sample_compute_percentiles

March 15, 2022

The Coastal Grain Size Portal (C-GRASP) dataset Will Speiser, Daniel Buscombe, Evan Goldstein > Interpolate Percentiles from Other Dataset Percentiles

The purpose of this notebook
This notebook will output a CSV containing all of the data from a chosen C-GRASP dataset with a new field containing an cumulative distribution percentile interpolated from pre-existing dataset distribution percentile values. As C-Grasp file sizes vary completion of this task will vary with internet connectivity. This notebook provides simple code that interpolates input distribution percentile values from already calculated values.
To do so, a user can choose a dataset of choice and then types the percentile they wish to calculate. The notebook then runs uses a the scipy interpolation function to calculate the input percentile in mm units.

[1]:
```
import pandas as pd
import scipy
from scipy.interpolate import interp1d
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', '…
```

Enter a distribution you want to calculate into the textbox e.g.: 'd86'

[3]:
```
dist=ipywidgets.Text(
    value='d86',
    placeholder='Type something',
    description='Distribution:',
    disabled=False
)

display(dist)
```

```
Text(value='d86', description='Distribution:', placeholder='Type something')
```

Download the dataset

[4]:
```
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`

[5]:
```
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

[6]:
```
df.head()
```

[6]:

	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]
```

The next cell will create seperate the number value from the distribution you input for calculations in the cell after (e.g. '86' from 'd86)

[7]:
```
percentile_value=dist.value.split('d')[1]
prcntl=float(percentile_value)/100
```

0.1 In this cell you will estimate the input distribution percetile for each sample that has other distribution information available using the Scipy interpolation function and add it to a new dataframe column

[8]:
```
df[dist.value]='' #create a new blank column for your values calculated int he␣
 ↪loop below

#This loop will iterate for each sample in the dataset
for i in range(0,len(df)):
    try:
        #Set variables for columns of provided percentile distributions. E.g.:
        d10=df['d10'].iloc[i]
        d16=df['d16'].iloc[i]
        d25=df['d25'].iloc[i]
        d50=df['d50'].iloc[i]
        d65=df['d65'].iloc[i]
        d84=df['d84'].iloc[i]
        d90=df['d90'].iloc[i]
```

```
        d95=df['d95'].iloc[i]

        #Here, you are creating an array of the variables you just created. Make sure to put each one that you set in the brackets
        grain_size_bins=[d10,d16,d25,d50,d65,d84,d90,d95]

        #Here, you are creating an array of the percentile values of the distributions for the above respective variables. Make sure to put each one that you set in the brackets
        grain_size_frequencies=[.1,.16,.25,.5,.65,.84,.9,.95]

        #Here we will use scipy's interpolation toolbox to create a function that calculates unknow distributions of interest.
        distribution = scipy.interpolate.interp1d(grain_size_frequencies, grain_size_bins, bounds_error=False, fill_value='extrapolate')

        #Here we will create a new column for the input percentile distributions in which we would like to calculate respective grainsize values
        #The extracted numerical value from the input text will be put into the scipy interpolation tool
        df.loc[i,[dist.value]] = distribution(prcntl)
    except:
        pass
```

Let's check out that new distribution percentile column

[9]:
```
df[dist.value]
```

[9]:
```
0         1.1801496377013498
1         1.1801496377013498
2         1.1801496377013498
3         1.1801496377013498
4         1.1801496377013498
                 …
2108     0.5860622917060153
2109    0.33590998043052833
2110     0.6092682926829267
2111    0.28173399266666665
2112       1.50955207971414
Name: d86, Length: 2113, dtype: object
```

0.1.1 Write to file

Finally, define a csv file name for the output dataframe

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

Write the data to that csv file

df.to_csv(output_csvfile)