RACE_FinishedGoodsInventory_inidcator

June 22, 2023

1 Finished Goods Inventory indicator for RACE

Indicator E8 is calculated at the logistic areas operated at the production sites such as motor vehicle manufacturing, the large number of metal objects (e.g. machinery, vehicles) produces a strong signature in the radar backscatter signal. The observations are provided by the Synthetic Aperture Radar such as the C-band Copernicus Sentinel-1 satellites (source: RACE ESA)

1.1 Importing needed modules

[1]:
```
#imports
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

[2]:
```
import datetime as dt
import os

from matplotlib import mlab
from shapely import geometry, wkt

import geopandas as gpd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from datetime import datetime

from sentinelhub import (
    CRS,
    BBox,
    DataCollection,
    Geometry,
    SentinelHubStatistical,
    SentinelHubStatisticalDownloadClient,
    SHConfig,
    parse_time,
    geometry,
```

```
    SentinelHubCatalog,
    bbox_to_dimensions,
)
```

1.2 SentinelHub account configuration

[3]:
```
sh_client_id=""
sh_client_secret=""

if not sh_client_id:
    sh_client_id = os.environ["SH_CLIENT_ID"]

if not sh_client_secret:
    sh_client_secret = os.environ["SH_CLIENT_SECRET"]


if not sh_client_id or not sh_client_secret:
    raise ValueError("No valid Sentinel HUB credentials are available. Plese␣
 ↪contact system administrator.")

config = SHConfig()

config.sh_client_id = sh_client_id
config.sh_client_secret = sh_client_secret

#OUTPUT_DIR = os.path.join(USR_PATH, "output") #local folder
```

1.3 Input Parameters

Volvo Cars, Ghent, Belgium

[4]:
```
aoi = "POLYGON ((3.754824 51.096633, \
              3.753451 51.096242, \
              3.755747 51.093102, \
              3.755661 51.09511, \
              3.755211 51.094989, \
              3.754953 51.095393, \
              3.755211 51.09608, \
              3.755125 51.0967, \
              3.755447 51.097953, \
              3.755168 51.098048, \
              3.755009 51.097886, \
              3.754116 51.097697, \
              3.754824 51.096633))"
```

```
time_period= "2023-01-01/2023-05-31"

OUTPUT_DIR = "/home/jovyan/PLES_WORKSPACE/Anca_E8/result-data/e8" # will be␣
 ↪copied to the local folder of the user requesting the data
```

1.3.1 Parameter preparation

[5]:
```
from_to = time_period.split("/")

interval = (f'{from_to[0]}T00:00:00Z', f'{from_to[1]}T23:59:59Z')
print(f'interval: {interval}')

geometry = Geometry(aoi, crs=CRS.WGS84)
print(geometry)

size = bbox_to_dimensions(geometry.bbox, resolution=5)
print(f'size: {size}')

collection = DataCollection.SENTINEL1_IW
print(f'collection: {collection}')
```

```
interval: ('2023-01-01T00:00:00Z', '2023-05-31T23:59:59Z')
Geometry(POLYGON ((3.754824 51.096633, 3.753451 51.096242, 3.755747 51.093102,
3.755661 51.09511, 3.755211 51.094989, 3.754953 51.095393, 3.755211 51.09608,
3.755125 51.0967, 3.755447 51.097953, 3.755168 51.098048, 3.755009 51.097886,
3.754116 51.097697, 3.754824 51.096633)), crs=CRS('4326'))
size: (31, 110)
collection: DataCollection.SENTINEL1_IW
```

1.4 Evalscript definition

Evalscript to retreve the VH channel in linear units

[6]:
```
#evalscript (linear units)
evalscript = """
//VERSION=3
function setup() {
  return {
    input: [{
      bands: ["VH", "dataMask"]
    }],
    output: [
      {
        id: "default",
        bands: 1
```

```
      },
      {
        id: "dataMask",
        bands: 1
      }]
  };
}

function evaluatePixel(samples) {
    return {
        default: [samples.VH],
        dataMask: [samples.dataMask],
    };
}

"""
```

1.5 SentinelHub statistical API request (linear units)

[7]:
```
# Definition of the SentinelHUB statistical API request (linear unit)
request = SentinelHubStatistical(
    aggregation=SentinelHubStatistical.aggregation(
        evalscript=evalscript,
        time_interval=interval,
        aggregation_interval='P1D',
        size=size
    ),
    input_data=[
        SentinelHubStatistical.input_data(
            collection,
            other_args={"dataFilter": {"mosaickingOrder":␣
 ↪"mostRecent","resolution": "HIGH"},
                        "processing": {"orthorectify": "True","backCoeff":␣
 ↪"GAMMA0_TERRAIN","demInstance": "COPERNICUS"}},
      ),
    ],
    geometry=geometry,
    config=config
)
```

[8]:
```
# helper function

def stats_to_df(stats_data):
    """Transform Statistical API response into a pandas.DataFrame"""
    df_data = []
```

```
    for single_data in stats_data["data"]:
        df_entry = {}
        is_valid_entry = True

        df_entry["interval_from"] = parse_time(single_data["interval"]["from"]).
↪date()
        df_entry["interval_to"] = parse_time(single_data["interval"]["to"]).
↪date()

        for output_name, output_data in single_data["outputs"].items():
            for band_name, band_values in output_data["bands"].items():

                band_stats = band_values["stats"]
                if band_stats["sampleCount"] == band_stats["noDataCount"]:
                    is_valid_entry = False
                    break
                band_name='gamma0'
                for stat_name, value in band_stats.items():
                    col_name = f"{output_name}_{band_name}_{stat_name}"
                    if stat_name == "percentiles":
                        for perc, perc_val in value.items():
                            perc_col_name = f"{col_name}_{perc}"
                            df_entry[perc_col_name] = perc_val
                    else:
                        df_entry[col_name] = value

        if is_valid_entry:
            df_data.append(df_entry)

    return pd.DataFrame(df_data)
```

1.6 Execute statistical API call

[9]:
```
%%time

stats = request.get_data()[0]
```

```
CPU times: user 41.8 ms, sys: 4.53 ms, total: 46.3 ms
Wall time: 7.46 s
```

[10]:
```
stats
```

[10]:
```
{'data': [{'interval': {'from': '2023-01-02T00:00:00Z',
    'to': '2023-01-03T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
```

```
        'max': 0.16340191662311554,
        'mean': 0.020170348377154566,
        'stDev': 0.022850378505722715,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-01-11T00:00:00Z', 'to': '2023-01-12T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.00053760054288432,
        'max': 0.24312031269073486,
        'mean': 0.04211890468930785,
        'stDev': 0.03629304571080026,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-01-14T00:00:00Z', 'to': '2023-01-15T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0006729860324412584,
        'max': 0.19775378704071045,
        'mean': 0.031004727622934405,
        'stDev': 0.027735737722442297,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-01-23T00:00:00Z', 'to': '2023-01-24T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0009634400485083461,
        'max': 0.2604427635669708,
        'mean': 0.037104581457690876,
        'stDev': 0.029410454415559686,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-01-26T00:00:00Z', 'to': '2023-01-27T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.19697800278663635,
        'mean': 0.014065844334907308,
        'stDev': 0.017876060166642574,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-02-04T00:00:00Z', 'to': '2023-02-05T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0001648256293265149,
        'max': 0.2849183678627014,
        'mean': 0.044030191910618834,
        'stDev': 0.038471051113867075,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-02-07T00:00:00Z', 'to': '2023-02-08T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.29712384939193726,
```

```
        'mean': 0.027096869881119905,
        'stDev': 0.03569749047693281,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-02-16T00:00:00Z', 'to': '2023-02-17T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0004944409593008459,
        'max': 0.3378330171108246,
        'mean': 0.0458258435200669,
        'stDev': 0.0403675600325962,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-02-19T00:00:00Z', 'to': '2023-02-20T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0004193929780740291,
        'max': 0.2099660336971283,
        'mean': 0.026614709294525835,
        'stDev': 0.023548831413114472,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-02-28T00:00:00Z', 'to': '2023-03-01T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0006654243916273117,
        'max': 0.3600568175315857,
        'mean': 0.047297481470391124,
        'stDev': 0.05489599729503184,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-03-03T00:00:00Z', 'to': '2023-03-04T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.29821091890335083,
        'mean': 0.03507009022842485,
        'stDev': 0.04387753391867001,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-03-12T00:00:00Z', 'to': '2023-03-13T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.4448762536048889,
        'mean': 0.04779726556414885,
        'stDev': 0.04985294549104764,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-03-15T00:00:00Z', 'to': '2023-03-16T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0008856073254719377,
        'max': 0.37921154499053955,
        'mean': 0.02871270166527556,
```

```
        'stDev': 0.034716001276427405,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-03-24T00:00:00Z', 'to': '2023-03-25T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.23608514666557312,
        'mean': 0.0243165254313782,
        'stDev': 0.032566251029009466,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-03-27T00:00:00Z', 'to': '2023-03-28T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.23828335106372833,
        'mean': 0.01775118943566635,
        'stDev': 0.018320137752728106,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-04-05T00:00:00Z', 'to': '2023-04-06T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.45296838879585266,
        'mean': 0.03900137488058902,
        'stDev': 0.04202459387819732,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-04-08T00:00:00Z', 'to': '2023-04-09T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.2152615636587143,
        'mean': 0.022333543463880293,
        'stDev': 0.026202080323672713,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-04-17T00:00:00Z', 'to': '2023-04-18T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.002032041549682617,
        'max': 0.5498997569084167,
        'mean': 0.044835481947130516,
        'stDev': 0.04601284805089234,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-04-20T00:00:00Z', 'to': '2023-04-21T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.16349147260189056,
        'mean': 0.02873210008248631,
        'stDev': 0.023935063075955932,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}},
  {'interval': {'from': '2023-04-29T00:00:00Z', 'to': '2023-04-30T00:00:00Z'},
```

```
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.0006535442662425339,
        'max': 0.47605401277542114,
        'mean': 0.03541633579179504,
        'stDev': 0.040027245171222525,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}},
  {'interval': {'from': '2023-05-02T00:00:00Z', 'to': '2023-05-03T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.17263391613960266,
        'mean': 0.02240079491528319,
        'stDev': 0.020896857186153875,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}},
  {'interval': {'from': '2023-05-11T00:00:00Z', 'to': '2023-05-12T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.00099738291464746,
        'max': 0.5997887849807739,
        'mean': 0.04238483274904582,
        'stDev': 0.04961376767103343,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}},
  {'interval': {'from': '2023-05-14T00:00:00Z', 'to': '2023-05-15T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.1937440186738968,
        'mean': 0.027808189507421043,
        'stDev': 0.02670213142682897,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}},
  {'interval': {'from': '2023-05-23T00:00:00Z', 'to': '2023-05-24T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min':
0.00015107712533790618,
        'max': 0.7867693305015564,
        'mean': 0.038771781861980645,
        'stDev': 0.04988185161692733,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}},
  {'interval': {'from': '2023-05-26T00:00:00Z', 'to': '2023-05-27T00:00:00Z'},
   'outputs': {'default': {'bands': {'B0': {'stats': {'min': 0.0,
        'max': 0.13981038331985474,
        'mean': 0.02774544701894828,
        'stDev': 0.0230227049374178,
        'sampleCount': 3410,
        'noDataCount': 2014}}}}}],
 'status': 'OK'}
```

2 Convert to Dataframe

[11]:
```
#convert statistical data to pandas dataframe
df = stats_to_df(stats)
df=df[['interval_from','default_gamma0_mean']]
df.rename(columns = {'interval_from':'Acquisition date','default_gamma0_mean':
 ↪'mean_gamma0'}, inplace = True)
df.head()
```

[11]:

	Acquisition date	mean_gamma0
0	2023-01-02	0.020170
1	2023-01-11	0.042119
2	2023-01-14	0.031005
3	2023-01-23	0.037105
4	2023-01-26	0.014066

2.1 Removing outliers

[12]:
```
def remove_and_interpolate_outliers(df):
    import pandas as pd
    import numpy as np
    # Read stats file

    # create thresholds
    min_threshold, max_threshold = df['mean_gamma0'].quantile([0.01,0.99])

    # create a new dataframe excluding the outlier rows
    outlier_removed = df[(df['mean_gamma0'] < max_threshold)&(df['mean_gamma0']␣
 ↪> min_threshold)]
    outliersId = df[(df['mean_gamma0'] >= max_threshold) | (df['mean_gamma0']␣
 ↪<= min_threshold)].index

    # diplay the shape of new dataset
    print(outlier_removed.shape)
    # the orignal shape
    print(df.shape)

    df2=df.copy()
    # fill with nans
    df2.loc[outliersId,['mean_gamma0']]=np.nan
    # interpolate nans
    outlier_rem_interp = df2.interpolate(method='nearest',order=2)
    return outlier_rem_interp
```

[13]:
```
df_filt=remove_and_interpolate_outliers(df)
df_filt.head()
```

```
(23, 2)
(25, 2)
```

[13]:

	Acquisition date	mean_gamma0
0	2023-01-02	0.020170
1	2023-01-11	0.042119
2	2023-01-14	0.031005
3	2023-01-23	0.037105
4	2023-01-26	0.037105

2.2 Exporting results

[14]:
```
df.to_csv(os.path.join(OUTPUT_DIR,'average_gamma0_time_series.csv'),index=False)
```

[15]:
```
import matplotlib.pyplot as plt
import matplotlib

fig = df_filt.plot(x='Acquisition date', figsize=(20, 10), fontsize=10).
 ↪get_figure()
plt.title('Time series of normalized radar backscatter [gamma0]')
plt.ylabel('gamma0')
fig.savefig(os.path.join(OUTPUT_DIR,'average_gamma0_time_series.pdf'))
```