sample_query

March 15, 2022

The Coastal Grain Size Portal (C-GRASP) dataset Will Speiser, Daniel Buscombe, Evan Goldstein > Query Samples

The purpose of this notebook
This notebook will output a csv file containing all of the data from a chosen C-GRASP dataset that the user has queried spatially and temporally to best suit their needs. It will also display statistics such as the number of samples queried and histograms of the dates they cover. This notebook provides simple code to interactively query C-GRASP data.
To do so, the user selects their dataset of choice and the year range of data in which they are interested in. Then, the notebook creates an interactive map showing the selected dataset using iPyleaflet. The user then draws a polygon on this map to select which samples they are interested in collecting Then the notebook converts each CUDEM cell value to a csv containing the CUDEM file's depth value and location for each cell. After, these csv's are combined into one dataframe After the CUDEM data conversion, the chosen C-GRASP dataset is downloaded and converted to a dataframe. Finally the two datasets are converted to GeoPandasData frames and are joined by proximity, assigning each downloaded CGRASP sample a depth from the nearest CUDEM value. This data is then downloaded as a csv to the user's system.

[1]:
```
import pandas as pd
import geopandas as gpd
import shapefile
import json
from ipyleaflet import (
    Map,
    Marker,
    TileLayer, ImageOverlay,
    Polyline, Polygon, Rectangle, Circle, CircleMarker,
    GeoJSON,
    DrawControl, basemaps, basemap_to_tiles, GeoData
)

import pyproj
from shapely.ops import transform
from shapely.geometry import Polygon
import ipywidgets
import numpy as np
import urllib.request
```

0.1 In this cell, choose your dataset of interest and then enter in the year range that you want to query. Note: If you are only interested in one year, enter that year as both your start and end date

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

display(zen)


print('Enter Year Range of Interest Interest Below:')

#Lower bound year text enter widget

y0=ipywidgets.Text(
    value='yyyy',
    placeholder='Type something',
    description='Start Year:',
    disabled=False
)

display(y0)

#Upper bound year text enter widget
y1=ipywidgets.Text(
    value='yyyy',
    placeholder='Type something',
    description='End Year:',
    disabled=False
)

display(y1)
```

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

```
Enter Year Range of Interest Interest Below:
```

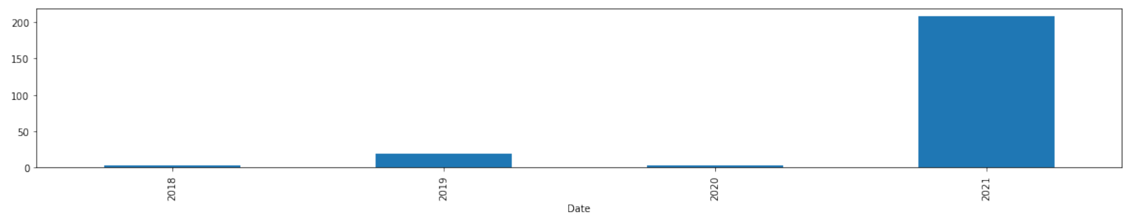
```
Text(value='yyyy', description='Start Year:', placeholder='Type something')
```

```
Text(value='yyyy', description='End Year:', placeholder='Type something')
```

0.1.1 Download your data!

[3]:
```
if float(y1.value)-float(y0.value)<0: #Making sure your date range is␣
↪appropriate
    print('Error! End Date Proceeds Start Date!')

#Call the chosen dataset from Zenodo
else:
    url = 'https://zenodo.org/record/6099266/files/'
    if zen.value=='Entire Coastal Dataset':
        filename='dataset_10kmcoast.csv'
    if zen.value=='Estimated Onshore Dataset':
        filename='Data_EstimatedOnshore.csv'
    if zen.value=='Verified Onshore Dataset':
        filename='Data_VerifiedOnshore.csv'

    url=(url+filename)
    print('Retrieving Data, Please Wait')
    #retrieve data
    df=pd.read_csv(url)
    print('Data Retrieved! Now Subsetting to Year')
    df['Date'] = pd.to_datetime(df['Date']) #Convert sample date to pandas␣
↪datetime object
    df['year'] = df['Date'].dt.year #Extract year from datetime object
    df=df.loc[df.year>=float(y0.value), :]      #subset by lower year
    df=df.loc[df.year<=float(y1.value), :]  #subset by upper year

    df = pd.DataFrame(df.drop(columns='year'))

    if float(y1.value)-float(y0.value)>0:
        print()
        print('Done. '+str(len(df))+' samples available from dates the␣
↪following dates:')
        df=df[df['Date'].astype("datetime64").dt.year<2060] #remove any data␣
↪with typos from source data
        #print statistics
        df['Mean'].groupby(df['Date'].astype("datetime64").dt.year).count().
↪plot(kind="bar",figsize=(20,3))
        print()
    else:
        print('Done. '+str(len(df))+' samples available')
```

```
Retrieving Data, Please Wait
Data Retrieved! Now Subsetting to Year

Done. 269 samples available from dates the following dates:
```



Turn the data into a GeoDataFrame so you can visualize samples on the interactive map

```
[4]: #make map with geodataframe
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.longitude, df.
 ↪latitude)) #convert dataframe to geodataframe
gdf['Date'] = gdf['Date'].astype(str)

gdf= GeoData(geo_dataframe = gdf)
watercolor = basemap_to_tiles(basemaps.Esri.WorldImagery, crs='espg:4326')
```

0.2 Use iPyLeaflet to query your samples

Once you run the cell below, an interactive map will appear In this next cell, an ipyleafletmap will appear where you spatially query your desired data (the points are all available data from your temporally queried dataset).

- To do so, navigate around the map and then click the square button to call the rectangle selection tool. Click and drag the square over your samples of interest.

See the Readme.md file for more information on how to use this map.

```
[5]: m = Map(layers=(watercolor, ), center=(35,-75 ), zoom=4)

m.add_layer(gdf)



#Add draw tool

dc = DrawControl(marker={'shapeOptions': {'color': '#0000FF'}},
                 rectangle={'shapeOptions': {'color': '#0000FF'}},
                 circle={'shapeOptions': {'color': '#0000FF'}},
                 circlemarker={},
                 )
def handle_draw(target, action, geo_json):
    print(action)
    print(geo_json)

dc.on_draw(handle_draw)
m.add_control(dc)
```

```
m
```

```
Map(center=[35, -75], controls=(ZoomControl(options=['position', 'zoom_in_text',␣
↪'zoom_in_title', 'zoom_out_te…
```

```
created
{'type': 'Feature', 'properties': {'style': {'stroke': True, 'color': '#0000FF',
'weight': 4, 'opacity': 0.5, 'fill': True, 'fillColor': None, 'fillOpacity':
0.2, 'clickable': True}}, 'geometry': {'type': 'Polygon', 'coordinates':
[[[-91.36559, 28.556862], [-91.36559, 32.714588], [-85.211972, 32.714588],
[-85.211972, 28.556862], [-91.36559, 28.556862]]]}}
```

This cell extracts the data from your above query

```
[7]: #This extracts the polygon drawn in the above map
bounds= (dc.last_draw).get("geometry").get("coordinates")[0] #get geometry of␣
↪drawn polygon
bounds_poly=Polygon(bounds) #make geometry into a spatial polygon
bounds_gpd=gpd.GeoDataFrame(geometry=[bounds_poly]) # Turn the polygon into a␣
↪geopandas dataframe
polygon=bounds_gpd.set_crs('EPSG:4326') #set the crs to the same as the sample␣
↪geodataframe
```

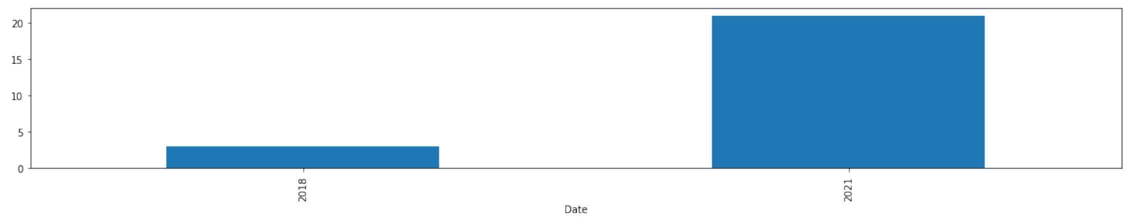
#This then subsets your data set

```
[8]: df=gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.longitude, df.
↪latitude))
df=df.set_crs('EPSG:4326')
df=gpd.clip(df, polygon)
df=pd.DataFrame(df)
```

Lets take a look at some of the statistics of your queried data

```
[9]: #Print date statistics if more than one year of data
if float(y1.value)-float(y0.value)>0:
    print(str(len(df))+' samples downloaded from dates:')
    df['Mean'].groupby(df['Date'].astype("datetime64").dt.year).count().
↪plot(kind="bar",figsize=(20,3))
#Print length statistics if one year of data
else:
    print(str(len(df)) +'samples downloaded')
```

```
32 samples downloaded from dates:
```



0.3 Define the output folder path and file name

```
[ ]: out_path='../data_queried.csv'
```

And finally, download the data

```
[ ]: df.to_csv('../data.csv')

     print('Data downloaded!')
```