```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed May 15 09:41:40 2019

@author: torjus



THIS PYTHON CODE IS USED FOR THE NODE SELECTION IN THE TRADE FLOW ANALYSIS


"""

import sqlite3
import matplotlib.pyplot as plt 
import time
import datetime
from mpl_toolkits.basemap import Basemap
import numpy as np
from scipy import stats
import pandas as pd
import sklearn.cluster as cluster 
import scipy.cluster.hierarchy as hcluster 
import loc_check as LC
import networkx as nx
from itertools import cycle , groupby 
from pylab import boxplot
import matplotlib.colors as mcolors
import unixTimeConvert as UTC
from mpl_toolkits.axes_grid1.inset_locator import inset_axes
from datetime import timedelta
from shapely.geometry import MultiPoint
from math import radians, cos, sin, asin, sqrt
import AIS_Analysis 
import AIS_Plotting as PAIS
# Import NodeList
nodelist = pd.read_excel('nodelist.xlsx',sheet_name='Initial')


liq = nodelist[nodelist['type']==2]
reg = nodelist[nodelist['type']==1]

#Initial number of Nodes
init_n_nodes = len(liq)+len(reg)

#Sort by latitude as primary and longitude as secondary
liq.sort_values(by = ['latitude','longitude'],inplace=True)
reg.sort_values(by = ['latitude','longitude'],inplace=True)


#Drop duplicate positions
liq.drop_duplicates(subset=['latitude','longitude'],inplace=True)
reg.drop_duplicates(subset=['latitude','longitude'],inplace=True)

#reset index
liq.reset_index(inplace=True,drop = True)
reg.reset_index(inplace=True,drop = True)



#Drop values close to each other
liq_out = pd.DataFrame(columns=liq.columns)
reg_out = pd.DataFrame(columns=reg.columns)

km =10 #minimum distance between nodes
liq_out = liq_out.append(liq.iloc[0],ignore_index=True)
reg_out = reg_out.append(reg.iloc[0])
liq_drop = pd.DataFrame(columns=liq.columns)
reg_drop = pd.DataFrame(columns=reg.columns)
for row in range(0,len(liq)-1):
    row_n = row+1
```

```
    if AIS_Analysis.haversine(liq['longitude'][row],liq['latitude'][row],liq['longitude'][row_n],liq['lat
        liq_out = liq_out.append(liq.iloc[row_n],ignore_index=True)
    else:
        liq_drop = liq_drop.append(liq.iloc[row_n],ignore_index=True)
```

```
for row in range(0,len(reg)-1):
    row_n = row+1
    if AIS_Analysis.haversine(reg['longitude'][row],reg['latitude'][row],reg['longitude'][row_n],reg['lat
        reg_out = reg_out.append(reg.iloc[row_n],ignore_index=True)
    else:
        reg_drop = reg_drop.append(reg.iloc[row_n],ignore_index=True)
#Number of Nodes after all close are removed
n_nodes_second = len(liq_out)+len(reg_out)
```

```
# Plot liquefaction in red
Pos = [180,90,-180,-90]
m = PAIS.CreateMap(Pos)
x1, y1 = m(liq_out['longitude'],liq_out['latitude'])
m.scatter(x1,y1,30,marker='o',c='red',zorder=4)
x2, y2 = m(reg_out['longitude'],reg_out['latitude'])
m.scatter(x2,y2,15,marker='o',c='blue',zorder=5)
```

```
#node out
node_out = pd.concat([liq_out,reg_out],ignore_index=True)
node_out.sort_values(['country'],inplace=True)
node_out.reset_index(inplace=True,drop = True)
node_out['node']=node_out.index
node_out['radius'] = 10
node_out.to_excel('nodelist_merged1.xlsx',sheet_name='test')
```