Brezice2017

November 3, 2020

Notebook za prevzem Sentinel2 podatkov za obmoja CRP.
Osnovni gradniki EOLEARN so EOPatch, EOTask in EOWorkflow.

In [1]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
```

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
from eolearn.core import EOTask, EOPatch, LinearWorkflow, Dependency, FeatureType, SaveT
from eolearn.io import ExportToTiff
from eolearn.io import S2L2AWCSInput, AddSen2CorClassificationFeature
from eolearn.mask import AddValidDataMaskTask
from eolearn.features import SimpleFilterTask
```

In [109]:
```
from sentinelhub import BBoxSplitter, CRS, MimeType, BBox, CustomUrlParam, Geometry
```

In [4]:
```
import geopandas
```

In [5]:
```
obmocja = geopandas.read_file('E:/2015/V4-1811/obmocja/CRP_monitoring_obmocja_TM.shp')
```

In [6]:
```
obmocja.head(6)
```

Out[6]:

	OBM_IME	geometry
0	LENART	POLYGON ((569631.33 150483.7, 559631.439999999...
1	KRAS	POLYGON ((420198.18 68145.00999999999, 410198...
2	BARJE	POLYGON ((468135.22 87345.17, 458135.21 87345...
3	BOHINJ	POLYGON ((423723.02 124388.2, 413722.73 124388...
4	GORICKO	POLYGON ((586192.9300000001 177889.19, 576193...
5	BREZICE	POLYGON ((548636.73 80855.83, 538636.76 80856...

In [7]:
```
obmocja_utm = obmocja.to_crs(epsg='32633')
```

In [8]:
```
obmocja_utm.head(6)
```

Out[8]:

	OBM_IME	geometry
0	LENART	POLYGON ((569610.4385112505 5148938.400481003,...
1	KRAS	POLYGON ((420222.1229409681 5066624.414557803,...
2	BARJE	POLYGON ((468144.7803903104 5085818.813933894,...
3	BOHINJ	POLYGON ((423745.9053831937 5122850.72991376, ...
4	GORICKO	POLYGON ((586167.0695342036 5176335.668011942,...
5	BREZICE	POLYGON ((548622.1375213418 5079331.420930542,...

```
In [9]: brezice_obmocje = obmocja_utm['geometry'].values[5]

In [10]: type(brezice_obmocje)

Out[10]: shapely.geometry.polygon.Polygon

In [11]: brezice_obmocje.bounds

Out[11]: (538625.1678124554, 5079331.420930542, 548622.3374613344, 5089328.65056158)

In [12]: bbox_brezice = BBox((538625, 5079330, 548625, 5089330), crs=CRS.UTM_33N)

In [13]: INSTANCE_ID='e50ce20d-d6ba-41b4-a4f1-67044a7ccfde'

In [14]: input_task = S2L2AWCSInput('BANDS-S2-L2A', resx='10m', resy='10m', maxcc=0.8, instance_

In [15]: brezice2017 = input_task.execute(bbox=bbox_brezice, time_interval=('2017-01-01','2017-1

C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\sentinelhub\data_request
  category=SHDeprecationWarning)


In [16]: brezice2017

Out[16]: EOPatch(
           data: {
             BANDS-S2-L2A: numpy.ndarray(shape=(73, 1000, 1000, 12), dtype=float32)
           }
           mask: {
             IS_DATA: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=bool)
           }
           scalar: {}
           label: {}
           vector: {}
           data_timeless: {}
           mask_timeless: {}
           scalar_timeless: {}
           label_timeless: {}
           vector_timeless: {}
           meta_info: {
             maxcc: 0.8
             service_type: 'wcs'
             size_x: '10m'
             size_y: '10m'
             time_difference: datetime.timedelta(days=-1, seconds=86399)
             time_interval: ('2017-01-01', '2017-12-31')
           }
           bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
           timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
         )
```

```
In [17]: brezice2017.timestamp
```

```
Out[17]: [datetime.datetime(2017, 1, 1, 10, 4, 7),
          datetime.datetime(2017, 1, 11, 10, 3, 51),
          datetime.datetime(2017, 1, 28, 9, 55, 57),
          datetime.datetime(2017, 2, 17, 9, 50, 44),
          datetime.datetime(2017, 2, 20, 10, 6, 35),
          datetime.datetime(2017, 2, 27, 9, 56, 13),
          datetime.datetime(2017, 3, 9, 9, 50, 21),
          datetime.datetime(2017, 3, 12, 10, 7, 6),
          datetime.datetime(2017, 3, 22, 10, 0, 19),
          datetime.datetime(2017, 3, 29, 9, 50, 24),
          datetime.datetime(2017, 4, 1, 10, 0, 22),
          datetime.datetime(2017, 4, 8, 9, 57, 11),
          datetime.datetime(2017, 4, 11, 10, 0, 25),
          datetime.datetime(2017, 4, 21, 10, 5, 41),
          datetime.datetime(2017, 5, 1, 10, 0, 29),
          datetime.datetime(2017, 5, 8, 9, 56, 51),
          datetime.datetime(2017, 5, 18, 9, 57, 16),
          datetime.datetime(2017, 5, 21, 10, 0, 29),
          datetime.datetime(2017, 5, 28, 9, 55, 31),
          datetime.datetime(2017, 5, 31, 10, 5, 36),
          datetime.datetime(2017, 6, 17, 9, 55, 46),
          datetime.datetime(2017, 6, 20, 10, 4, 53),
          datetime.datetime(2017, 7, 2, 9, 57, 13),
          datetime.datetime(2017, 7, 5, 10, 0, 26),
          datetime.datetime(2017, 7, 7, 9, 52, 57),
          datetime.datetime(2017, 7, 10, 10, 5, 40),
          datetime.datetime(2017, 7, 12, 9, 56, 23),
          datetime.datetime(2017, 7, 15, 10, 0, 26),
          datetime.datetime(2017, 7, 17, 9, 56, 31),
          datetime.datetime(2017, 7, 20, 10, 0, 27),
          datetime.datetime(2017, 7, 22, 9, 55, 21),
          datetime.datetime(2017, 7, 25, 10, 5, 36),
          datetime.datetime(2017, 7, 27, 9, 54, 29),
          datetime.datetime(2017, 7, 30, 10, 5, 35),
          datetime.datetime(2017, 8, 1, 9, 57, 12),
          datetime.datetime(2017, 8, 4, 10, 6, 8),
          datetime.datetime(2017, 8, 6, 9, 57, 44),
          datetime.datetime(2017, 8, 9, 10, 0, 28),
          datetime.datetime(2017, 8, 11, 9, 55, 12),
          datetime.datetime(2017, 8, 16, 9, 57, 37),
          datetime.datetime(2017, 8, 19, 10, 7, 22),
          datetime.datetime(2017, 8, 21, 9, 54, 53),
          datetime.datetime(2017, 8, 24, 10, 0, 22),
          datetime.datetime(2017, 8, 26, 9, 50, 30),
          datetime.datetime(2017, 8, 29, 10, 0, 26),
          datetime.datetime(2017, 8, 31, 9, 50, 23),
```

```
        datetime.datetime(2017, 9, 5, 9, 50, 28),
        datetime.datetime(2017, 9, 10, 9, 50, 21),
        datetime.datetime(2017, 9, 18, 10, 0, 23),
        datetime.datetime(2017, 9, 23, 10, 5, 2),
        datetime.datetime(2017, 9, 28, 10, 6, 17),
        datetime.datetime(2017, 9, 30, 9, 50, 13),
        datetime.datetime(2017, 10, 5, 9, 50, 27),
        datetime.datetime(2017, 10, 8, 10, 3, 22),
        datetime.datetime(2017, 10, 13, 10, 0, 12),
        datetime.datetime(2017, 10, 15, 9, 53, 57),
        datetime.datetime(2017, 10, 18, 10, 2),
        datetime.datetime(2017, 10, 20, 9, 52, 4),
        datetime.datetime(2017, 10, 25, 9, 50, 57),
        datetime.datetime(2017, 10, 30, 9, 55, 29),
        datetime.datetime(2017, 11, 4, 9, 53, 20),
        datetime.datetime(2017, 11, 12, 10, 2, 29),
        datetime.datetime(2017, 11, 22, 10, 6, 14),
        datetime.datetime(2017, 11, 24, 9, 53, 50),
        datetime.datetime(2017, 11, 27, 10, 3, 39),
        datetime.datetime(2017, 12, 2, 10, 6, 47),
        datetime.datetime(2017, 12, 7, 10, 7, 25),
        datetime.datetime(2017, 12, 14, 9, 57, 19),
        datetime.datetime(2017, 12, 17, 10, 5, 40),
        datetime.datetime(2017, 12, 19, 9, 54, 10),
        datetime.datetime(2017, 12, 22, 10, 4, 15),
        datetime.datetime(2017, 12, 24, 9, 56, 27),
        datetime.datetime(2017, 12, 29, 9, 57, 10)]
```

In [ ]:
```
Sen2Cors scene classification (SCL) contains 11 classes with the following values and me
    1 - SC_SATURATED_DEFECTIVE

    2 - SC_DARK_FEATURE_SHADOW

    3 - SC_CLOUD_SHADOW

    4 - VEGETATION

    5 - NOT-VEGETATED

    6 - WATER

    7 - SC_CLOUD_LOW_PROBA / UNCLASSIFIED

    8 - SC_CLOUD_MEDIUM_PROBA

    9 - CLOUD_HIGH_PROBABILITY

    10 - THIN_CIRRUS
```

```
        11 - SNOW
```

In [18]:
```
# 2. Add Sen2Cor's scene classification map
add_SCL = AddSen2CorClassificationFeature(sen2cor_classification='SCL',layer='BANDS-S2-
```

In [19]:
```
brezice = add_SCL.execute(brezice2017)
```

In [20]:
```
brezice2017
```

Out[20]:
```
EOPatch(
  data: {
    BANDS-S2-L2A: numpy.ndarray(shape=(73, 1000, 1000, 12), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=int32)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {}
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {}
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
)
```

In [21]:
```
class Sen2CorValidData:
    """
    Combine Sen2Cor's classification map with `IS_DATA` to define a `VALID_DATA_S2C` ma

    The Sen2Cor's classification map is asumed to be found in eopatch.mask['SCL']
    """
    def __call__(self, eopatch):
        sen2cor_valid = np.zeros_like(eopatch.mask['SCL'], dtype=np.bool)

        valid_classes = [2, 4, 5, 6, 7, 11]
        for valid in valid_classes:
```

```
                    sen2cor_valid = np.logical_or(sen2cor_valid, (eopatch.mask['SCL']==valid).a

                return np.logical_and(eopatch.mask['IS_DATA'].astype(np.bool), sen2cor_valid)
```

```
In [22]: add_s2c_valmask = AddValidDataMaskTask(Sen2CorValidData(), 'VALID_DATA_S2C')
```

```
In [23]: brezice = add_s2c_valmask.execute(brezice2017)
```

```
In [24]: brezice2017
```

```
Out[24]: EOPatch(
           data: {
             BANDS-S2-L2A: numpy.ndarray(shape=(73, 1000, 1000, 12), dtype=float32)
           }
           mask: {
             IS_DATA: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=bool)
             SCL: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=int32)
             VALID_DATA_S2C: numpy.ndarray(shape=(73, 1000, 1000, 1), dtype=bool)
           }
           scalar: {}
           label: {}
           vector: {}
           data_timeless: {}
           mask_timeless: {}
           scalar_timeless: {}
           label_timeless: {}
           vector_timeless: {}
           meta_info: {
             maxcc: 0.8
             service_type: 'wcs'
             size_x: '10m'
             size_y: '10m'
             time_difference: datetime.timedelta(days=-1, seconds=86399)
             time_interval: ('2017-01-01', '2017-12-31')
           }
           bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
           timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
         )
```

```
In [25]: from persen.tasks import SentinelHubValidData, Sen2CorValidData, MergeMasks, ValidDataF
```

```
In [26]: valid_data_predicate = ValidDataFractionPredicate(0.85)
         filter_task = SimpleFilterTask((FeatureType.MASK, 'VALID_DATA_S2C'), valid_data_predica
```

```
In [28]: filter_task.execute(brezice2017)
```

```
Out[28]: EOPatch(
           data: {
             BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
```

```
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {}
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {}
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
)
```

In [29]:
```
brezice2017.timestamp
```

Out[29]:
```
[datetime.datetime(2017, 1, 1, 10, 4, 7),
 datetime.datetime(2017, 2, 17, 9, 50, 44),
 datetime.datetime(2017, 2, 20, 10, 6, 35),
 datetime.datetime(2017, 3, 29, 9, 50, 24),
 datetime.datetime(2017, 4, 1, 10, 0, 22),
 datetime.datetime(2017, 4, 8, 9, 57, 11),
 datetime.datetime(2017, 4, 21, 10, 5, 41),
 datetime.datetime(2017, 5, 18, 9, 57, 16),
 datetime.datetime(2017, 5, 28, 9, 55, 31),
 datetime.datetime(2017, 6, 17, 9, 55, 46),
 datetime.datetime(2017, 6, 20, 10, 4, 53),
 datetime.datetime(2017, 7, 2, 9, 57, 13),
 datetime.datetime(2017, 7, 5, 10, 0, 26),
 datetime.datetime(2017, 7, 7, 9, 52, 57),
 datetime.datetime(2017, 7, 10, 10, 5, 40),
 datetime.datetime(2017, 7, 12, 9, 56, 23),
 datetime.datetime(2017, 7, 17, 9, 56, 31),
 datetime.datetime(2017, 7, 20, 10, 0, 27),
 datetime.datetime(2017, 7, 25, 10, 5, 36),
 datetime.datetime(2017, 7, 30, 10, 5, 35),
```

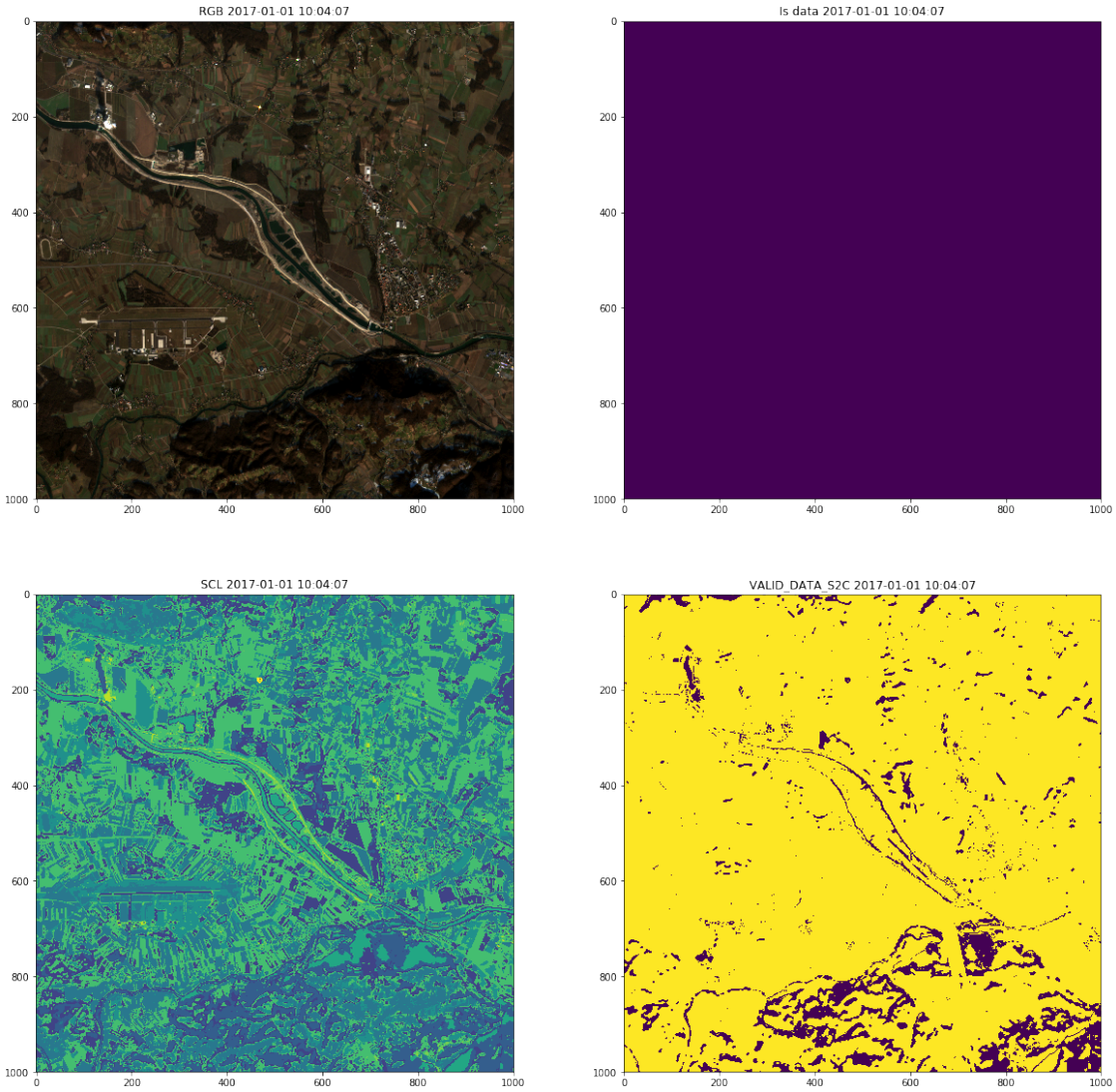
```
            datetime.datetime(2017, 8, 1, 9, 57, 12),
            datetime.datetime(2017, 8, 4, 10, 6, 8),
            datetime.datetime(2017, 8, 9, 10, 0, 28),
            datetime.datetime(2017, 8, 11, 9, 55, 12),
            datetime.datetime(2017, 8, 21, 9, 54, 53),
            datetime.datetime(2017, 8, 24, 10, 0, 22),
            datetime.datetime(2017, 8, 26, 9, 50, 30),
            datetime.datetime(2017, 8, 29, 10, 0, 26),
            datetime.datetime(2017, 8, 31, 9, 50, 23),
            datetime.datetime(2017, 9, 5, 9, 50, 28),
            datetime.datetime(2017, 9, 18, 10, 0, 23),
            datetime.datetime(2017, 9, 23, 10, 5, 2),
            datetime.datetime(2017, 9, 30, 9, 50, 13),
            datetime.datetime(2017, 10, 5, 9, 50, 27),
            datetime.datetime(2017, 10, 13, 10, 0, 12),
            datetime.datetime(2017, 10, 20, 9, 52, 4),
            datetime.datetime(2017, 10, 25, 9, 50, 57),
            datetime.datetime(2017, 11, 24, 9, 53, 50),
            datetime.datetime(2017, 11, 27, 10, 3, 39),
            datetime.datetime(2017, 12, 7, 10, 7, 25),
            datetime.datetime(2017, 12, 24, 9, 56, 27)]
```

In [32]:
```
def plot_frame(patch, idx):
    fig, axs = plt.subplots(nrows=2, ncols=2, figsize=(20,20))

    axs[0,0].imshow(patch.data['BANDS-S2-L2A'][idx][...,[3,2,1]]*2.5)
    axs[0,0].set_title(f'RGB {patch.timestamp[idx]}')
    axs[0,1].imshow(patch.mask['IS_DATA'][idx, ..., 0])
    axs[0,1].set_title(f'Is data {patch.timestamp[idx]}')
    axs[1,0].imshow(patch.mask['SCL'][idx, ..., 0])
    axs[1,0].set_title(f'SCL {patch.timestamp[idx]}')
    axs[1,1].imshow(patch.mask['VALID_DATA_S2C'][idx, ..., 0])
    axs[1,1].set_title(f'VALID_DATA_S2C {patch.timestamp[idx]}')
```

In [33]:
```
plot_frame(brezice2017, 0)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] f
```




```
In [35]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Normalised Difference Index (OSAVI) between two b
             OSAVI = 1.5*((A-B)/(A+B+0,16).
             """
             def __init__(self, in_feature_name):
                 self.in_feature_name = in_feature_name

             def execute(self, eopatch):
                 NIR = eopatch.data[self.in_feature_name][..., 8]
                 RED = eopatch.data[self.in_feature_name][..., 4]

                 OSAVI = 1.5*((NIR - RED) / (NIR + RED +0.16))
```

```
                eopatch.add_feature(FeatureType.DATA, "OSAVI", OSAVI[..., np.newaxis])

                return eopatch
```

```
In [36]: osavitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [37]: osavitask.execute(brezice2017)
```

```
Out[37]: EOPatch(
           data: {
             BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
             OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
           }
           mask: {
             IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
             SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
             VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
           }
           scalar: {}
           label: {}
           vector: {}
           data_timeless: {}
           mask_timeless: {}
           scalar_timeless: {}
           label_timeless: {}
           vector_timeless: {}
           meta_info: {
             maxcc: 0.8
             service_type: 'wcs'
             size_x: '10m'
             size_y: '10m'
             time_difference: datetime.timedelta(days=-1, seconds=86399)
             time_interval: ('2017-01-01', '2017-12-31')
           }
           bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
           timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
         )
```

```
In [38]: brezicex = osavitask.execute(brezice2017)
```

```
In [39]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Normalised Difference Index (NDVI) between two ba
             NDVI = (A-B)/(A+B).
             """
             def __init__(self, in_feature_name):
                 self.in_feature_name = in_feature_name

             def execute(self, eopatch):
```

```
                NIR = eopatch.data[self.in_feature_name][..., 8]
                RED = eopatch.data[self.in_feature_name][..., 4]

                NDVI = (NIR - RED) / (NIR + RED)

                eopatch.add_feature(FeatureType.DATA, "NDVI", NDVI[..., np.newaxis])

                return eopatch
```

```
In [40]: ndvitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [41]: brezicex = ndvitask.execute(brezice2017)
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:13
  del sys.path[0]
```

```
In [42]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Normalised Difference Index (MCARI705) between 3
             MCARI = ((AB)0.2(AC))*(A/B)
             """
             def __init__(self, in_feature_name):
                 self.in_feature_name = in_feature_name

             def execute(self, eopatch):
                 RE740 = eopatch.data[self.in_feature_name][..., 6]
                 RE705 = eopatch.data[self.in_feature_name][..., 5]
                 GREEN = eopatch.data[self.in_feature_name][..., 3]

                 MCARI = (RE740 - RE705) - (0.2 * (RE740 - GREEN) * (RE740 + RE705))

                 eopatch.add_feature(FeatureType.DATA, "MCARI", MCARI[..., np.newaxis])

                 return eopatch
```

```
In [44]: mcaritask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [45]: brezicex = mcaritask.execute(brezice2017)
```

```
In [46]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Green Normalised Difference Index (GNDVI) between
            GNDVI = (AB)/(A+B)
             """
             def __init__(self, in_feature_name):
                 self.in_feature_name = in_feature_name

             def execute(self, eopatch):
```

```
        NIR = eopatch.data[self.in_feature_name][..., 8]
        GREEN = eopatch.data[self.in_feature_name][..., 3]

        GNDVI= (NIR - GREEN) / (NIR + GREEN)

        eopatch.add_feature(FeatureType.DATA, "GNDVI", GNDVI[..., np.newaxis])

        return eopatch
```

In [47]:
```
gndvitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

In [48]:
```
brezicex = gndvitask.execute(brezice2017)
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:13
  del sys.path[0]
```

In [49]:
```
class NormalizedDifferenceIndex(EOTask):
    """
    The tasks calculates user defined Normalised Difference Index (OSAVI) between two b
    GOSAVI = ((A-B)/(A+B+0,16).
    """
    def __init__(self, in_feature_name):
        self.in_feature_name = in_feature_name

    def execute(self, eopatch):
        NIR = eopatch.data[self.in_feature_name][..., 8]
        GREEN = eopatch.data[self.in_feature_name][..., 3]

        GOSAVI = 1.5*((NIR - GREEN) / (NIR + GREEN + 0.16))

        eopatch.add_feature(FeatureType.DATA, "GOSAVI", GOSAVI[..., np.newaxis])

        return eopatch
```

In [50]:
```
gosavitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

In [52]:
```
brezicex = gosavitask.execute(brezice2017)
```

In [53]:
```
class NormalizedDifferenceIndex(EOTask):
    """
    The tasks calculates user defined Normalised Difference Index (PSSRa) between two b
    PSSRa = A/B .
    """
    def __init__(self, in_feature_name):
        self.in_feature_name = in_feature_name

    def execute(self, eopatch):
        NIR = eopatch.data[self.in_feature_name][..., 8]
```

```
                RED = eopatch.data[self.in_feature_name][..., 4]

                PSSRA = NIR / RED

                eopatch.add_feature(FeatureType.DATA, "PSSRA", PSSRA[..., np.newaxis])

                return eopatch
```

```
In [54]: pssratask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [56]: brezicex = pssratask.execute(brezice2017)
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:13
  del sys.path[0]
```

```
In [57]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Normalised Difference Index (TCARI) between 3 ban
             3((700nm670nm)0.2(700nm550nm)(700nm670nm))
             TCARI = (AB)0.2*(AC))*(A/B)
             """
             def __init__(self, in_feature_name):
                 self.in_feature_name = in_feature_name

             def execute(self, eopatch):
                 RE705 = eopatch.data[self.in_feature_name][..., 5]
                 RED   = eopatch.data[self.in_feature_name][..., 4]
                 GREEN = eopatch.data[self.in_feature_name][..., 3]

                 TCARI = (RE705 - RED) - (0.2 * (RE705 - GREEN) * (RE705 / RED))

                 eopatch.add_feature(FeatureType.DATA, "TCARI", TCARI[..., np.newaxis])

                 return eopatch
```

```
In [58]: tcaritask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [59]: brezicex = tcaritask.execute(brezice2017)
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:15
  from ipykernel import kernelapp as app
```

```
In [60]: class NormalizedDifferenceIndex(EOTask):
             """
             The tasks calculates user defined Triangular Vegetation Index (TVI) between 3 bands
             0.5(120(750nm550nm)200(670nm550nm))
             TVI = 0.5*(120*(AC)200*(BC))
```

```
            """
            def __init__(self, in_feature_name):
                self.in_feature_name = in_feature_name

            def execute(self, eopatch):
                RE740 = eopatch.data[self.in_feature_name][..., 6]
                RED   = eopatch.data[self.in_feature_name][..., 4]
                GREEN = eopatch.data[self.in_feature_name][..., 3]

                TVI = 0.5 * (120 * (RE740 - GREEN) - 200 * (RED - GREEN))

                eopatch.add_feature(FeatureType.DATA, "TVI", TVI[..., np.newaxis])

                return eopatch
```

```
In [61]: tvitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [62]: brezicex = tvitask.execute(brezice2017)
```

```
In [63]: class NormalizedDifferenceIndex(EOTask):
            """
            The tasks calculates user  Green Leaf Index (GLI) between 3 bands A , B  and C as:
            2GREENREDBLUE/2GREEN+RED+BLUE
            GLI = ((2*B)A-C)/((2*B)+A+C)
            """
            def __init__(self, in_feature_name):
                self.in_feature_name = in_feature_name

            def execute(self, eopatch):
                RED   = eopatch.data[self.in_feature_name][..., 4]
                GREEN = eopatch.data[self.in_feature_name][..., 3]
                BLUE = eopatch.data[self.in_feature_name][..., 2]

                GLI = ((2 * GREEN) - RED - BLUE) /((2 * GREEN + RED + BLUE))

                eopatch.add_feature(FeatureType.DATA, "GLI", GLI[..., np.newaxis])

                return eopatch
```

```
In [64]: glitask = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [65]: brezicex = glitask.execute(brezice2017)
```

```
C:\Users\matejk.KIS\AppData\Local\Continuum\anaconda3\lib\site-packages\ipykernel_launcher.py:15
  from ipykernel import kernelapp as app
```

```
In [66]: class NormalizedDifferenceIndex(EOTask):
            """
```

```
            The tasks calculates user          Modified Triangular Vegetation Index 1 (MTVI1) b
                   1.2(1.2(800nm550nm)2.5(670nm550nm))
            MTVI1 = 1.2 * (1.2 * (A-B) 2.5 * (C-B))
            """
            def __init__(self, in_feature_name):
                self.in_feature_name = in_feature_name

            def execute(self, eopatch):
                NIR   = eopatch.data[self.in_feature_name][..., 8]
                RED = eopatch.data[self.in_feature_name][..., 4]
                GREEN = eopatch.data[self.in_feature_name][..., 3]

                MTVI1= 1.2 * (1.2 * (NIR - GREEN) - 2.5 * (RED - GREEN))

                eopatch.add_feature(FeatureType.DATA, "MTVI1", MTVI1[..., np.newaxis])

                return eopatch
```

```
In [67]: mtvi1task = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [68]: brezicex = mtvi1task.execute(brezice2017)
```

```
In [69]: class NormalizedDifferenceIndex(EOTask):
            """
            The tasks calculates user          Ehanced Vegetation Index 2 (EVI2) between 3 band
            2.4 * (NIR - RED) / (NIR + RED + 1)
            EVI =  2.4 * (B08 - B04) / (B08 + B04 + 1.0)
            """
            def __init__(self, in_feature_name):
                self.in_feature_name = in_feature_name

            def execute(self, eopatch):
                NIR   = eopatch.data[self.in_feature_name][..., 8]
                RED = eopatch.data[self.in_feature_name][..., 4]

                EVI2= 2.4 * (NIR + RED) / (NIR + RED + 1)

                eopatch.add_feature(FeatureType.DATA, "EVI2", EVI2[..., np.newaxis])

                return eopatch
```

```
In [70]: evi2task = NormalizedDifferenceIndex("BANDS-S2-L2A")
```

```
In [81]: brezicex = evi2task.execute(brezice2017)
```

```
In [82]: brezice2017
```

```
Out[82]: EOPatch(
           data: {
```

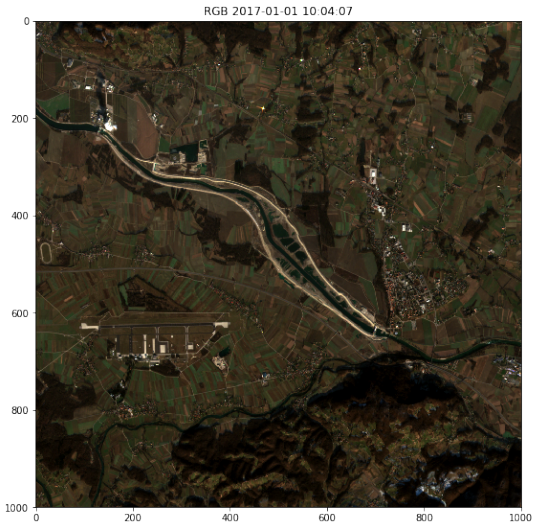
```
    BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
    EVI2: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GLI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GNDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GOSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MTVI1: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    NDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    PSSRA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {}
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {}
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12,
)
```

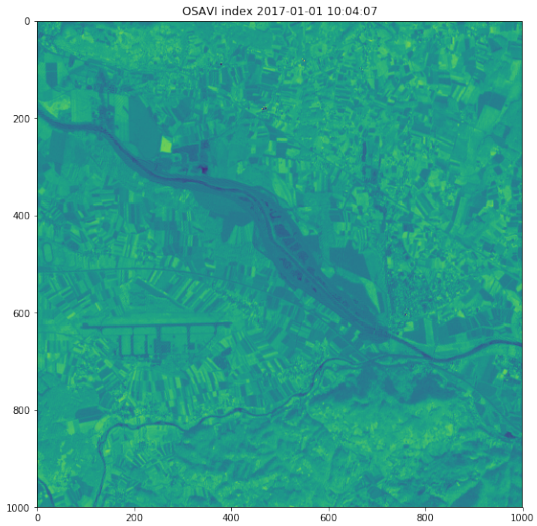
```
In [83]: def plot_savi(patch, idx):
             fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(20,20))

             axs[0].imshow(patch.data['BANDS-S2-L2A'][idx][...,[3,2,1]]*2.5)
             axs[0].set_title(f'RGB {patch.timestamp[idx]}')
             axs[1].imshow(patch.data['OSAVI'][idx, ..., 0])
             axs[1].set_title(f'OSAVI index {patch.timestamp[idx]}')
```

```
In [84]: plot_savi(brezice2017,0)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] f
```






```
In [85]: subvencije2017_brezice = geopandas.read_file('E:/2015/V4-1811/BREZICE/2017/brezice_subv

In [86]: subvencije2017_brezice.info()

<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 5889 entries, 0 to 5888
Data columns (total 86 columns):
FID_CRP_mo    5889 non-null int64
OBM_IME       5889 non-null object
FID_ZV_pri    5889 non-null int64
KMG_MID       5889 non-null int64
BLOK_ID       5889 non-null int64
GERK_PID      5889 non-null int64
RABA_ID       5889 non-null int64
GERK_POVR_    5889 non-null float64
POLJINA_ID    5889 non-null float64
POLJ_POVR_    5889 non-null float64
POSEVEK       5889 non-null object
SIFRA_KMRS    5889 non-null object
NAZIV_ZELE    67 non-null object
AKT_AR        5889 non-null float64
BR_AR         5889 non-null float64
DOD_NR_AR     5889 non-null float64
EK_AR         5889 non-null float64
EKSEME_AR     5889 non-null float64
GEN_SEME_A    5889 non-null float64
GEN_SOR_AR    5889 non-null float64
HAB_KOS_AR    5889 non-null float64
```

```
HAB_MRVA_A    5889 non-null float64
HAB_NPAS_A    5889 non-null float64
HAB_ORGG_A    5889 non-null float64
HML_BIOV_A    5889 non-null float64
HML_KOMP_A    5889 non-null float64
HML_NIZI_A    5889 non-null float64
HML_NMIN_A    5889 non-null float64
HML_POKT_A    5889 non-null float64
KRA_CRED_A    5889 non-null float64
KRA_GRB_AR    5889 non-null float64
KRA_OGRM_A    5889 non-null float64
KRA_PAST_A    5889 non-null float64
KRA_S5O_AR    5889 non-null float64
KRA_VARPA_    5889 non-null float64
KRA_VARPP_    5889 non-null float64
KRA_VTSA_A    5889 non-null float64
MET_KOS_AR    5889 non-null float64
MET_MRVA_A    5889 non-null float64
MET_NPAS_A    5889 non-null float64
OMD_AR        5889 non-null float64
OOTT_AR       5889 non-null float64
PEP_AR        5889 non-null float64
PONO35_AR     5889 non-null float64
PONO5O_AR     5889 non-null float64
POZ_FFSM_A    5889 non-null float64
POZ_FFSV_A    5889 non-null float64
POZ_KOL_AR    5889 non-null float64
POZ_KONZ_A    5889 non-null float64
POZ_MEHZ_A    5889 non-null float64
POZ_NEP_AR    5889 non-null float64
POZ_NIZI_A    5889 non-null float64
POZ_NMIN_A    5889 non-null float64
POZ_POD_AR    5889 non-null float64
POZ_ZEL_AR    5889 non-null float64
SAD_EKGN_A    5889 non-null float64
SAD_KONF_A    5889 non-null float64
SAD_MEHZ_A    5889 non-null float64
SAD_POKT_A    5889 non-null float64
SAD_VABE_A    5889 non-null float64
STE_KOS_AR    5889 non-null float64
STE_NPAS_A    5889 non-null float64
SÄźËİ_AR       5889 non-null float64
TRZ_II_NIZ    5889 non-null float64
TRZ_II_NPA    5889 non-null float64
TRZ_II_OSI    5889 non-null float64
TRZ_I_MRVA    5889 non-null float64
TRZ_I_NIZI    5889 non-null float64
TRZ_I_NPAS    5889 non-null float64
```

```
TRZ_I_OSIL    5889 non-null float64
VIN_EKGN_A    5889 non-null float64
VIN_INSK_A    5889 non-null float64
VIN_MEDV_A    5889 non-null float64
VIN_MEHZ_A    5889 non-null float64
VIN_POKT_A    5889 non-null float64
VIN_VABE_A    5889 non-null float64
VOD_FFSV_A    5889 non-null float64
VOD_NEP_AR    5889 non-null float64
VOD_POD_AR    5889 non-null float64
VOD_ZEL_AR    5889 non-null float64
VTR_KOS_AR    5889 non-null float64
VTR_NPAS_A    5889 non-null float64
ZL_AR         5889 non-null float64
BUFF_DIST     5889 non-null float64
ORIG_FID      5889 non-null int64
geometry      5889 non-null geometry
dtypes: float64(74), geometry(1), int64(7), object(4)
memory usage: 3.9+ MB
```

```
In [87]: len(subvencije2017_brezice)
```

```
Out[87]: 5889
```

filtriraj geometrije in obdri samo tiste, ki se sekajo z bboxom od brezice eopatcha

```
In [112]: brezice_subvencije = subvencije2017_brezice[subvencije2017_brezice.geometry.intersects
```

```
In [113]: class AddVectorFeature(EOTask):
              """
              Add vector data from dataframe.
              """
              def __init__(self, feature):
                  self.feature_type, self.feature_name = next(self._parse_features(feature)())

              def execute(self, eopatch, dataframe):
                  eopatch[self.feature_type][self.feature_name] = dataframe

                  return eopatch
```

```
In [114]: add_subvencije = AddVectorFeature(feature=(FeatureType.VECTOR_TIMELESS, 'subvencijebre
```

```
In [115]: brezice2017 = add_subvencije.execute(brezice2017, dataframe=subvencije2017_brezice)
```

```
In [116]: brezice2017
```

```
Out[116]: EOPatch(
            data: {
```

```
    BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
    EVI2: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GLI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GNDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GOSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MTVI1: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    NDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    PSSRA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {}
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {
    subvencijebrezice2017: geopandas.GeoDataFrame(columns=['FID_CRP_mo', 'OBM_IME', 'F
  }
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12
)
```

In [117]:
```
from eolearn.geometry import VectorToRaster
```

In [118]:
```
subvencijebrezice2017= subvencije2017_brezice[(subvencije2017_brezice.OBM_IME=='BREZIC
```

In [119]:
```
add_subraster = VectorToRaster(subvencije2017_brezice, (FeatureType.MASK_TIMELESS,'sub
```

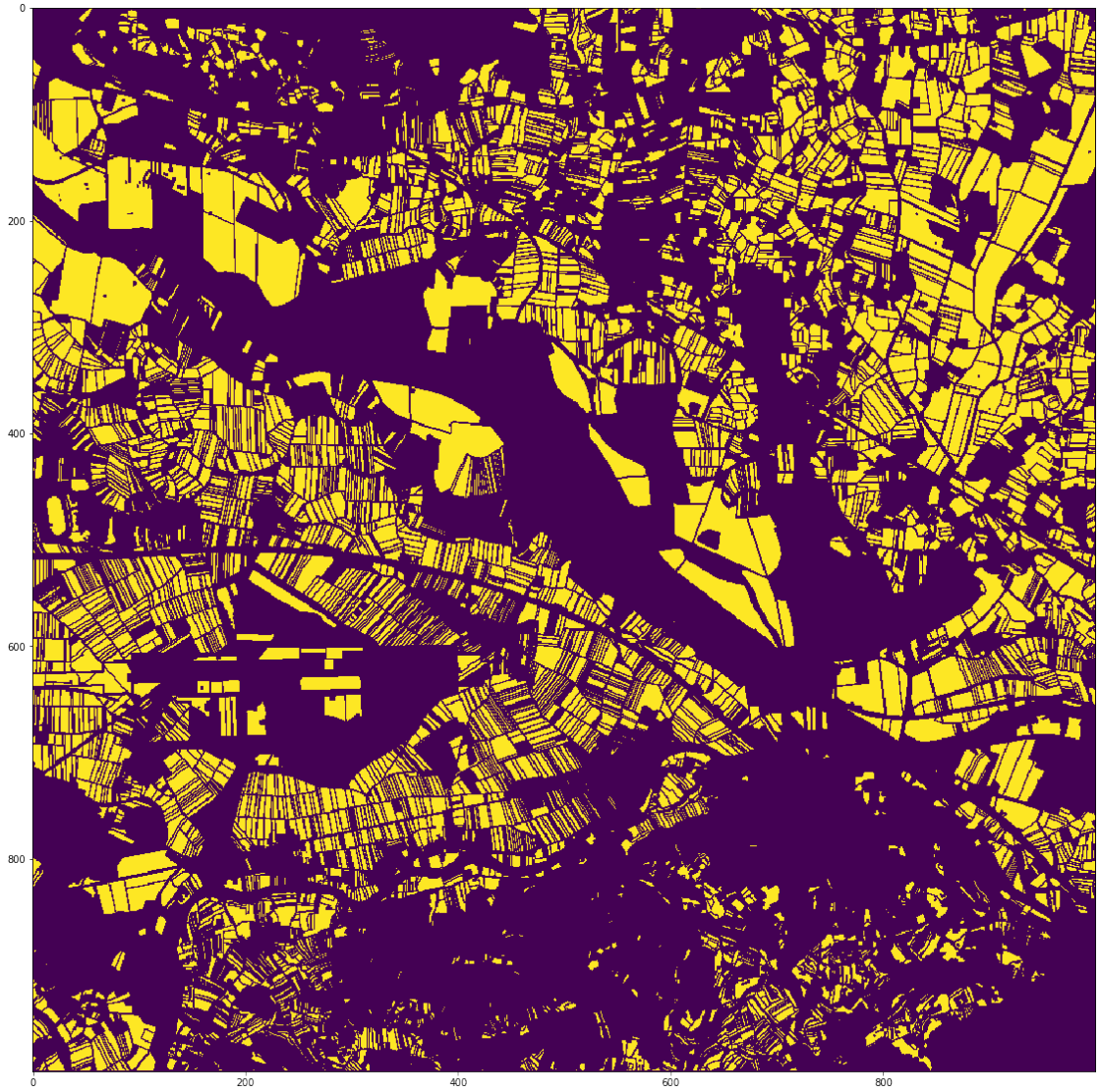
In [120]:
```
add_subraster.execute(brezice2017)
```

Out[120]:
```
EOPatch(
  data: {
```

```
    BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
    EVI2: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GLI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GNDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GOSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MTVI1: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    NDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    PSSRA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {
    subvencijebrezice2017: numpy.ndarray(shape=(1000, 1000, 1), dtype=uint8)
  }
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {
    subvencijebrezice2017: geopandas.GeoDataFrame(columns=['FID_CRP_mo', 'OBM_IME', 'F
  }
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12
)
```

In [122]:
```
plt.figure(figsize=(20,20))
plt.imshow(brezice2017.mask_timeless['subvencijebrezice2017'][...,0])
```

Out[122]:
```
<matplotlib.image.AxesImage at 0x29247668>
```




```
In [125]: # features indeksov
          indeksi = [
              (FeatureType.DATA, 'EVI2'),
              (FeatureType.DATA, 'GLI'),
              (FeatureType.DATA, 'GNDVI'),
              (FeatureType.DATA, 'GOSAVI'),
              (FeatureType.DATA, 'MCARI'),
              (FeatureType.DATA, 'MTVI1'),
              (FeatureType.DATA, 'NDVI'),
              (FeatureType.DATA, 'OSAVI'),
              (FeatureType.DATA, 'PSSRA'),
              (FeatureType.DATA, 'TCARI'),
              (FeatureType.DATA, 'TVI')
          ]
```

In [126]:
```
# vse indekse se zdruzi v en feature, zato da export task lazje to zapise

# task za zdruzitev indeksov
class ConcatenateTask(EOTask):

    def __init__(self, new_feature, features):
        self.new_feature = new_feature
        self.features = features

    def execute(self, eopatch):

        eopatch[self.new_feature] = np.concatenate([eopatch[feature] for feature in se
        return eopatch

concatenate_task = ConcatenateTask((FeatureType.DATA, 'INDEKSI'), indeksi)
concatenate_task.execute(brezice2017)
```

Out[126]:
```
EOPatch(
  data: {
    BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
    EVI2: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GLI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GNDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GOSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    INDEKSI: numpy.ndarray(shape=(41, 1000, 1000, 11), dtype=float32)
    MCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MTVI1: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    NDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    PSSRA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {
    subvencijebrezice2017: numpy.ndarray(shape=(1000, 1000, 1), dtype=uint8)
  }
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {
```

```
    subvencijebrezice2017: geopandas.GeoDataFrame(columns=['FID_CRP_mo', 'OBM_IME', 'F
  }
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12
)
```

In [127]:
```
#izracun preseka s subvencijami

# task za presek indeksov
class PresekTask(EOTask):

    def __init__(self, mask_feature, data_feature):
        self.mask_feature = mask_feature
        self.data_feature = data_feature

    def execute(self, eopatch):

        mask = eopatch[self.mask_feature].squeeze() == 1

        for idx in range(len(eopatch.timestamp)):
            eopatch[self.data_feature][idx, ~mask, ...] = -1 # nova vrednost izven pre

        return eopatch

presek_task = PresekTask((FeatureType.MASK_TIMELESS, 'subvencijebrezice2017'), (Featur
presek_task.execute(brezice2017)
```

Out[127]:
```
EOPatch(
  data: {
    BANDS-S2-L2A: numpy.ndarray(shape=(41, 1000, 1000, 12), dtype=float32)
    EVI2: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GLI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GNDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    GOSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    INDEKSI: numpy.ndarray(shape=(41, 1000, 1000, 11), dtype=float32)
    MCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    MTVI1: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    NDVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    OSAVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    PSSRA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
```

```
    TCARI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
    TVI: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=float32)
  }
  mask: {
    IS_DATA: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
    SCL: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=int32)
    VALID_DATA_S2C: numpy.ndarray(shape=(41, 1000, 1000, 1), dtype=bool)
  }
  scalar: {}
  label: {}
  vector: {}
  data_timeless: {}
  mask_timeless: {
    subvencijebrezice2017: numpy.ndarray(shape=(1000, 1000, 1), dtype=uint8)
  }
  scalar_timeless: {}
  label_timeless: {}
  vector_timeless: {
    subvencijebrezice2017: geopandas.GeoDataFrame(columns=['FID_CRP_mo', 'OBM_IME', 'F
  }
  meta_info: {
    maxcc: 0.8
    service_type: 'wcs'
    size_x: '10m'
    size_y: '10m'
    time_difference: datetime.timedelta(days=-1, seconds=86399)
    time_interval: ('2017-01-01', '2017-12-31')
  }
  bbox: BBox(((538625.0, 5079330.0), (548625.0, 5089330.0)), crs=CRS('32633'))
  timestamp: [datetime.datetime(2017, 1, 1, 10, 4, 7), ..., datetime.datetime(2017, 12
)
```

In [130]:
```
# za vsak timestamp ustvari en task, ki izpise vse indekse v isti tiff

for idx, tstmp in enumerate(brezice2017.timestamp):
    task = ExportToTiff(folder=f'E:/2015/V4-1811/BREICE/2017/test/indeksi_{idx}_{str(t
                        date_indices=[idx], band_indices=list(range(11))
                       )

    task.execute(brezice2017)
```

In [ ]:

In [ ]:

In [ ]: