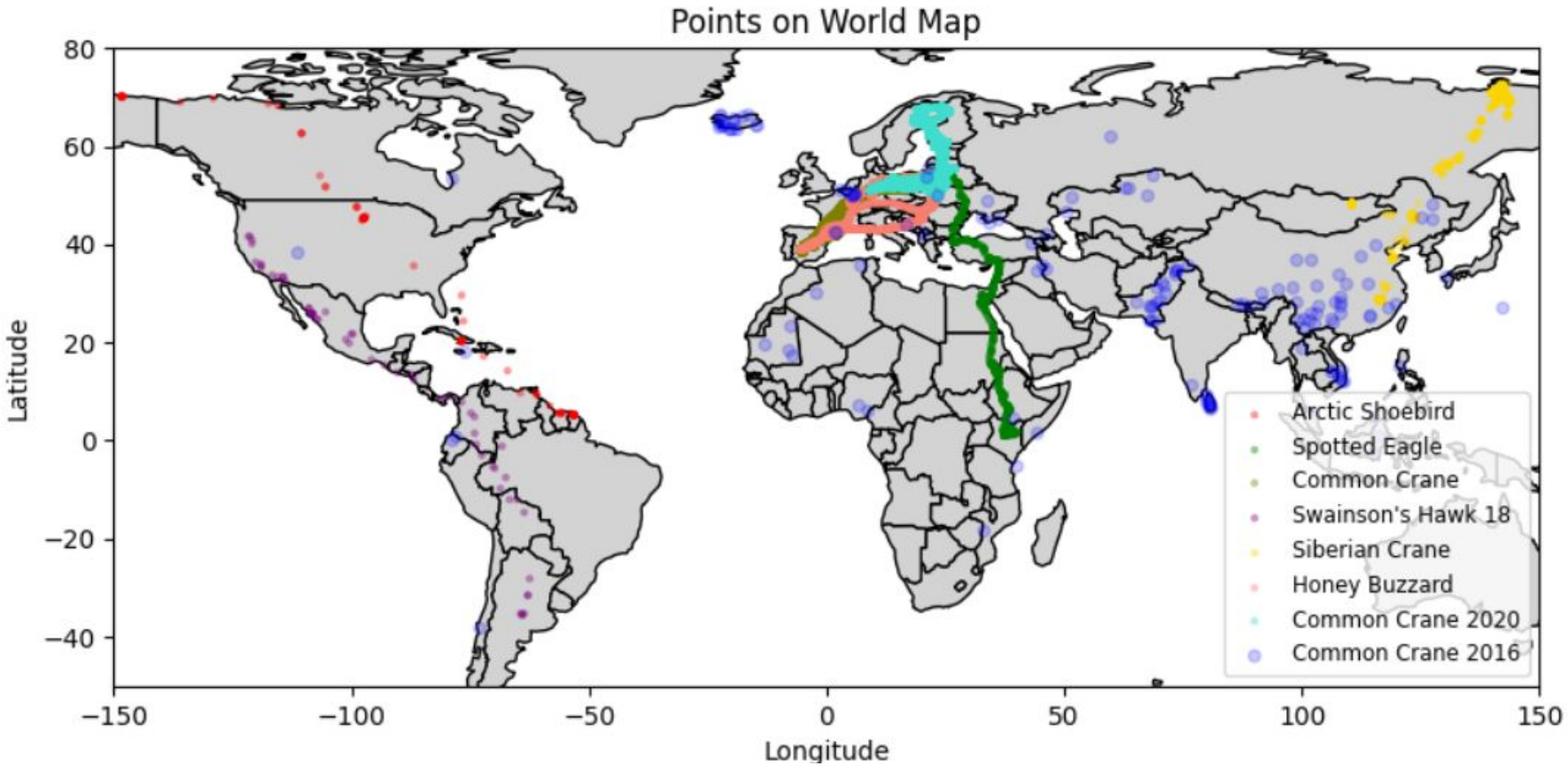
Polygon boundary

region spread

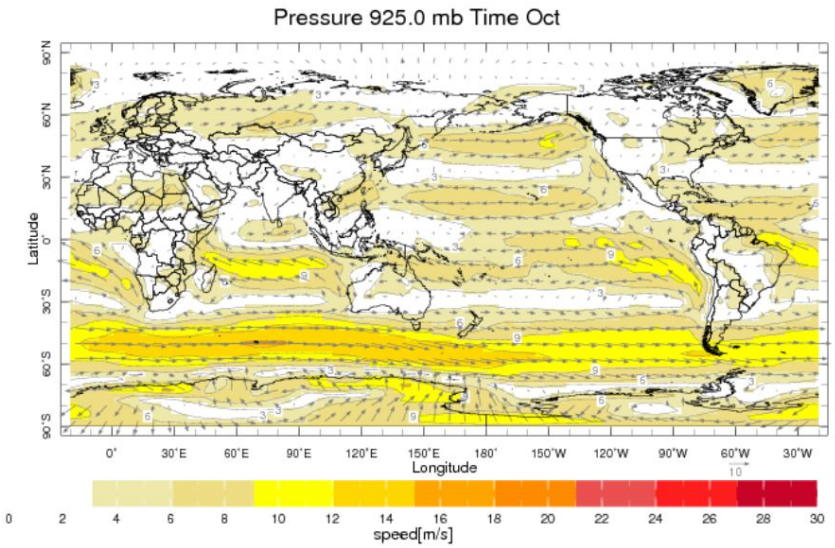
Point locations

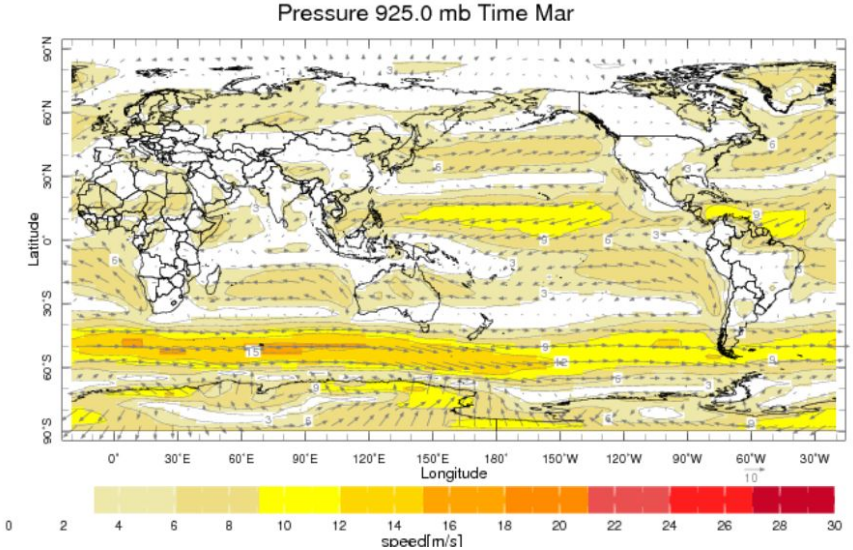


Overall View



https://iridl.ldeo.columbia.edu/maproom/Global/Climatologies/Vector_Winds.html?P=925&T=Mar





Network Construction

Haversine
Dispersal Probability
Swainson's Hawk

Github Repo: https://github.com/YYashraj/Bird-Migration-Network

What after the mid-semester roadblock?

After a not so great attempt at constructing an IBA-centered network involving birds world-wide, we decided to focus our attention first on one of the earliest data we acquired, the Smithsonian's data on Swainson's Hawk as it had an almost overlapping pattern amongst all its individual tracked birds so we planned/hoped to figure out what we could do for this species which we may later expand for few other aviators.

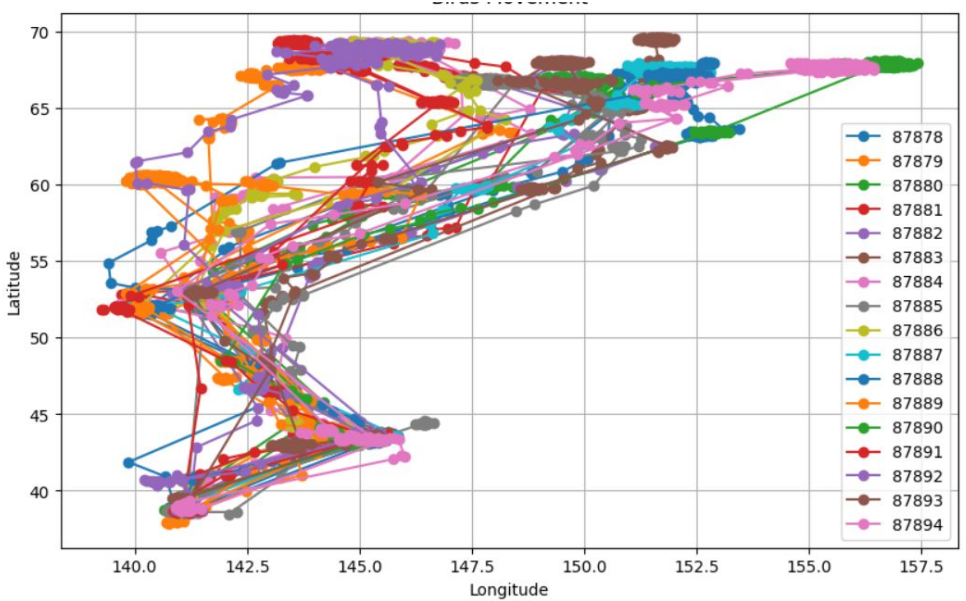
What mistake were we making

Well this dawned on me when I was watching the documentary on the movie night, we made the same mistake those folks made; we never plotted the degree distribution!

We plotted that for Whooper Swan (data from MoveBank). We had assumed that since we only got a set of coordinates of where a tracked bird had been, connecting consecutive *n* nodes would only result in just a trail with *n-1* edges, meaning almost all nodes would have *degree=2* (though that did turn out to be the case for Swainson's Hawk though, bummer).

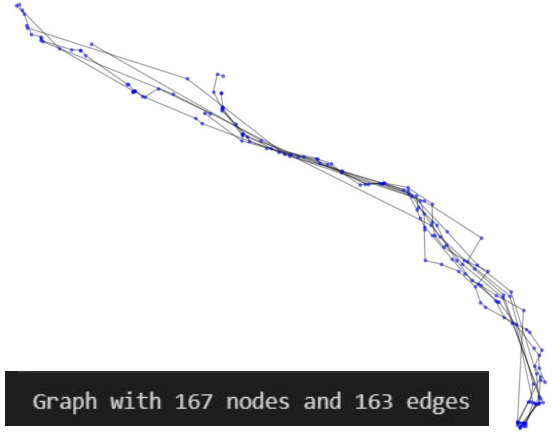
To our surprise and excitement we got a very different distribution then we had imagined. This thus meaned that different birds were indeed crossing over the same points/locations giving rise to some degree distribution, and in turn a 'network-esque' network.

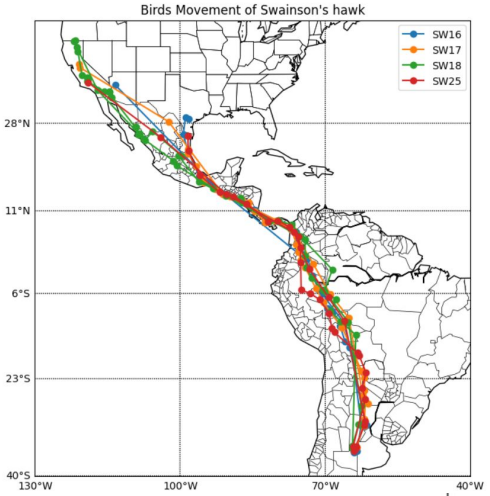
```
Degree 1: 39
Degree 2: 22871
Degree 4: 512
Degree 3: 9
Degree 6: 60
Degree 5: 4
Degree 8: 8
```

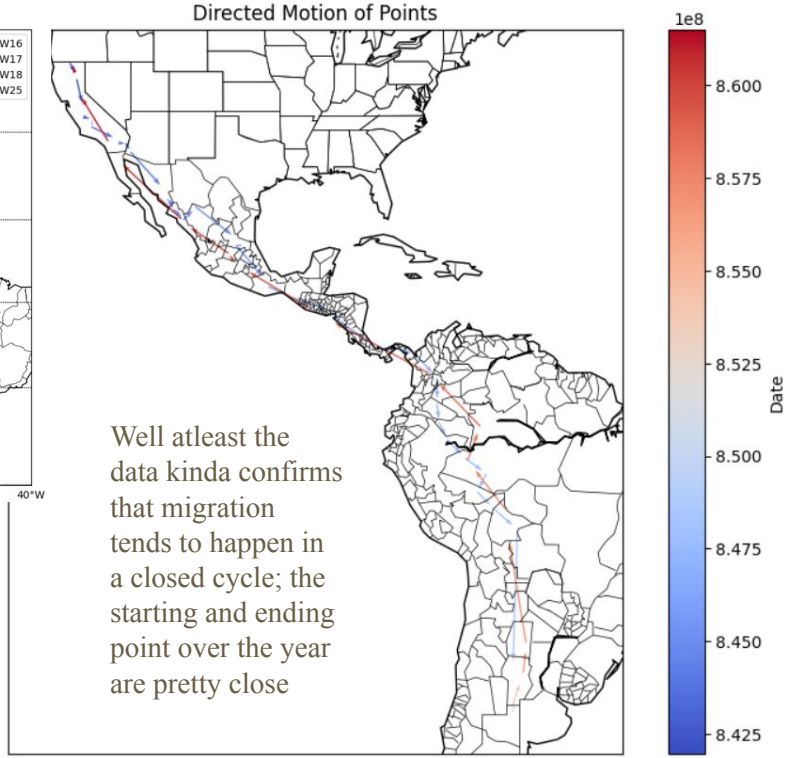



```
Graph with 23503 nodes and 24150 edges
```

Well with the success/ discovery on the previous data, we looked forward to achieve the same with SWH (Swainson's Hawk). However, we were left disappointed.







Well atleast the data kinda confirms that migration tends to happen in a closed cycle; the starting and ending point over the year are pretty close

This is nothing but ∑(n-1)=N-4. So basically our 'network' is just four independent trails. Not at all suitable for any network analysis.

A different attempt at network construction

It took few weeks but we came to terms with the fact that we won't readily find a data on the web that will map out a good network for us to work upon. What we have now, and probably still get even after digging more sites, are just coordinates of where a bird has been, effectively, our network's nodes. It was completely up to us now how we would create a network out of this data. (We found out pretty late that it was quite a common thing in network science research to face this problem; the edges in such cases would be constructed using some preconceived notion or hypothesis regarding the data in subject)

Some papers we came across used some probabilistic or mathematical model being used, after some node pre-processing and some additional constraints on links between nodes, to construct edges of the network. This could give a more network-like representation of the migratory route.

A different attempt at network construction

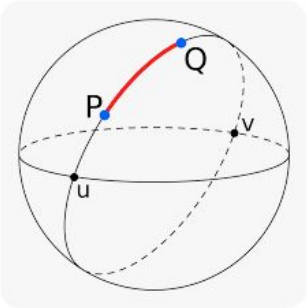
One such paper was **Detecting Critical Scales in Fragmented Landscapes** by Keitt et.al (https://www.ecologyandsociety.org/vol1/iss1/art4/). It developed a method to quantify habitat connectivity between different patches. It used distance as a parameter of probability of whether two nodes could be connected. We were planning to test this technique for our network construction, however we didn't do it as we later felt that the two problems were slightly different from each other. In their problem, few habitat spots were already forming clusters and their algorithm was more focused on finding inter-habitat relationships, which is not our case.

So instead we devised a different, and much simpler (why didn't we think of this earlier), approach to our network construction. Since all the tracked birds of SWH were of the same species they would move due to same reasons, for breeding or climate etc. This means that their weather requirements would be similar so it is quite possible for bird-1 to pass through few points that some other bird had passed through. Moreover, we found that SWH likes to be where it is summer. So when it leaves US or when it return back there, the weather of the places it's been to would be similar. So it can use nodes from one direction traversal in that of other direction travel as well. This was the reason to reduce our network from directed to undirected.

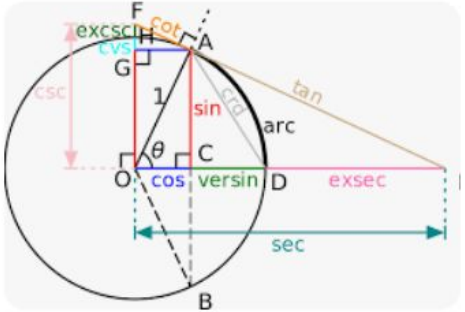
A different attempt at network construction

We defined an edge in our network as a link between two locations if SWH is capable of a direct non-stop flight between them. So to quantify this we calculated the max distance a SWH travelled in a day acc. to our data (we termed it migratory stretch). We also calculated median and mean distance they cover in a day (will be used in next slide).
Over the previous network, we made new links between two nodes if the distance between them is less than the migratory stretch. For distance we didn't use Euclidean. To take into account the spherical nature of the surface, we preferred to use Azimuthal Projection/ Haversine.

```
SWH_migratory_stretch = 676
SWH_median_migration = 165
SWH_mean_migration = 205
```

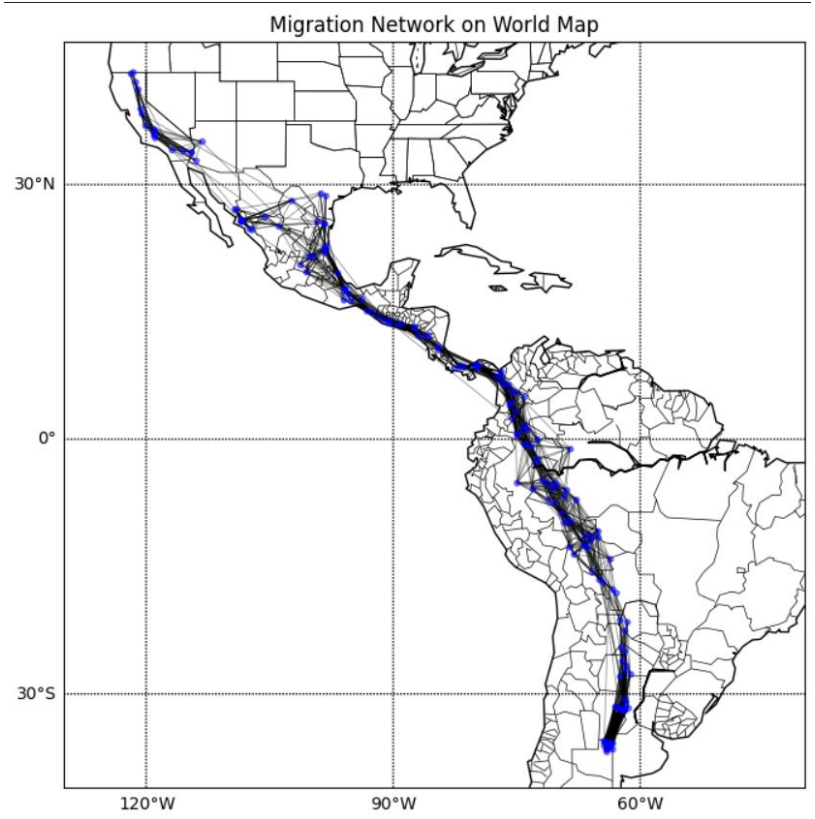


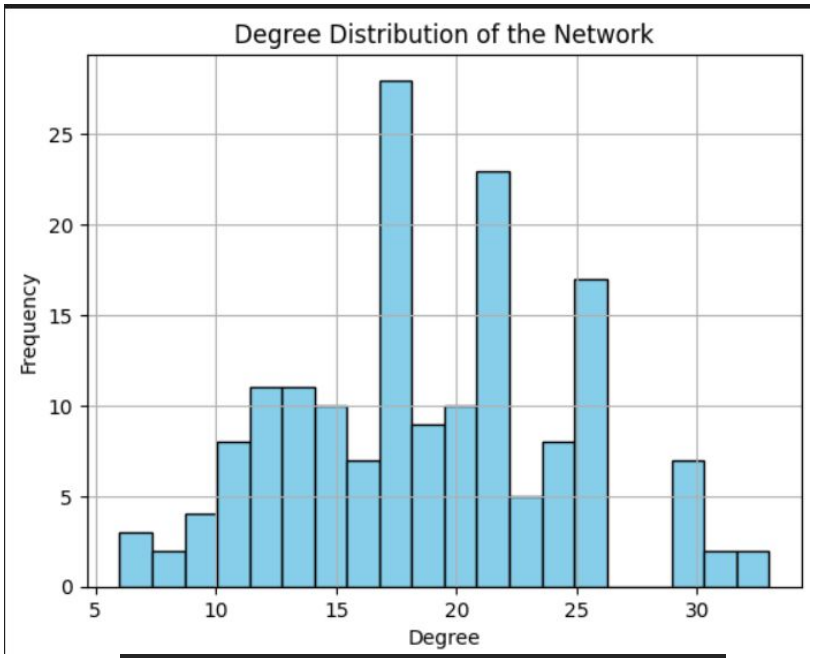
W Wikipedia
Haversine formula - Wik...



esri Esri Community
Distance on a sphere: The Haversine ...

This method of construction gives the following network. The number of edges has nearly grown 10 folds and we get a good degree distribution too.

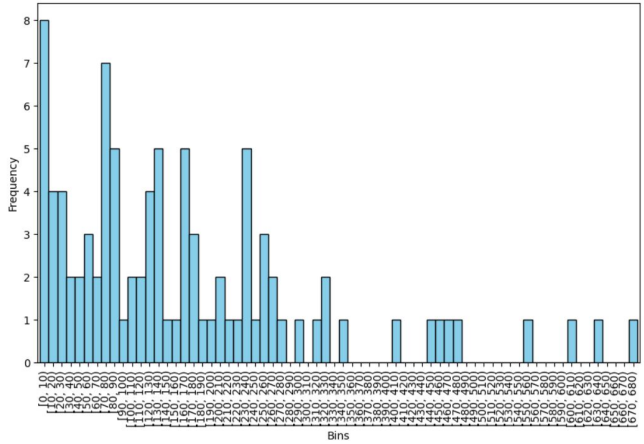





```
print(SWH_network)
```

```
Graph with 167 nodes and 1566 edges
```

A different attempt at network construction



We now have a decent network ready but something is missing. As you can see in the graph on the right, SWH has travelled the length of its migratory stretch but its frequency is way less as compared to smaller distances. It implies that birds don't often to travel large distances when smaller travel is possible. But in the current model, edges with larger distance have same importance as those with smaller ones. What's the fix then? Its making the network weighted!

```
k_SWH = estimate_k_bisection(SWH_median_migration)
print("Estimated value of Dispersal Constant, k:", k_SWH)

n_SWH = df["birdID"].nunique()
print("No. of Swainson's Hawk, n :", n_SWH)
```

```
Estimated value of Dispersal Constant, k: 0.004150390625
No. of Swainson's Hawk, n : 4
```

```
#  p ( d )  is the probability of dispersing at least distance d, and k is the dispersal coefficient
def p(d, k):
    return math.exp(-k * d)
```

```
def weight_of_edge(n, k, d):
    return n * p(d, k)
```

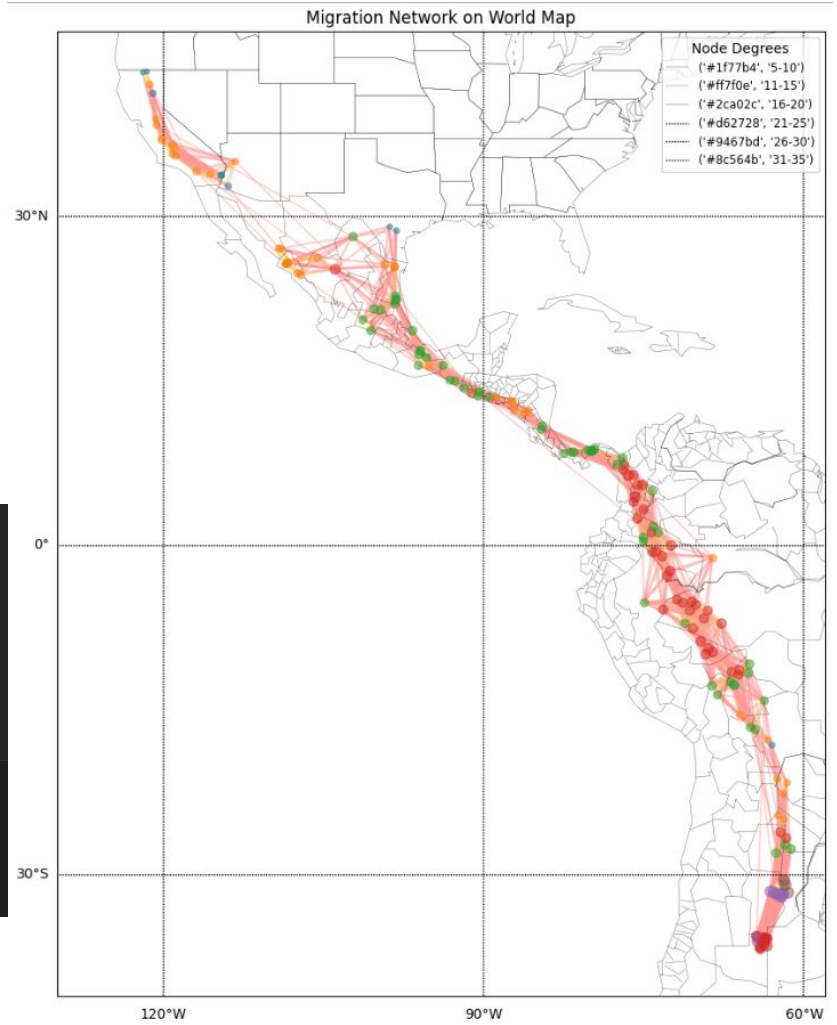
```
print(p(SWH_median_migration, k_SWH))
print(p(SWH_median_migration, 1/SWH_mean_migration))

print(p(SWH_migratory_stretch, k_SWH))
print(p(SWH_migratory_stretch, 1/SWH_mean_migration))

# max dist
print(p(2186, k_SWH))
print(p(2186, 1/SWH_mean_migration))

# min dist
print(p(0.68, k_SWH))
print(p(0.68, 1/SWH_mean_migration))
```

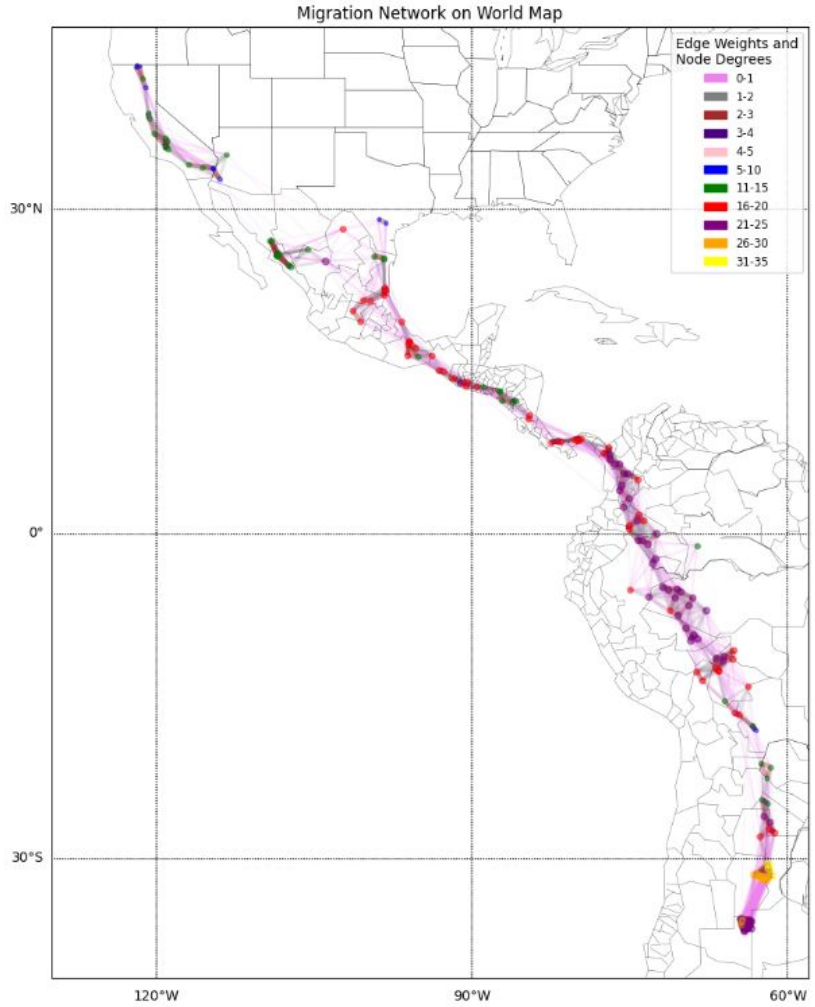
```
0.504183770619497
0.4471424528019328
0.060466604233003825
0.03697323614139773
0.0001147500921183098
2.3385026486803328e-05
0.9971817132226273
0.9966884222385741
```

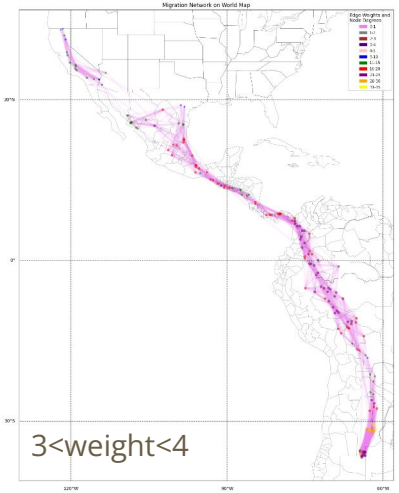
```
SWH_migratory_stretch = 676
SWH_median_migration = 165
SWH_mean_migration = 205
```

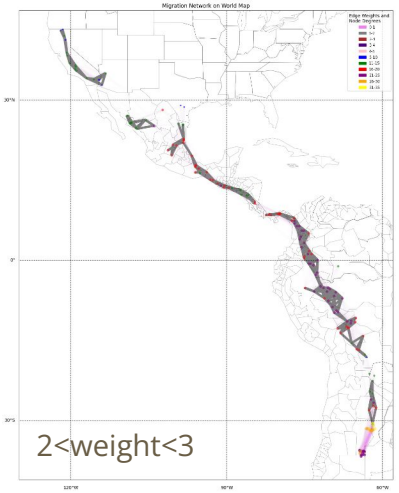
```
print(SWH_network)
```

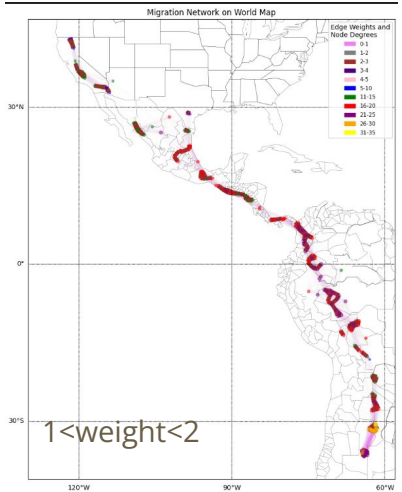
```
Graph with 167 nodes and 1566 edges
```

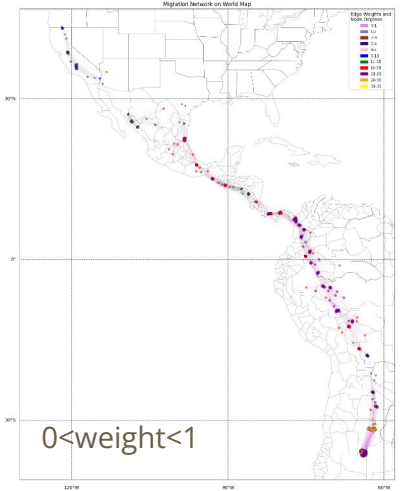



Migration Network on World Map
Edge Weights and Node Degrees
0-1
1-2
2-3
3-4
4-5
5-10
11-15
16-20
21-25
26-30
31-35
30°N
0°
30°S
120°W
90°W
60°W


3<weight<4


2<weight<3


1<weight<2


0<weight<1

Trying different network construction

```
SWH_dist[SWH_dist['time'] < 3][SWH_dist['distance'] < 500]['distance'].max()
```

```
C:\Users\Yashraj Deshmukh\AppData\Local\Temp\ipykernel_4180\4016690045.py:1: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  SWH_dist[SWH_dist['time'] < 3][SWH_dist['distance'] < 500]['distance'].max()

477.0961216935548
```

```
SWH_dist[SWH_dist['distance'] < 478]['distance'].median() , SWH_dist[SWH_dist['distance'] < 478]['distance'].mean()
```

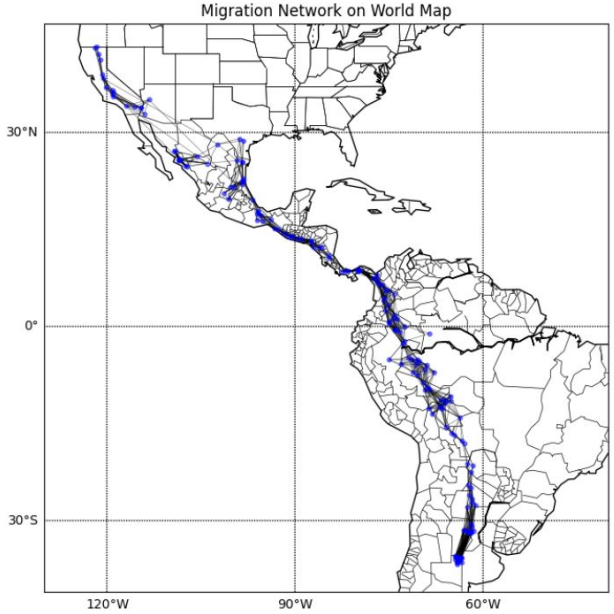
```
(134.635304338803, 162.19031381977132)
```

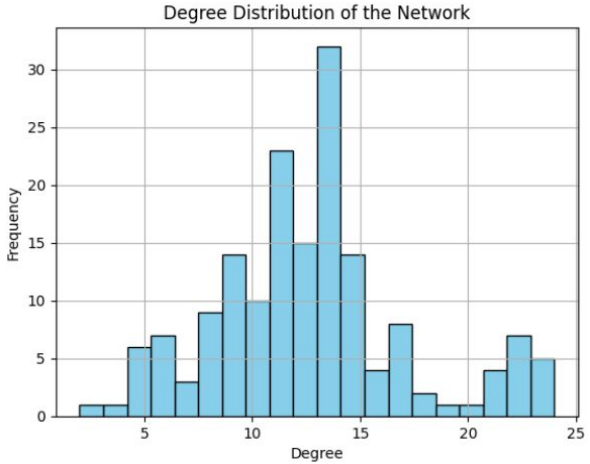
```
SWH_migratory_stretch = 478
SWH_median_migration = 135
SWH_mean_migration = 163
```

```
k_SWH = estimate_k_bisection(SWH_median_migration)
print("Estimated value of Dispersal Constant, k:", k_SWH)

n_SWH = df["birdID"].nunique()
print("No. of Swainson's Hawk, n :", n_SWH)
```

```
Estimated value of Dispersal Constant, k: 0.005126953125
No. of Swainson's Hawk, n : 4
```

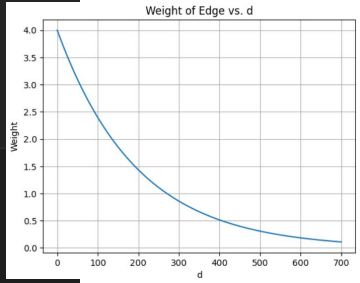
```
Graph with 167 nodes and 1054 edges
```

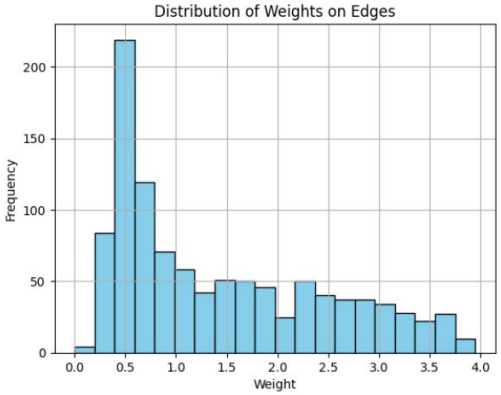




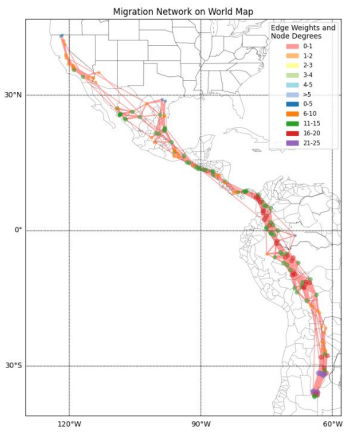

```
print(p(SWH_median_migration, k_SWH))
print(p(SWH_median_migration, 1/SWH_mean_migration))

print(p(SWH_migratory_stretch, k_SWH))
print(p(SWH_migratory_stretch, 1/SWH_mean_migration))

# max dist
print(p(2186, k_SWH))
print(p(2186, 1/SWH_mean_migration))

# min dist
print(p(0.68, k_SWH))
print(p(0.68, 1/SWH_mean_migration))
```

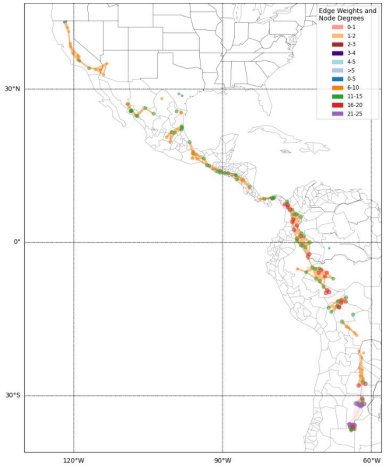
```
0.5005045087004147
0.436825770062579
0.0862346169089532
0.0532628953547274
1.3571758146295913e-05
1.498504503605287e-06
0.996519742060629
0.9958369106413459
```

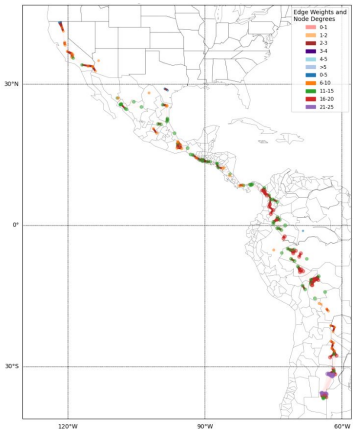


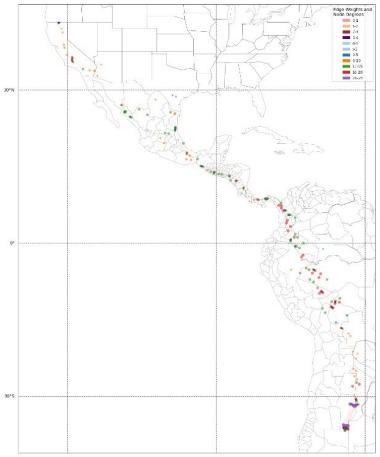


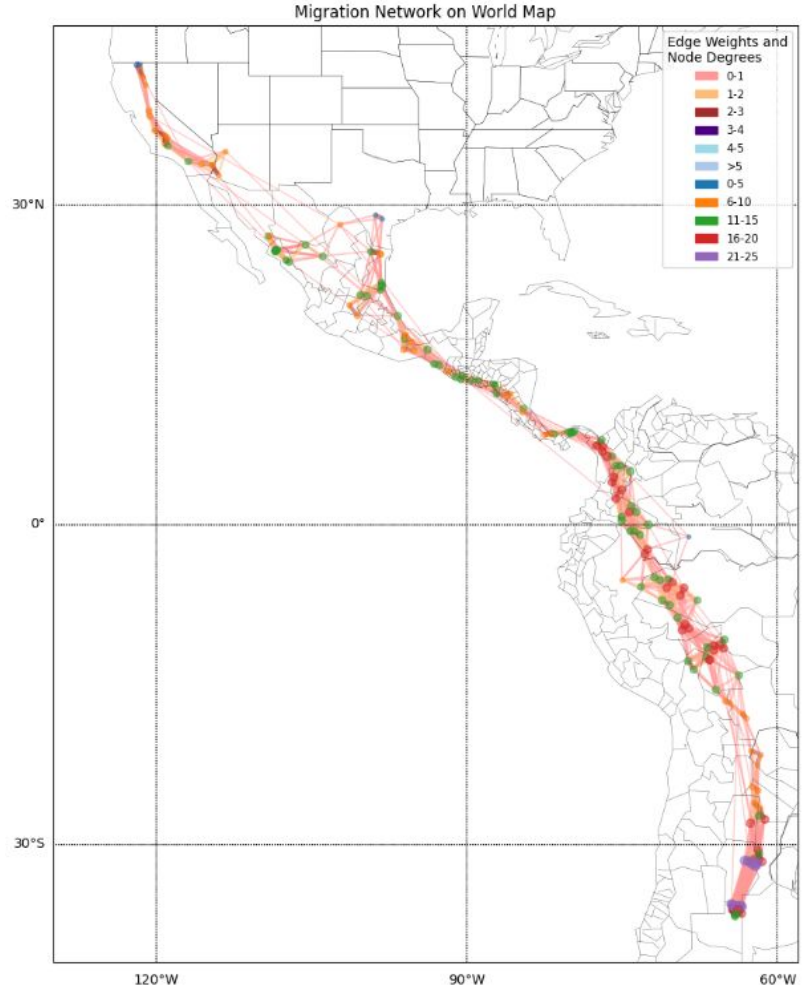
Trying different network construction

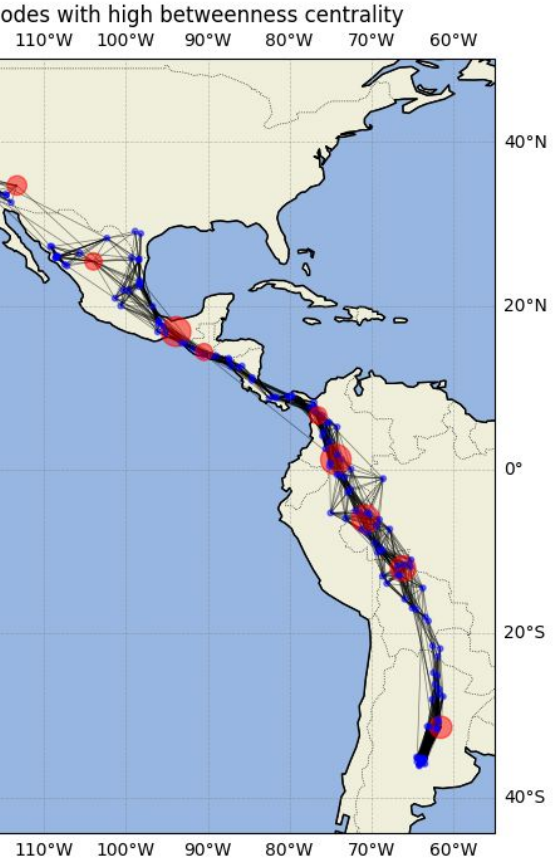






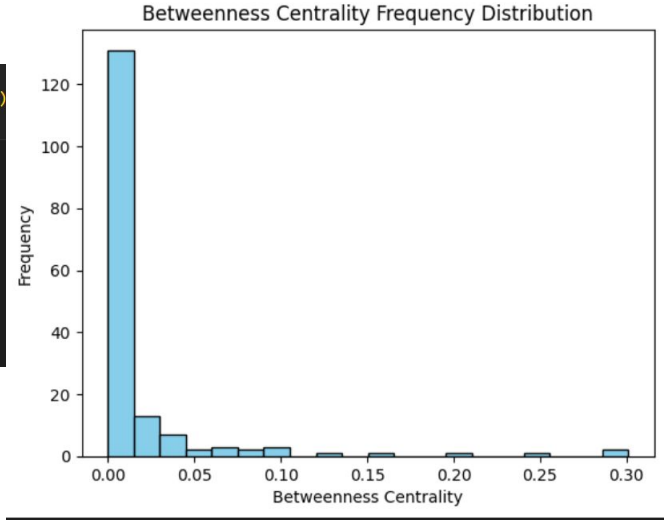




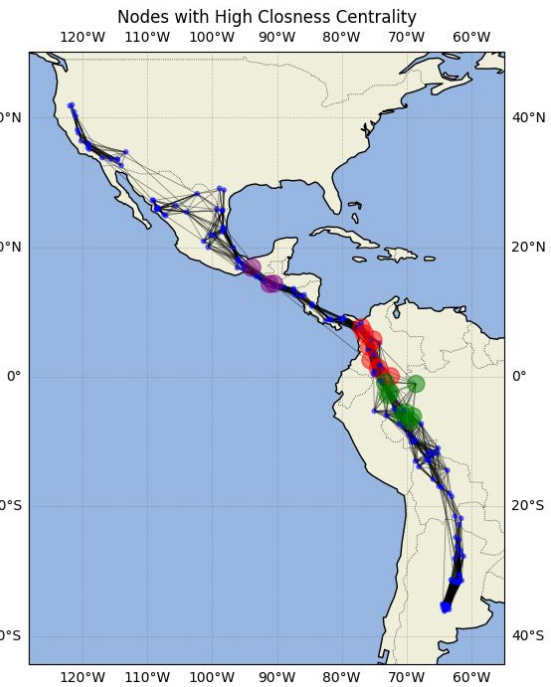



```
    if(centrality>0.09):
        print(f"Important Node {node} with centraility = {centrality}")
```

```
Important Node (-31.451, -61.495) with centraility = 0.1600216453192016
Important Node (-12.056, -66.116) with centraility = 0.20706839409443817
Important Node (-6.011, -70.692) with centraility = 0.2507450834641664
Important Node (1.248, -74.331) with centraility = 0.3008657131377322
Important Node (16.873, -93.871) with centraility = 0.28650218116614684
Important Node (34.712, -113.315) with centraility = 0.12282483756942658
Important Node (6.625, -76.498) with centraility = 0.09667127194256077
Important Node (14.384, -90.45) with centraility = 0.093047341070307 7
Important Node (25.437, -103.972) with centraility = 0.09640643993045585
```



We figured out betweenness centrality of all nodes and noticed choosing 0.09 and greater nodes give us important nodes in dispersed regions. We then highlighted those nodes in hope that we might be able to link it with some other measure on our way.




```
for node, centrality in closeness_centrality.items():
    if centrality > 0.30 :
        print(f"Node {node}: Closeness Centrality = {centrality}")
```

✓ 0.1s

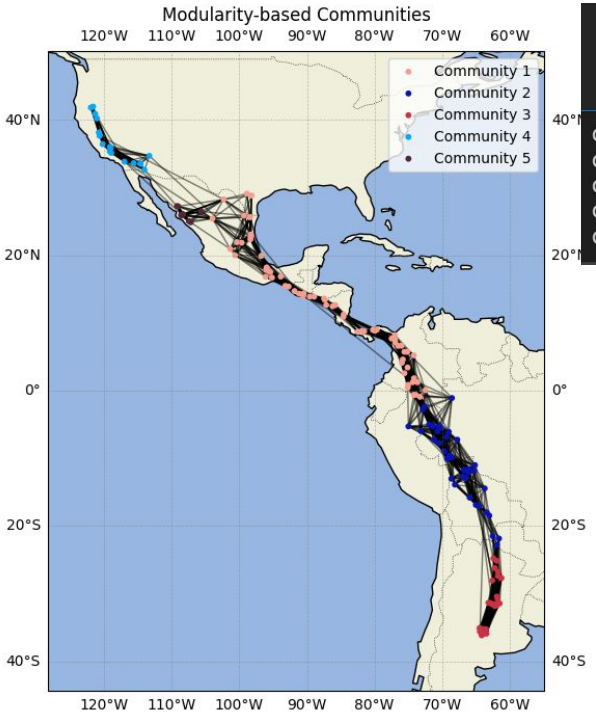
```
Node (4.609, -75.806): Closeness Centrality = 0.30127041742286753
Node (-2.357, -72.574): Closeness Centrality = 0.3143939393939394
Node (-6.011, -70.692): Closeness Centrality = 0.3333333333333333
Node (1.248, -74.331): Closeness Centrality = 0.3516949152542373
Node (16.873, -93.871): Closeness Centrality = 0.3287128712871287
Node (2.576, -75.574): Closeness Centrality = 0.302367941712204
Node (-6.086, -69.01): Closeness Centrality = 0.30127041742286753
Node (0.06, -72.468): Closeness Centrality = 0.3204633204633205
Node (7.621, -76.95): Closeness Centrality = 0.31379962192816635
Node (14.333, -91.134): Closeness Centrality = 0.3029197080291971
Node (-2.784, -72.87): Closeness Centrality = 0.3143939393939394
Node (-5.499, -70.166): Closeness Centrality = 0.30347349177330896
Node (-1.088, -68.548): Closeness Centrality = 0.304029304029304
Node (5.745, -75.149): Closeness Centrality = 0.31379962192816635
Node (-6.827, -69.356): Closeness Centrality = 0.30855018587360594
Node (-0.904, -73.269): Closeness Centrality = 0.33067729083665337
Node (6.625, -76.498): Closeness Centrality = 0.3235867446393762
Node (14.384, -90.45): Closeness Centrality = 0.302367941712204
```

```
point_name, min_dist = find_closest_point(16.873,-93.871)
print("Closest point:", point_name)
print("Minimum distance:", min_dist)
```

✓ 0.6s

```
Closest point: El Ocote
Minimum distance: 26.50468861262013
```

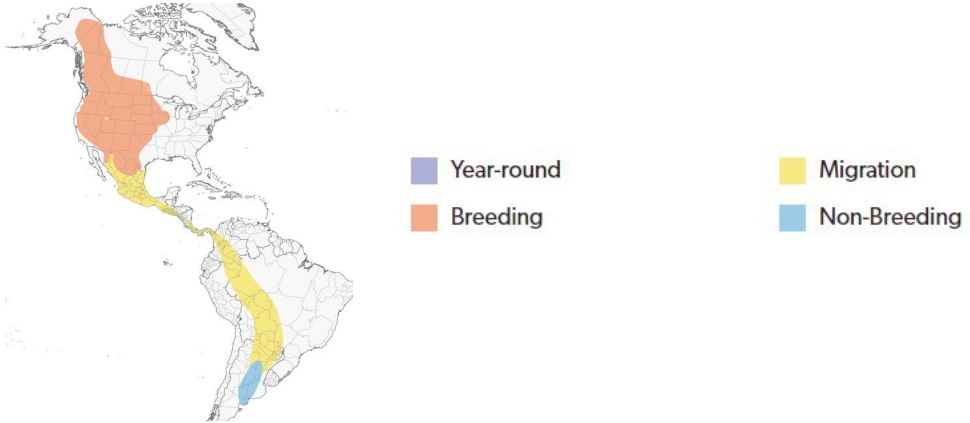
We noticed points with high closeness centrality are in the range -10 to 10 Latitude and some being near 15 in the mexican region. So we colored them differently when highlighting them. First thought was that the node near the equator could be an important point and hence we tested it out in the robustness part. Also the points in the Mexican region are important because of the most narrow landscape in bird's complete network.




```
communities = community.greedy_modularity_communities(SWH_network)
for i, comm in enumerate(communities):
    print(f"Community {i+1}: {comm}")
```

✓ 0.1s

```
Community 1: frozenset({(16.873, -93.871), (21.875, -99.654), (13.895, -88.938), (22.85, -98.235), (4.
Community 2: frozenset({(-9.893, -68.592), (-16.888, -65.005), (-14.448, -63.698), (-1.088, -68.548),
Community 3: frozenset({(-30.818, -61.783), (-31.356, -63.15), (-31.517, -62.602), (-26.773, -61.724),
Community 4: frozenset({(33.649, -115.622), (40.919, -121.315), (36.061, -119.144), (41.85, -121.96),
Community 5: frozenset({(25.884, -108.588), (26.009, -108.582), (25.882, -108.543), (26.073, -108.333)
```



Formed Communities using networkX inbuilt function and noticed an important pattern. The nodes which had high betweenness centrality were nicely distributed along these communities. Also these communities represent different regions of the bird network as represented by the subsequent graph.


```
from scipy.stats import poisson
from scipy.stats import chisquare
import numpy as np

# Example data (replace this with your actual degree distribution data)
observed_degrees = np.array([i for u,i in SWH_network.degree()]) # observed frequencies

# Calculate expected frequencies assuming a Poisson distribution
lambda_param = mean_deg
expected_degrees = poisson.pmf(np.arange(size), lambda_param) * sum

# Perform Chi-square goodness-of-fit test
chi_square_stat, p_value = chisquare(observed_degrees, f_exp=expected_degrees, ddof=1)

# Print results
print("Chi-square statistic:", chi_square_stat)
print("p-value:", p_value)
```

```
Chi-square statistic: 5.57526601991158e+93
p-value: 0.0
```

```
nx.average_clustering(SWH_network)
```

✓ 0.0s

```
0.7664280180088582
```

```
nx.degree_assortativity_coefficient(SWH_network)
```

✓ 0.0s

```
0.6786947465572553
```

```
average_shortest_path = nx.average_shortest_path_length(SWH_network)
print("Average Shortest Path Length:", average_shortest_path)
```

✓ 0.0s

```
Average Shortest Path Length: 4.058365197316211
```

Found out about Ecological Corridors - Regions connecting different wildlife populations separated by human activities/buildings.
Also calculated other metrics which just proved the high dense nature of the graph.
Conducted a p-value test on the degree-distribution to check whether it follows a Poisson Distribution but it doesn’t.

Network Reconstruction

Cost Function
Trail Reconstruction Algorithm
Vulnerable Nodes

Node Locator

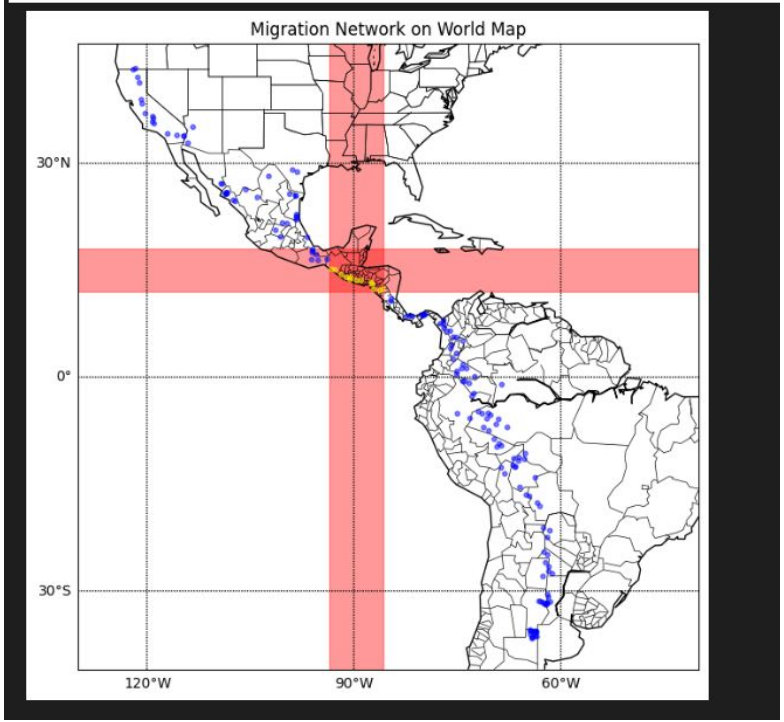
```
Vulnerable nodes = 13: [(15.395, -92.714), (13.9, -89.592), (13.432, -87.335), (12.657, -85.811),
```

vulnerable[0] 12.30

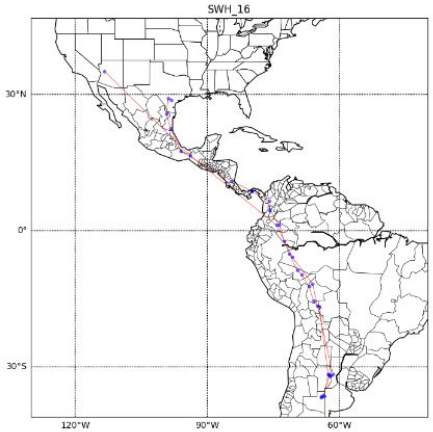
vulnerable[1] 18.40

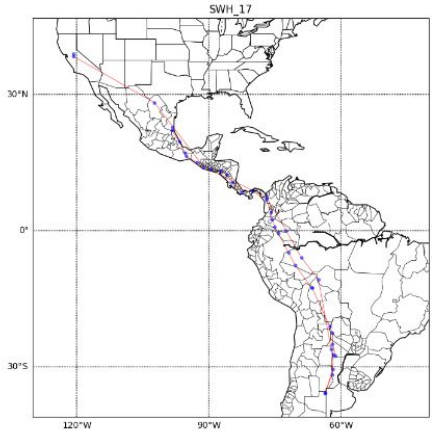
vulnerable[2] −85.70

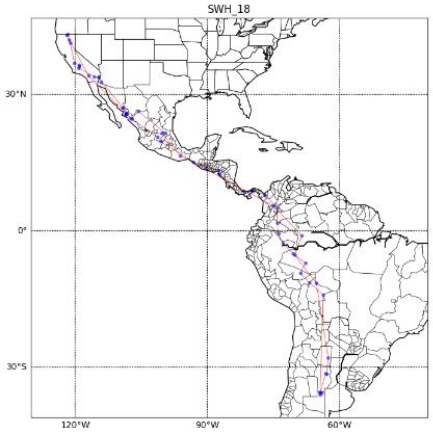
vulnerable[3] −93.60

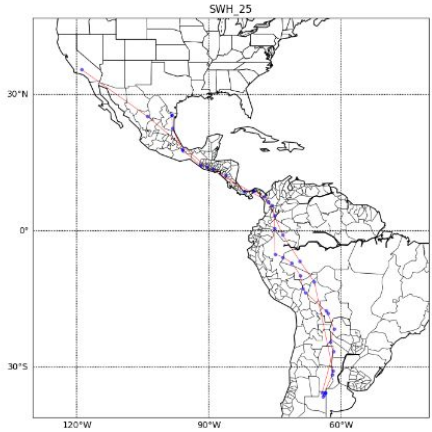


Bird Trails









Shortest Trail Traversal

Algorithm

1) Brute Force - fast for trail cost computation, exponentially expensive for entire network, hence not suitable
2) Optimised Algorithm - Time Complexity: O(V*E)
   Computes least cost and returns path of travel by backpropagation through predecessors

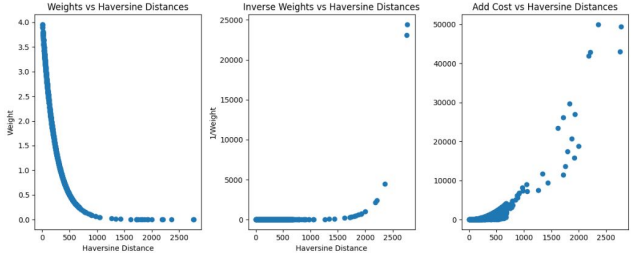
Cost Function

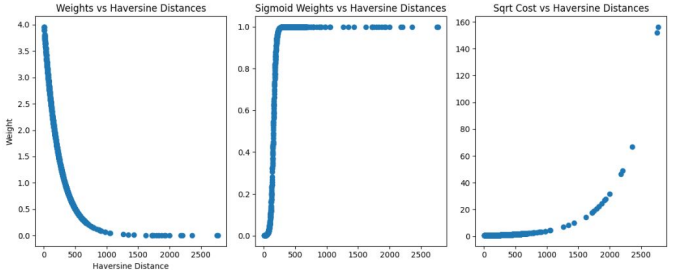
```
print("Maximum haversine distance:", max_distance)
print("Minimum weight:", min_weight)
print("Max Cost:", 1/min_weight)
```

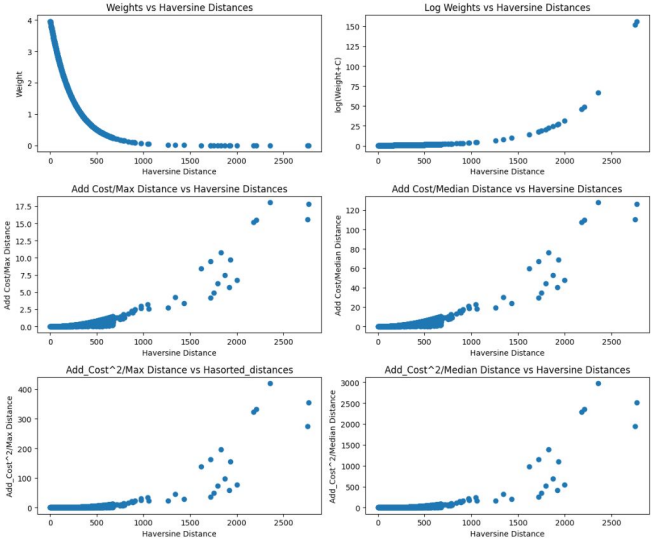
```
Maximum haversine distance: 2768.556291504218
Minimum weight: 4.09034676077418886e-05
Max Cost: 24447.805002496363
```

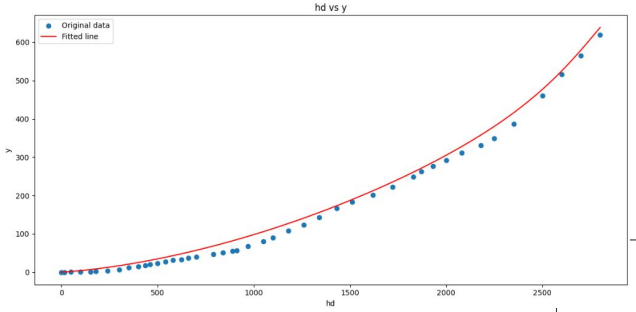
Limitations of simple cost function

Cost Function - Experimenting with cost function



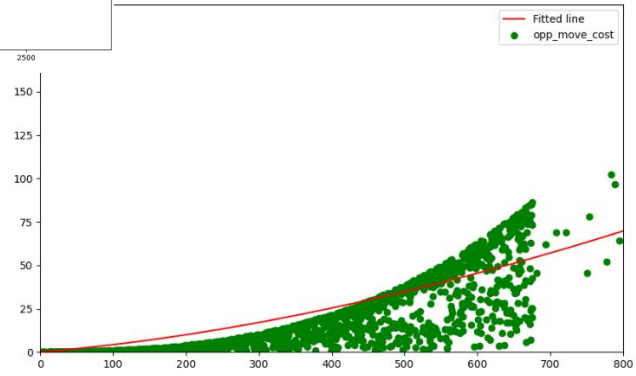







```
Coefficients:  [ 5.28015584e-03  5.32880556e-02 -5.35349889e-08  4.95687954e-05
  1.23519628e+00]
Bias term:  -4.80810452156922
```

```python
# Create a range of x values
x_values = np.linspace(min(hd), max(hd), num=200)

# Create the corresponding y values
b_values = weight_of_edge(n_SWH, k_SWH, x_values)
a_values = 1 / b_values
e_values = a_values ** 2

X_values = np.column_stack((a_values, x_values, e_values, x_values**2, b_values))
y_values = model.predict(X_values)
```



Final Cost Function

```
print("n_SWH:", n_SWH)
print("k_SWH:", k_SWH)
print("median_distance:", median_distance)
```

```
n_SWH: 4
k_SWH: 0.004150390625
median_distance: 390.7105977636296
```

```
def cost_of_edge_(hd):
    b = weight_of_edge(n_SWH, k_SWH, hd)
    a = 1 / b
    e = a ** 2
    return model.predict(np.column_stack((a,hd,e,hd**2,b)))

def cost_of_edge(hd,b):
    a = 1 / b
    e = a ** 2
    return model.predict(np.column_stack((a,hd,e,hd**2,b)))

def cost_of_opposite_movement(lat1,lat2,d):
    if lat1>lat2:
        return 0
    else:
        return d*(abs(lat1-lat2)+1)**2 / median_distance
```

COST = cost_edge(weight,distance) + cost_opp_motion(lat1,lat2,distance) + func(path_length)

discourage choosing longer routes

Cost function derived using ridge regression

Utilises weights of edge to output cost of choosing that route. Distance is used to smoothen the resultant function output.

(for exact weights and parameters refer to the presentation slides or github repo of the project)

penalise for motion against the wind stream

$$= \begin{cases} \dfrac{\text{distance} * ((\text{lat1}-\text{lat2})+1)^2}{\text{median\_distance}} & \text{; for lat1} > \text{lat2} \\ 0 & \text{; otherwise} \end{cases}$$

discourage longer hops

$$f(n) = \begin{cases} n*\log(n+1) & \text{; if node is the end node} \\ 0 & \text{; otherwise} \end{cases}$$

for paths with nearly equat costs, assists in choosing the one with smaller path length

For algorithm refer to reconstruction.ipynb in our repo

Influence of temperature on migration network

Temperature as a Migration Compass: Swainson Hawk follows a preferential migration path advancing towards higher temperatures.

Thermally Guided Journeys: Over 80% of the migration paths were green edges, highlighting its preference for warmer regions during their extensive migrations.

Climate-Influenced Routes: This predominance of green edges suggests potential vulnerability to global climate changes.

Statistical Analysis of temperature dependency

We tracked the movement of Swainson Hawk, and noted the temperature of that place on that day.

We compared the temperatures and if the new temperature is higher than its previous temperature (with 1^0C tolerance), and then marked that edge to be green , else red.

We aim to check if there is any correlation between the bird migration route and temperature of points on that route.

Temperature Dependence Plots

We see a very heavy temperature biased bias in the route.

In its southward path, the bird travels 82.53% towards higher temperatures.

In its northward journey, the statistical figure is 83.13%.