sample_compute_distance_to_shore

March 15, 2022

The Coastal Grain Size Portal (C-GRASP) dataset Will Speiser, Daniel Buscombe, Evan Goldstein > Compute Sample Distance from Shore

The purpose of this notebook
This notebook will output a dataframe containing a new field with an estimated of the distance between each sample from a chosen C-GRASP dataset and the coastline. As C-Grasp file sizes vary completion of this task will vary with internet connectivity and computer processing power. This notebook provides simple code that calculates the distance between each sample from a chosen CGRASP dataset and the NaturalEarth 1:50m physical coastline polyline.
To do so, a user can choose a dataset of choice. This dataset is then called to your system and then converted into a GeoDataFrame The notebook then uses Cartopy to call the NaturalEarth 1:10m coastline polygon This coastline is then converted to a GeoDataFrame object and is cropped to an extent of the Eastern United States
This coastline is then converted to a GeoDataFrame object and is cropped to an extent of the Eastern United States Finally, using the GeoPandas distance function, the distance between nearest features in both GeoDataFrames (the chosen samples and the coastline polylines) is calculated and added to a new "Distance" field for each sample

[1]:
```
import pandas as pd
import geopandas as gpd
import shapefile
import json
import ipywidgets
from cartopy.feature import NaturalEarthFeature
import cartopy.io.shapereader as shpreader
import numpy as np
import matplotlib.pyplot as plt
```

Select a dataset

[2]:
```
#Dataset collection widget
zen=ipywidgets.Select(
    options=['Entire Dataset', 'Estimated Onshore Data', 'Verified Onshore␣
 ↪Data', 'Verified Onshore Post 2012 Data'],
    value='Entire Dataset',
    # rows=10,
    description='Dataset:',
    disabled=False
```

```
)

display(zen)
```

```
Select(description='Dataset:', options=('Entire Dataset', 'Estimated Onshore␣
↪Data', 'Verified Onshore Data', '…
```

Download the dataset

[3]:
```
url = 'https://zenodo.org/record/6099266/files/'
if zen.value=='Entire Dataset':
    filename='dataset_10kmcoast.csv'
if zen.value=='Estimated Onshore Data':
    filename='Data_EstimatedOnshore.csv'
if zen.value=='Verified Onshore Data':
    filename='Data_VerifiedOnshore.csv'
if zen.value=='Verified Onshore Post 2012 Data':
    filename='Data_Post2012_VerifiedOnshore.csv'
print("Downloading {}".format(url+filename))
```

```
Downloading
https://zenodo.org/record/6099266/files/Data_Post2012_VerifiedOnshore.csv
```

The next cell will download the CGRASP dataset and read it in as a pandas dataframe with variable name `df`

[4]:
```
url=(url+filename)
print('Retrieving Data, Please Wait')
#retrieve data
df=pd.read_csv(url)
print('Sediment Data Retrieved!')
```

```
Retrieving Data, Please Wait
Sediment Data Retrieved!
```

Let's take a quick look at the top of the file

[5]:
```
df.head()
```

[5]:

	ID	Sample_ID	Sample_Type_Code	Project	dataset	\
0	876	SPIbeach5	1	SandSnap, image taken by:	sandsnap	
1	878	SPI6	1	SandSnap, image taken by:	sandsnap	
2	877	SPI6	1	SandSnap, image taken by:	sandsnap	
3	1429	SPIbeach4	1	SandSnap, image taken by:	sandsnap	
4	1430	SPIbeach3	1	SandSnap, image taken by:	sandsnap	

	Date	Location_Type	latitude	longitude	Contact	…	\
0	2021-11-08	Beach?Y	26.12871	-97.16718	Sandsnap, USACE	…	
1	2021-11-08	Beach?Y	26.12899	-97.16713	Sandsnap, USACE	…	
2	2021-11-08	Beach?Y	26.12899	-97.16713	Sandsnap, USACE	…	

```
3  2021-11-08       Beach?Y  26.16883  -97.17248  Sandsnap, USACE  …
4  2021-11-08       Beach?Y  26.16885  -97.17284  Sandsnap, USACE  …

        d16       d25       d30       d50       d65       d75       d84  \
0  0.565657  0.624976  0.657068  0.785439  0.889342  1.016927  1.131754
1  0.565657  0.624976  0.657068  0.785439  0.889342  1.016927  1.131754
2  0.565657  0.624976  0.657068  0.785439  0.889342  1.016927  1.131754
3  0.565657  0.624976  0.657068  0.785439  0.889342  1.016927  1.131754
4  0.565657  0.624976  0.657068  0.785439  0.889342  1.016927  1.131754

        d90       d95 Notes
0  1.276942  1.397932   NaN
1  1.276942  1.397932   NaN
2  1.276942  1.397932   NaN
3  1.276942  1.397932   NaN
4  1.276942  1.397932   NaN

[5 rows x 34 columns]
```

Now we'll convert that data frame to a GeoDataFrame, that way we can do spatial calculations on it.

[6]:
```
gdf=gpd.GeoDataFrame(df,geometry=gpd.points_from_xy(df.longitude, df.latitude))
gdf=gdf.set_crs(epsg=4326)
print('Sediment Data Converted to a GeoDataFrame, Next cell retrieves Natural␣
 ↪Earth Data')
```

```
Sediment Data Converted to a GeoDataFrame, Next cell retrieves Natural Earth
Data
```

Now let's call that NaturalEarth Coastline polyline:

[7]:
```
coast = shpreader.natural_earth(resolution='10m',category='physical',␣
 ↪name='coastline')
coast=gpd.read_file(coast)
print('Natural Earth Data retrieved and converted to a GeoDataFrame')
```

```
Natural Earth Data retrieved and converted to a GeoDataFrame
```

This next cell creates a json to define bounds for the East Coast US (C-Grasp's area of study)

[8]:
```
#Define Bounds for the East Coast to Clip US Coastal Poly
eastcoast=json.loads(("""
     {"type": "FeatureCollection", "features": [{
      "type": "Feature",
      "properties": {},
      "geometry": {
        "type": "Polygon",
        "coordinates": [
```

```
        [
          [
            -97.294921875,
            24.766784522874453
          ],
          [
            -94.130859375,
            22.51255695405145
          ],
          [
            -86.30859375,
            21.94304553343818
          ],
          [
            -80.244140625,
            16.130262012034756
          ],
          [
            -65.0390625,
            15.792253570362446
          ],
          [
            -66.610107421875,
            44.66083904265621
          ],
          [
            -66.7694091796875,
            44.8500274926005
          ],
          [
            -66.8902587890625,
            44.779885502772736
          ],
          [
            -67.5,
            47.69497434186282
          ],
          [
            -69.60937499999999,
            47.754097979680026
          ],
          [
            -71.630859375,
            45.521743896993634
          ],
          [
            -101.25,
```

```
              30.221101852485987
            ],
            [
              -97.294921875,
              24.766784522874453
            ]
          ]
        ]
      }
    }]}
"""))
```

Here, we crop that NaturalEarth coastline with the East Coast Json

[9]:
```
eastcoast = gpd.GeoDataFrame.from_features(eastcoast) #convert east coast json␣
 ↪to geoadataframe
eastcoast=eastcoast.set_crs(epsg=4326)
eastcoast_line=gpd.clip(coast, eastcoast) #clip natural earth data to east coat
eastcoast = eastcoast_line.reset_index(drop=True)
```

In this next cell we will convert both all of the spatial data to EPSG 3857 crs so our distance measurements are in meters

[10]:
```
eastcoast=eastcoast.to_crs(epsg=3857)#project to 3857 to make calculation in␣
 ↪meters rather than degrees
gdf=gdf.to_crs(epsg=3857)#project to 3857 to make calculation in meters rather␣
 ↪than degrees
gdf = gdf.reset_index(drop=True)
```

This next cell runs the distance calculation

[11]:
```
gdf['Distance']=gdf.geometry.apply(lambda x: eastcoast.distance(x).min())␣
 ↪#calculate distance using geopandas and set it to df
```

Let's see how that worked:

[12]:
```
print(gdf['Distance'])
```

```
0       326.228328
1       318.924893
2       318.924893
3       353.620370
4       393.133333
           …
2108    123.020696
2109    123.020696
2110    129.659719
2111    314.878962
```

```
2112    314.878962
Name: Distance, Length: 2113, dtype: float64
```

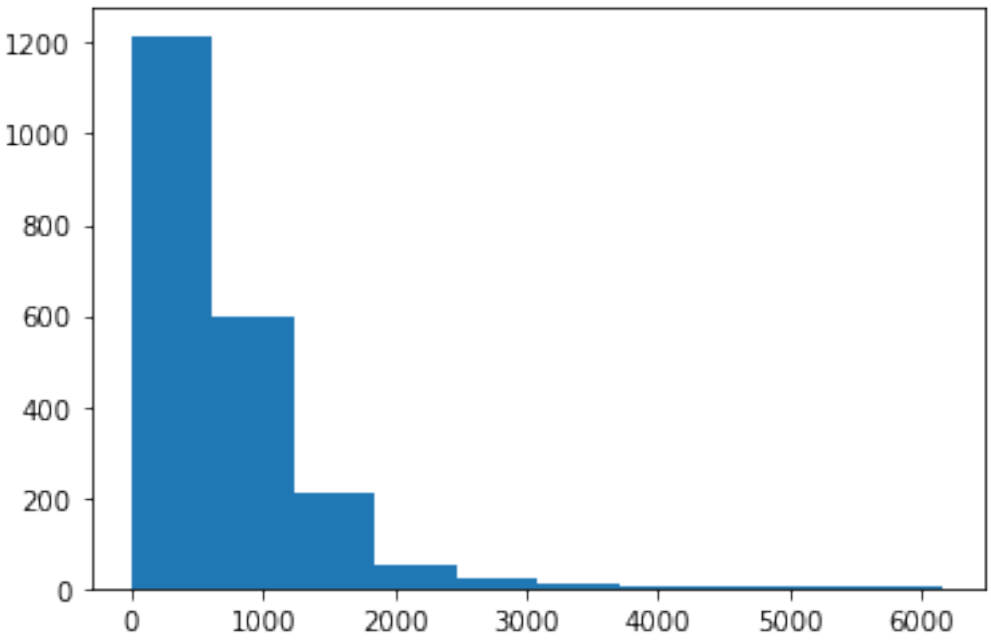
Now lets turn the dataframe back to a Pandas dataframe and drop the "geometry" column

[13]:
```
df=pd.DataFrame(gdf) #convert geodataframe to pandas data frame
df=df.drop(columns='geometry')
```

Make a plot of the distribution of distances

[17]:
```
plt.hist(df['Distance'])
```

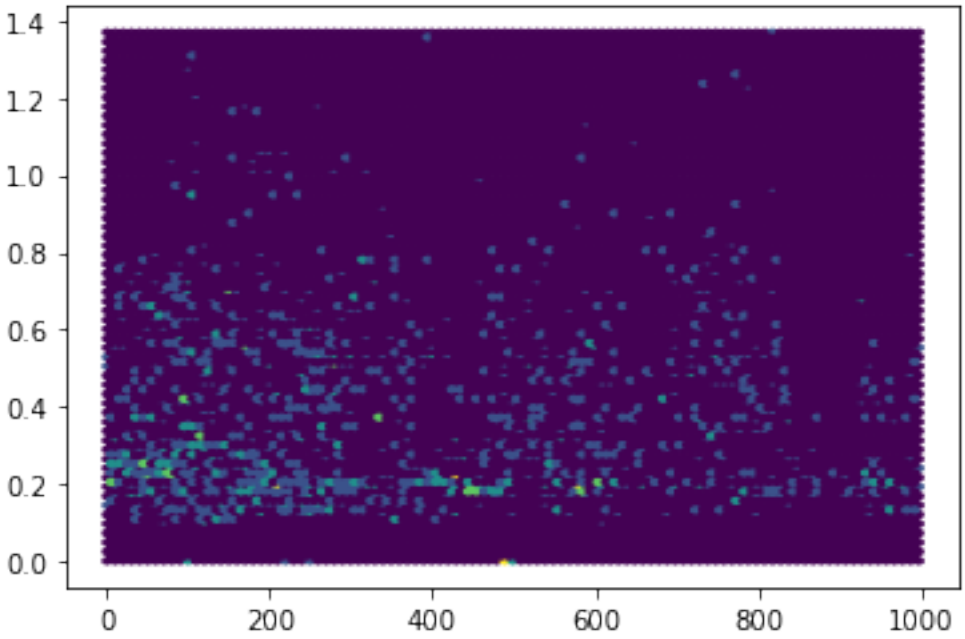
[17]:
```
(array([1216.,  597.,  208.,   49.,   20.,    9.,    4.,    5.,    3.,
           2.]),
 array([1.39043965e-01, 6.16692680e+02, 1.23324632e+03, 1.84979995e+03,
        2.46635359e+03, 3.08290722e+03, 3.69946086e+03, 4.31601449e+03,
        4.93256813e+03, 5.54912176e+03, 6.16567540e+03]),
 <BarContainer object of 10 artists>)
```



Plot the distributions of sand (d50 $<$ 2mm) versus distance within 1km of the shore

[24]:
```
plt.hexbin(df['Distance'][(df['d50']<2) & (df['Distance']<1000)],␣
↪df['d50'][(df['d50']<2) & (df['Distance']<1000)])
```

[24]:
```
<matplotlib.collections.PolyCollection at 0x7f00b0634be0>
```



0.0.1 Write to file

Finally, define a csv file name for the output dataframe

[ ]:
```
output_csvfile='../data_plus_locations.csv'
```

write the data to that csv file

[ ]:
```
df.to_csv(output_csvfile)
```