AIS Analyze Vessel Clusters and EDA

EDA, Stats, K-Mean Cluster, Plots for a given Vessel based on an input MMSI Id

In [1]:
```
# from IPython.display import Image, HTML
import os
import numpy as np
import math
import pandas as pd
import datetime
from glob import glob
import geopy.distance
import folium
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns; sns.set()

import warnings
warnings.filterwarnings("ignore")           # Suppress Warning
```

Global Variables

In [3]:
```
# s3://vault-data-corpus/vessel data/ConsolidatedAIS/
WorkingFolder = "vessel data/ConsolidatedAIS/"
OutputDir = WorkingFolder

PROC_YEAR = '2017'
MAX_CLUSTER = 5
```

Common Functions

In [4]:
```
def Calc_Centroid_Radius(center_latlon, lat_max, lat_min, lon_max, lon_min):
    dist = list()

    # Center distance to the outer most coordinates
    dist.append(geopy.distance.distance((center_latlon[0], center_latlon[1]), (lat_max, lon_max)).mi)
    dist.append(geopy.distance.distance((center_latlon[0], center_latlon[1]), (lat_max, lon_min)).mi)
    dist.append(geopy.distance.distance((center_latlon[0], center_latlon[1]), (lat_min, lon_max)).mi)
    dist.append(geopy.distance.distance((center_latlon[0], center_latlon[1]), (lat_min, lon_min)).mi)

    print(dist)

    return round(max(dist), 2)

# Calc_Centroid_Radius(centers[0], stat.lat[1], stat.lat[2], stat.lon[1], stat.lon[2])
```

Load Broadcast Data

In [5]:
```
Broadcast = pd.read_csv(WorkingFolder + "Broadcast_{}.csv".format(PROC_YEAR), sep=",", parse_dates=['date_time'])
Broadcast.head()
```

Out[5]:

	mmsi_id	date_time	lat	lon	speed_over_ground	course_over_ground	voyage_id	heading	status
0	366940480	2017-01-04 11:39:36	52.48730	-174.02316	10.0	-140.7	NaN	267.0	undefined
1	366940480	2017-01-04 11:40:45	52.48718	-174.02835	10.0	-141.6	NaN	266.0	undefined
2	366940480	2017-01-04 11:42:26	52.48705	-174.03608	10.0	-142.3	NaN	267.0	undefined
3	366940480	2017-01-04 13:51:07	52.41575	-174.60041	9.1	-154.0	NaN	251.0	undefined
4	366940480	2017-01-04 13:55:17	52.41311	-174.61718	9.1	-157.3	NaN	251.0	undefined

In [6]:
```
print("Raw Count:", Broadcast.shape[0])
```

```
Raw Count: 3125152
```

In [7]:
```
# Assign 0 to blank voyage id
Broadcast['voyage_id'] = Broadcast['voyage_id'].fillna(0)
Broadcast = Broadcast.astype({"voyage_id": int})           # cast type to int
```

EDA and Stats

In [ ]:
```
# Broadcast_Zone03_2017_01
# ========================
# 367094420      ARCTIC MARINER      type: 1001 - Fishing, 31x9.28       lots of points   multiple trips, same rou
# 367565680      ADAMANT             type: 1001 - Fishing, 18x8          lots of points   ?
# 367373760      NORTH SEA           type: 1001 - Fishing, 37x10                         ?
# 440102000      TORAH               type: 1004 - Freight, 115.8x16.62   lots of points   weird
# 636014222      Zim Rotter          type: 1004 - Freight, 349x45.73                     Going some where
# 367322830      GYRFALCON           type: 1025 - Towing, 30.63x10.4     lots of points   weird behavior

# Broadcast_Zone10_2014_01
# ========================
# 316500126                          type: 31 - TugTow,     15x5        irregular pattern towing
# 367528210                          type: 31 - TugTow,     33x13       irregular pattern towing
# 316881510                          type: 52 - TugTow,     28x13       irregular pattern towing
# 366025993                          type: 52 - TugTow,     21x7        pattern?
# 538001471                          type: 70 - Cargo,      302x44      path pattern
# 538284070                          type: 70 - Cargo,      184x31      path pattern
# 229560200                          type: 70 - Cargo,      350x42      path pattern
# 235733603                          type: 80 - Tanker,     183x32      path pattern
# 538007477                          type: 0 - Tanker?,     170x27      Going some where
# 366089092                          type: 0, 30x6         Infrequent ping per trip

mmsi = 271041862
voyage_id = 0
```

In [8]:
```
# Select a specific Vessel
df = Broadcast.loc[Broadcast.mmsi_id==mmsi, ['date_time', 'lat', 'lon', 'speed_over_ground', 'voyage_id']]  # & (B
df.rename(columns={'lat':'latitude', 'lon':'longitude'}, inplace=True)

# Extract Date and Hour
df['PingDate'] = df['date_time'].dt.date
df['PingHour'] = df['date_time'].dt.hour

print("Rows:", df.shape)
df.head()
```

```
Rows: (748, 7)
```

Out[8]:

	date_time	latitude	longitude	speed_over_ground	voyage_id	PingDate	PingHour
9439	2017-01-21 04:00:17	54.66813	-174.01128	10.4	0	2017-01-21	4
116117	2017-01-20 06:55:47	54.54627	-168.00870	9.2	0	2017-01-20	6
116118	2017-01-20 06:54:26	54.54616	-168.00310	9.1	0	2017-01-20	6
116119	2017-01-20 06:58:46	54.54654	-168.02125	8.6	0	2017-01-20	6
116120	2017-01-20 07:06:07	54.54723	-168.05249	9.0	0	2017-01-20	7

In [9]:
```
# # Fudge some anomaly
# df.loc[df.date_time=='2017-01-10 01:12:31', 'latitude'] = 70.25
# df.loc[df.date_time=='2014-01-01 00:03:06', 'longitude'] = -100.123
# df.head()
```

In [10]:
```
# Get Unique voyage Id
df.voyage_id.unique()
```

Out[10]:
```
array([0])
```

In [11]:
```
# # Select 1 voyage
# df = df.loc[df.voyage_id==1]
# df.shape
```

In [12]:
```
# Calculate Statistics
# df.describe()
df.agg(['count', 'max', 'min', 'mean', 'median', 'std', 'nunique'])
```

Out[12]:

	date_time	latitude	longitude	speed_over_ground	voyage_id	PingDate	PingHour
count	748	748.000000	748.000000	748.000000	748.0	748	748.000000
max	2017-01-21 04:00:17	54.668130	-162.665270	13.600000	0.0	2017-01-21	23.000000
min	2017-01-19 10:29:59	54.133770	-174.011280	5.200000	0.0	2017-01-19	0.000000
mean	2017-01-20 00:03:39.533422592	54.440042	-166.332504	8.566578	0.0	NaN	11.528075
nunique	748	714.000000	748.000000	75.000000	1.0	3	19.000000
median	NaN	54.428390	-166.233505	8.700000	0.0	NaN	9.000000
std	NaN	0.071693	1.074679	1.697823	0.0	NaN	8.520894

Sampling Data Hourly

In [13]:
```
mygroup = df.groupby(['voyage_id', 'PingDate', 'PingHour'])
sampling = mygroup['latitude', 'longitude', 'speed_over_ground'].mean()
sampling['PingCount'] = mygroup['latitude'].count()
sampling.reset_index(inplace=True)

print("Rows:", sampling.shape[0])
sampling.head()
```

```
Rows: 20
```

Out[13]:

	voyage_id	PingDate	PingHour	latitude	longitude	speed_over_ground	PingCount
0	0	2017-01-19	10	54.133770	-162.665270	12.700000	1
1	0	2017-01-19	15	54.270693	-164.152723	10.900000	3
2	0	2017-01-19	16	54.295319	-164.515008	11.545455	33
3	0	2017-01-19	17	54.336627	-164.896077	13.050000	36
4	0	2017-01-19	18	54.360704	-165.147314	7.800000	49

In [14]:
```
# Formulate Popup text for the Map
sampling['Text'] = sampling.apply(lambda x: "{}:{}-V:{} - SOG: {} - Count: {} ({}, {})".format(x.PingDate, x.PingH
sampling.head()
```

Out[14]:

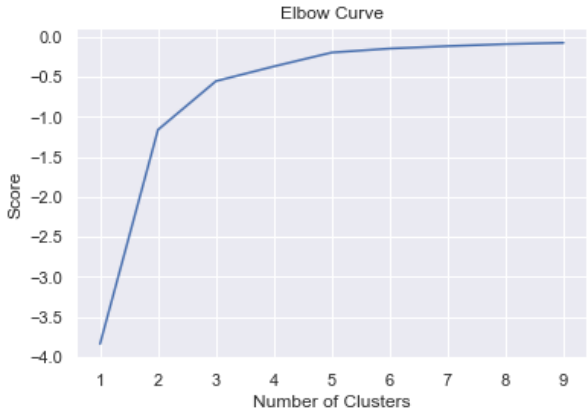
	voyage_id	PingDate	PingHour	latitude	longitude	speed_over_ground	PingCount	Text
0	0	2017-01-19	10	54.133770	-162.665270	12.700000	1	2017-01-19:10-V:0 - SOG: 12.7 - Count: 1 (54.1...
1	0	2017-01-19	15	54.270693	-164.152723	10.900000	3	2017-01-19:15-V:0 - SOG: 10.9 - Count: 3 (54.2...
2	0	2017-01-19	16	54.295319	-164.515008	11.545455	33	2017-01-19:16-V:0 - SOG: 11.5 - Count: 33 (54....
3	0	2017-01-19	17	54.336627	-164.896077	13.050000	36	2017-01-19:17-V:0 - SOG: 13.0 - Count: 36 (54....
4	0	2017-01-19	18	54.360704	-165.147314	7.800000	49	2017-01-19:18-V:0 - SOG: 7.8 - Count: 49 (54.3...

In [15]:
```
# Output Sampling Data
# sampling.to_csv(OutputDir + 'sampling_366089092.csv', index=None, header = True)
```

Clustering via K-means

- https://github.com/JosephMagiya/Clustering-GPS-Co-ordinates--Forming-Regions./blob/master/Clustering-GPS-Co-ordinates--Forming-Regions.ipynb (https://github.com/JosephMagiya/Clustering-GPS-Co-ordinates--Forming-Regions./blob/master/Clustering-GPS-Co-ordinates--Forming-Regions.ipynb)

In [16]:
```
K_clusters = range(1,10)
kmeans = [KMeans(n_clusters=i) for i in K_clusters]

Y_axis = df[['latitude']]
X_axis = df[['longitude']]

score = [kmeans[i].fit(Y_axis).score(Y_axis) for i in range(len(kmeans))]

# Plot labels
plt.plot(K_clusters, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```



In [17]:
```
# Cluster all the coordinances
X = df[['latitude', 'longitude']]

kmeans = KMeans(n_clusters=MAX_CLUSTER, init='k-means++')

kmeans.fit(X)                                   # Compute k-means clustering. # Compute k-means clustering.
df['cluster_label'] = kmeans.predict(X)         # Labels of each point
centers = kmeans.cluster_centers_               # Coordinates of cluster centers.

df.head(3)
```

Out[17]:

	date_time	latitude	longitude	speed_over_ground	voyage_id	PingDate	PingHour	cluster_label
9439	2017-01-21 04:00:17	54.66813	-174.01128	10.4	0	2017-01-21	4	4
116117	2017-01-20 06:55:47	54.54627	-168.00870	9.2	0	2017-01-20	6	1
116118	2017-01-20 06:54:26	54.54616	-168.00310	9.1	0	2017-01-20	6	1

In [18]:
```
centers
```

Out[18]:
```
array([[  54.34105019, -164.96484228],
       [  54.53354795, -167.77134319],
       [  54.41170451, -165.83678481],
       [  54.47421758, -166.78022547],
       [  54.66813   , -174.01128   ]])
```

In [19]:
```
# Distribution Summary for all clusters - Cluster, ping counts
PingCluster = df.groupby('cluster_label')['date_time'].count().reset_index()
PingCluster.rename(columns={'date_time':'PingCount'}, inplace=True)
PingCluster.sort_values('PingCount')
```

Out[19]:

	cluster_label	PingCount
4	4	1
0	0	158
1	1	166
3	3	191
2	2	232

In [20]:
```
# Locate and agg data for a selected cluster
Select_Cluster = 0

stat = df.loc[df.cluster_label==Select_Cluster].agg(['count', 'max', 'min'])
stat
```

Out[20]:

	date_time	latitude	longitude	speed_over_ground	voyage_id	PingDate	PingHour	cluster_label
count	158	158.00000	158.00000	158.0	158	158	158	158
max	2017-01-19 19:44:29	54.37692	-162.66527	13.6	0	2017-01-19	19	0
min	2017-01-19 10:29:59	54.13377	-165.40066	5.6	0	2017-01-19	10	0

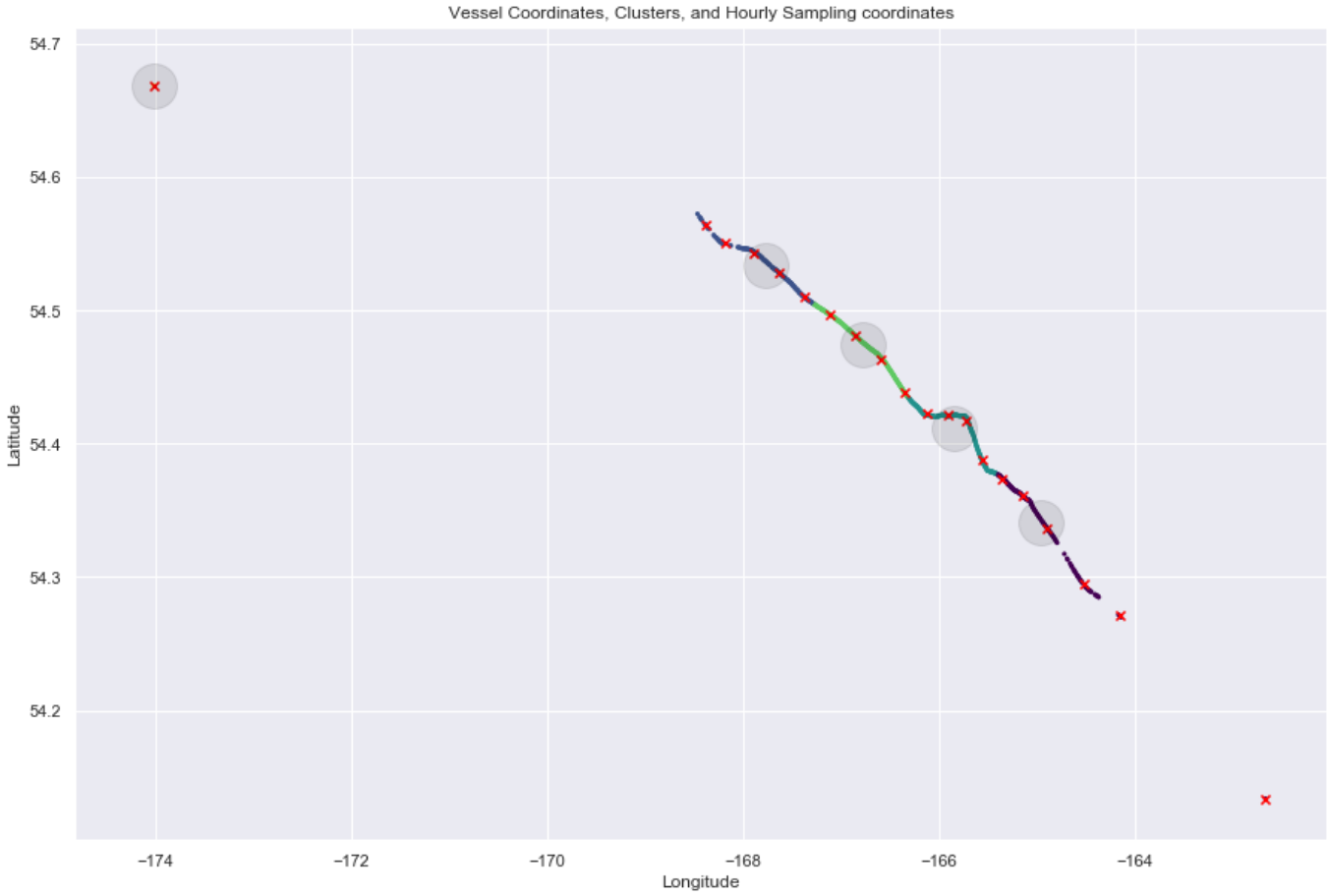
In [21]:
```
# Get Cluster Radius
Calc_Centroid_Radius(centers[Select_Cluster], stat.latitude[1], stat.latitude[2], stat.longitude[1], stat.longitude
```

```
[92.92141262321941, 17.779006758123256, 94.2586530053958, 22.74423485665943]
```

Out[21]:
```
94.26
```

Plot Vessel Coordinates, Clusters, and Hourly Sampling coordinates

In [22]:
```
# various scatter plots
fig = plt.figure(figsize=(15,10))
ax = fig.add_subplot()

# All coordinates
ax.scatter(df.longitude, df.latitude, s=10, lw=0, c=df['cluster_label'], cmap='viridis', alpha=1)
# Hourly Sampling coordinates
ax.scatter(sampling.longitude, sampling.latitude, marker='x', color='red')
# Clusters centroids
ax.scatter(centers[:, 1], centers[:, 0], c='black', s=900, alpha=0.1)

# Plot Labels
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Vessel Coordinates, Clusters, and Hourly Sampling coordinates')
plt.show()
```

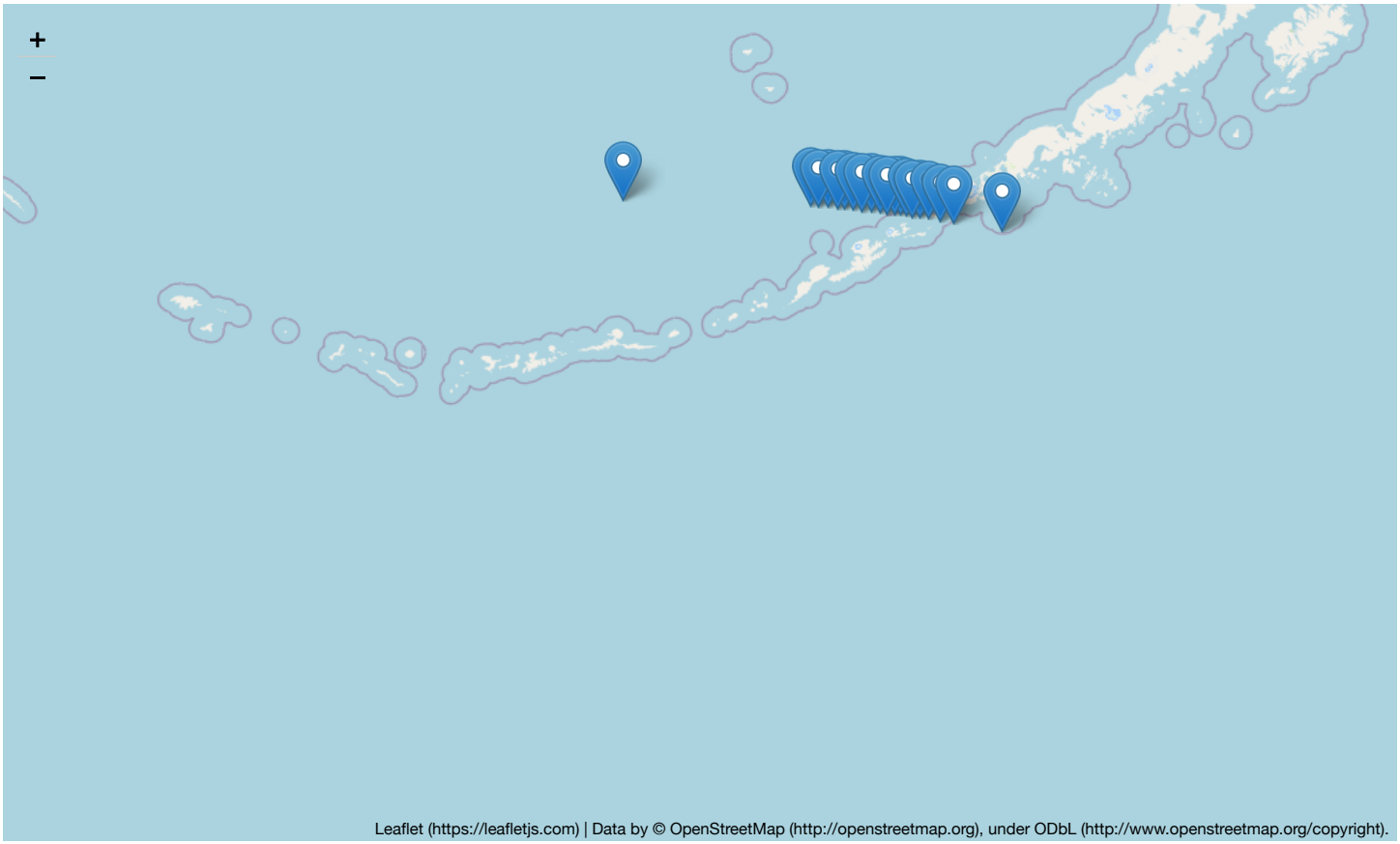


Real Map of Ping Coordinates and Popups

- https://github.com/python-visualization/folium (https://github.com/python-visualization/folium)
- https://github.com/collinreinking/longitude_latitude_dot_plots_in_python_with_folium/blob/master/MapsTutorials.ipynb (https://github.com/collinreinking/longitude_latitude_dot_plots_in_python_with_folium/blob/master/MapsTutorials.ipynb)
- https://georgetsilva.github.io/posts/mapping-points-with-folium/ (https://georgetsilva.github.io/posts/mapping-points-with-folium/)

In [23]:
```
# Plot just the Hourly Sampling Pings Coordinates
X = sampling

MapCoordinates = X[['latitude', 'longitude']].values
PopupText = X.Text.values

mapit = folium.Map(location=MapCoordinates[0], zoom_start=8)

for index in range(0, len(MapCoordinates)):
    folium.Marker(MapCoordinates[index], popup=PopupText[index]).add_to(mapit)  #, popup=PopupText[index]

# mapit.save( 'map.html')
mapit
```

Out[23]:



In [24]:
```
# # ********* Plot all coordinates; Very slow might need filtering to reduce points
# X = df  #.head()

# MapCoordinates = X[['latitude', 'longitude']].values
# mapit = folium.Map(location=MapCoordinates[0], zoom_start=8)

# for index in range(0, len(MapCoordinates)):
#     folium.Marker(MapCoordinates[index]).add_to(mapit)  #, popup=PopupText[index]

# # mapit.save( 'map.html')
# mapit
```