sample_assign_location

March 15, 2022

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

The purpose of this notebook
This notebook will output a dataframe containing all of the data from a chosen C-GRASP dataset with a new field containing the address of each sample. As the API needs to be called for each individual sample, it is recommended that the user selects data sparingly if time is a constrait as processing time may take a while depending on internet connectivity. This notebook provides simple code in order to assign an address/location name to samples within a dataset.
To do so, a user can input a dataset of choice. The notebook then calls in the Open Street Maps geocoder API and uses reverse geocoding to assign an address to a lat/lon location.

[1]:
```
import pandas as pd
import geocoder
import requests
import ipywidgets
```

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 that 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]
```

0.0.1 Add location field

This cell adds a new 'Location' column containing the address of each sample extracted from Open Street Maps

[6]:
```
#adding empty column
df["Location"] = ""

#Loop through each sample
count=0
for i in range(0,len(df)):
    try:
        lat=df['latitude'].iloc[i]
        lon=df['longitude'].iloc[i]
        #This next line runs a reverse geocode on your sample lat/lons using OSM
        g=geocoder.osm([lat,lon], method='reverse')
        #This line extracts the address fron the queried OSM json
        df['Location'].iloc[i]=g.json['address']
        count=count+1
    except:
        pass # This skips errors for locations that are not assignable (think␣
 ↪offshore samples etc)
```

```
/tmp/ipykernel_886252/3041134942.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['Location'].iloc[i]=g.json['address']
```

Let's view those locations

[7]:
```
df['Location']
```

[7]:
```
0        5656, Gulf Boulevard, South Padre Island, Came…
1        5612, Gulf Boulevard, South Padre Island, Came…
2        5612, Gulf Boulevard, South Padre Island, Came…
3        Ocean Boulevard, South Padre Island, Cameron C…
4        Ocean Boulevard, South Padre Island, Cameron C…
                               …
```

```
2108    5, Commonwealth Avenue, Salisbury, Essex Count…
2109    5, Commonwealth Avenue, Salisbury, Essex Count…
2110    5, Commonwealth Avenue, Salisbury, Essex Count…
2111    427, Mile Road, Wells Beach, Wells, York Count…
2112    427, Mile Road, Wells Beach, Wells, York Count…
Name: Location, Length: 2113, dtype: object
```

0.0.2 Write to file

Finally, define a csv file name for the output dataframe

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

write the data to that csv file

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

```
[ ]:
```