```
#####################################################################
# Finds average trucking costs (cents/tonne*km) between each sub-Saharan capital
#####################################################################

import csv
from math import sqrt
from sys import exit
import pickle
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.mlab as mlab
import os
from scipy.stats import norm
import matplotlib as mpl
import matplotlib.cm as cm
import matplotlib.colors as colors
import shapefile
from math import sin, cos, sqrt, atan2, radians, pi, degrees
from scipy import ndimage
from matplotlib.font_manager import FontProperties
import cartopy.crs as ccrs
import cartopy.io.shapereader as shpreader
import itertools

try:
        wddata='/Users/lilllianpetersen/iiasa/data/supply_chain/'
        wdfigs='/Users/lilllianpetersen/iiasa/figs/supply_chain/'
        wdvars='/Users/lilllianpetersen/iiasa/saved_vars/supply_chain/'
        f=open(wddata+'trading_across_borders2017.csv','r')
except:
        wddata='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/data/'
        wdfigs='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/figs/'
        wdvars='C:/Users/garyk/Documents/code/riskAssessmentFromPovertyEstimations/supply_chain/vars/'


################################
# Load variables
################################
subsaharancountry = np.load(wdvars+'subsaharancountry.npy')
subsaharancountry[subsaharancountry=='Congo']='Congo (DRC)'
subsaharancountry[subsaharancountry=='Congo (Republic of the)']='Congo'

countrycosted=np.load(wdvars+'countrycosted.npy')
countrycosted[countrycosted=='Congo']='Congo (DRC)'
countrycosted[countrycosted=='Congo (Republic of the)']='Congo'

capitalcosted=np.load(wdvars+'capitalcosted.npy')
subsaharancapital=np.load(wdvars+'subsaharancapital.npy')

factoryLatLon = np.load(wdvars+'capitalLatLon.npy')
SScapitalLatLon = np.load(wdvars+'subsaharancapitalLatLon.npy')

## Trucking cost dictionary
regionalTruckCost=np.zeros(shape=43)
truckCostDict={}
f=open(wddata+'travel_time/averagetkmcost.csv','r')
i=-1
for line in f:
        line=line[:-2]
        tmp=line.split(',')
        i+=1
        regionalTruckCost[i]=float(tmp[1])
        country=tmp[0]
        if country=='Congo': country='Congo (DRC)'
        if country=='Congo (Republic of the)': country='Congo'
```

```
        truckCostDict[country]=float(tmp[1])

##### national identifier grid #####
ds=gdal.Open(wddata+'../boundaries/gpw-v4-national-identifier-grid-rev10_30_sec_tif/gpw_v4_national_ident
width = ds.RasterXSize
height = ds.RasterYSize
gt = ds.GetGeoTransform()
minx = gt[0]
miny = gt[3] + width*gt[4] + height*gt[5]
maxx = gt[0] + width*gt[1] + height*gt[2]
maxy = gt[3]
pixelsizeNI=abs(gt[-1])

latc=np.ones(shape=(height))
lonc=np.ones(shape=(width))
for w in range(width):
        lonc[w]=minx+w*pixelsizeNI
for h in range(height):
        latc[h]=miny+h*pixelsizeNI

latc=latc[::-1]

nations=ds.ReadAsArray()
##### Scale to Africa #####
nations=nations[latc<28]
latc=latc[latc<28]
nations=nations[latc>-35]
latc=latc[latc>-35]

nations=nations[:,lonc<52]
lonc=lonc[lonc<52]
nations=nations[:,lonc>-19]
lonc=lonc[lonc>-19]

nations.dump(wdvars+'nations')

##### country codes #####
f=open(wddata+'../boundaries/countries_countryCodes.csv')
codes=np.zeros(shape=(247),dtype=int)
countryNames=[]
i=-1
for line in f:
        i+=1
        tmp=line.split(',')
        codes[i]=int(tmp[3])
        countryNames.append(tmp[0])

f=open(wddata+'../boundaries/africanCountries.csv','r')
africanCountries=[]
for line in f:
        africanCountries.append(line[:-1])

#indexing the country codes
countryToIndex={}
indexToCountry={}
indexedcodes=np.zeros(shape=len(africanCountries))
for i in range(len(africanCountries)):
        j=np.where(africanCountries[i]==np.array(countryNames))[0][0]
        indexedcodes[i]=codes[j]
        countryToIndex[africanCountries[i]]=indexedcodes[i]
        indexToCountry[indexedcodes[i]]=africanCountries[i]

countryToi={}
iToCountry={}
for i in range(len(indexedcodes)):
```

```
        index=indexedcodes[i]
        country=indexToCountry[index]
        countryToi[country]=i
        iToCountry[i]=country

################################
# Put truck cost on raster map
################################

truckCostMap=np.zeros(shape=(nations.shape))

for i in range(len(indexedcodes)):
        code=indexedcodes[i]
        country=indexToCountry[code]

        truckCostMap[nations==code]=truckCostDict[country]

truckCostMap[truckCostMap==0]=np.mean(regionalTruckCost)+0.01

plt.clf()
plt.imshow(truckCostMap,cmap=cm.jet)
#plt.plot([lonIndex0,lonIndex1],[latIndex0,latIndex1],'k*')
#plt.plot(x,yfit,'k.')
plt.title('Regional Trucking Costs, cents/tonne*km')
plt.colorbar()
plt.savefig(wdfigs +'truckCost',dpi=700)

################################
# Find avg cost
################################

print 'here'
# truckCostVector = array of avg trucking cost from each factory/port to each capital
truckCostVector=np.zeros(shape=(len(factoryLatLon[0]),len(SScapitalLatLon[0])))
for f in range(len(factoryLatLon[0])): # loop through factories
        latIndex0=np.abs(factoryLatLon[0,f]-latc).argmin() # nearest lat pixel to factory
        lonIndex0=np.abs(factoryLatLon[1,f]-lonc).argmin() # nearest lon pixel to factory

        for c in range(len(SScapitalLatLon[0])): # loop through capitals
                # check if same place
                if factoryLatLon[0,f]==SScapitalLatLon[0,c] and factoryLatLon[1,f]==SScapitalLatLon[1,c]:
                        continue

                latIndex1=np.abs(SScapitalLatLon[0,c]-latc).argmin() # nearest lat pixel to factory
                lonIndex1=np.abs(SScapitalLatLon[1,c]-lonc).argmin() # nearest lon pixel to factory

                lons=np.array([lonIndex0,lonIndex1])
                lats=np.array([latIndex0,latIndex1])
                iminlon=lons.argmin()
                imaxlon=lons.argmax()

                # x = each lon pixel between the cites
                x=np.zeros(shape=(abs(lonIndex1-lonIndex0)))
                for i in range(abs(lonIndex1-lonIndex0)):
                        x[i]=lons[iminlon]+i

                m,b=np.polyfit([lons[iminlon],lons[imaxlon]],[lats[iminlon],lats[imaxlon]],1)
                yfit=m*x+b # yfit = each (basically) lat pixel between the cities

                # add each pixel between the two cities
                for k in range(len(yfit)):
                        costtmp=truckCostMap[int(np.round(yfit[k],0)),int(x[k])]
                        truckCostVector[f,c]+=costtmp

                # average the costs
```

```
                truckCostVector[f,c] = truckCostVector[f,c]/len(yfit)
                print countrycosted[f],subsaharancountry[c],np.round(truckCostVector[f,c],3)

plt.clf()
plt.imshow(truckCostVector,cmap=cm.jet)
plt.title('Trucking Cost ($/tonne*km)')
plt.colorbar()
plt.ylabel('Factory/port')
plt.xlabel('Sub-Saharan Capitals')
plt.savefig(wdfigs+'vector_truckingcost.pdf')


groundDist = np.zeros(shape=(truckCostVector.shape))
File=open(wddata+'travel_time/INTLcapitaldistanceArray.csv')
f=-1
for line in File:
        f+=1
        tmp=line.split(',')
        for c in range(len(tmp)):
                groundDist[f,c]=float(tmp[c])

totalTruckingCost = truckCostVector*groundDist

plt.clf()
plt.imshow(groundDist,cmap=cm.jet)
plt.title('Ground Distance')
plt.colorbar()
plt.ylabel('Factory/port')
plt.xlabel('Sub-Saharan Capitals')
plt.savefig(wdfigs+'groundDist.pdf')

plt.clf()
plt.imshow(totalTruckingCost,cmap=cm.jet)
plt.title('Total Trucking Cost ($)')
plt.colorbar()
plt.ylabel('Factory/port')
plt.xlabel('Sub-Saharan Capitals')
plt.savefig(wdfigs+'total_trucking_cost.pdf')

np.save(wdvars+'truckCostVector.npy',truckCostVector)
np.save(wdvars+'groundDist.npy',groundDist)
np.save(wdvars+'totalTruckingCost.npy',totalTruckingCost)
```