Copernicus Master in Digital Earth

Student: Emanuel Goulart Farias

Student number: 12413874

Digital Earth: Big Earth Data Concepts

Assignment 6: Google Earth Engine

Google Earth Engine

In my GEE application, I choose to investigate the last floods took in place in Spain, more specifically in the region of Valencia. I opt to use multispectral satellite data from Sentinel-2. Furthermore, I match the date of the event and figure out images that capture the extension of the floods without no much cloud covering in the scene. To analyse, I calculate the Normalized Difference Water Index, in order to highlight the flooded area, comparing with the median of the last month. To finish up, I publish as a Google Earth Engine application. My visual geospatial analysis can be reached at the link:

Floods in Valencia - A visual inspection powered by Sentinel-2

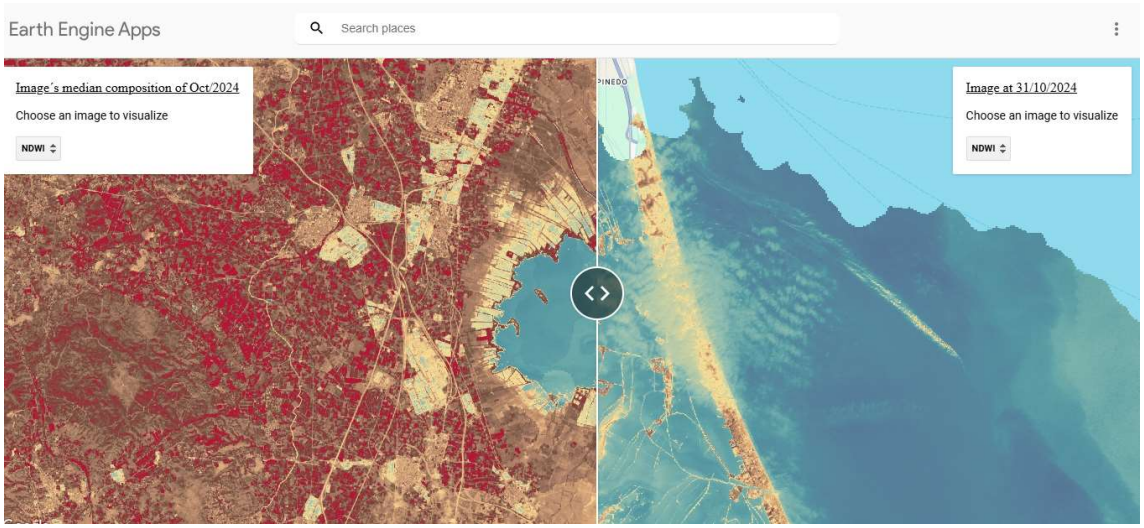

Figure 1 - Screenshot of my GEE application.

The code is compiled below:

---------- Start ----------

```
// Demonstrates before/after imagery comparison with a variety of dates.
var aoi =
    /* color: #d63000 */
    /* displayProperties: [
      {
        "type": "rectangle"
      }
    ] */
    ee.Geometry.Polygon(
        [[[-0.5009366240638458, 39.41074810267774],
          [-0.5009366240638458, 39.25619819028707],
          [-0.22902500297009576, 39.25619819028707],
          [-0.22902500297009576, 39.41074810267774]]], null, false);
print('aoi',aoi)
```

```
/*
 * Configure the imagery
 */

// ---------- ********** LEFT IMAGES **********----------
var left_images = {
  'Natural_image': left_getSentinelComposite(),
  'NDWI': left_getSentinelNDWI()
};
print(left_images)
// Composite the sentinel-2 ic for the period assigned below, compute the mean and composite
RGB
function left_getSentinelComposite() {
  // Cloud Masks
  function maskS2clouds(image) {
     var qa = image.select('QA60');

     // Bits 10 and 11 are clouds and cirrus, respectively.
     var cloudBitMask = 1 << 10;
     var cirrusBitMask = 1 << 11;

     // Both flags should be set to zero, indicating clear conditions.
     var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
        .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

     return
image.updateMask(mask).divide(10000).copyProperties(image,image.propertyNames());
    }

  // Calculate NDWI
  function indices(image){
     var ndwi = image.normalizedDifference(['B3','B8']).rename('NDWI')
     return image.addBands([ndwi]).copyProperties(image, ["system:time_start"])
  }


  // Sentinel-2 imagery collection
  var s2_rgb = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                    .filterBounds(aoi)

.filter(ee.Filter.date(ee.Date.fromYMD(2024,10,01),ee.Date.fromYMD(2024,10,30)))
                    .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',30)
                    .map(maskS2clouds)
                    .select(['B4','B3','B2'])
                    .median(); //calculates median

  // Define the visualisation parameters to display the image (change bands for different
visualisations)
  var vizParams_rgb = {
                  bands: ['B4', 'B3', 'B2'],
                  min: 0,
                  max: 0.4
                  };
```

```
  return s2_rgb.visualize(vizParams_rgb);
  }

// Composite the sentinel-2 ic for the period assigned below, compute the mean and composite
NDWI
function left_getSentinelNDWI() {
  // Cloud Masks
  function maskS2clouds(image) {
     var qa = image.select('QA60');

     // Bits 10 and 11 are clouds and cirrus, respectively.
     var cloudBitMask = 1 << 10;
     var cirrusBitMask = 1 << 11;

     // Both flags should be set to zero, indicating clear conditions.
     var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
        .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

     return
image.updateMask(mask).divide(10000).copyProperties(image,image.propertyNames());
    }

  // Calculate NDWI
  function indices(image){
     var ndwi = image.normalizedDifference(['B3','B8']).rename('NDWI')
     return image.addBands([ndwi]).copyProperties(image, ["system:time_start"])
  }


  // Sentinel-2 imagery collection
  var s2_ndwi = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                        .filterBounds(aoi)

.filter(ee.Filter.date(ee.Date.fromYMD(2024,10,01),ee.Date.fromYMD(2024,10,30)))
                        .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',60)
                        .map(maskS2clouds)
                        .map(indices)
                        .select('NDWI')
                        .median(); //calculates median

  // Define the visualisation parameters to display the image (change bands for different
visualisations)
  var vizNDWI = {
               bands:['NDWI'],
               min:-1,
               max:1,
               palette:['#af2442','#b71a3c','#bf0f35','#745644','#a77c65','#d6a379','#f9e098',
                     '#97c0b0','#7db2ac','#62a4a7','#5494a1','#56819b','#566f94','#555d8e'
                     ]
  }

  return s2_ndwi.visualize(vizNDWI);
  }

// ---------- ********** RIGHT IMAGES **********----------
```

```
var right_images ={
  'Natural_image': right_getSentinelComposite(),
  'NDWI': right_getSentinelNDWI()
}
print(right_images)
// Composite the sentinel-2 ic for the period assigned below, compute the mean and composite
RGB
function right_getSentinelComposite() {
  // Cloud Masks
  function maskS2clouds(image) {
     var qa = image.select('QA60');

     // Bits 10 and 11 are clouds and cirrus, respectively.
     var cloudBitMask = 1 << 10;
     var cirrusBitMask = 1 << 11;

     // Both flags should be set to zero, indicating clear conditions.
     var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
        .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

     return
image.updateMask(mask).divide(10000).copyProperties(image,image.propertyNames());
    }

  // Calculate NDWI
  function indices(image){
     var ndwi = image.normalizedDifference(['B3','B8']).rename('NDWI')
     return image.addBands([ndwi]).copyProperties(image, ["system:time_start"])
  }


  // Sentinel-2 imagery collection
  var s2_rgb = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                    .filterBounds(aoi)

.filter(ee.Filter.date(ee.Date.fromYMD(2024,10,30),ee.Date.fromYMD(2024,11,01)))
                    .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',60)
                    .map(maskS2clouds)
                    .select(['B4','B3','B2'])
                    .median(); //calculates median

  // Define the visualisation parameters to display the image (change bands for different
visualisations)
  var vizParams_rgb = {
                bands: ['B4', 'B3', 'B2'],
                min: 0,
                max: 0.4
                };

  return s2_rgb.visualize(vizParams_rgb);
  }

// Composite the sentinel-2 ic for the period assigned below, compute the mean and composite
NDWI
function right_getSentinelNDWI() {
```

```
 // Cloud Masks
 function maskS2clouds(image) {
    var qa = image.select('QA60');

    // Bits 10 and 11 are clouds and cirrus, respectively.
    var cloudBitMask = 1 << 10;
    var cirrusBitMask = 1 << 11;

    // Both flags should be set to zero, indicating clear conditions.
    var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
       .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

    return
image.updateMask(mask).divide(10000).copyProperties(image,image.propertyNames());
   }

 // Calculate NDWI
 function indices(image){
    var ndwi = image.normalizedDifference(['B3','B8']).rename('NDWI')
    return image.addBands([ndwi]).copyProperties(image, ["system:time_start"])
 }


 // Sentinel-2 imagery collection
 var s2_ndwi = ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                    .filterBounds(aoi)

.filter(ee.Filter.date(ee.Date.fromYMD(2024,10,30),ee.Date.fromYMD(2024,11,01)))
                    .filterMetadata('CLOUDY_PIXEL_PERCENTAGE','less_than',60)
                    .map(maskS2clouds)
                    .map(indices)
                    .select('NDWI')
                    .median(); //calculates median

 // Define the visualisation parameters to display the image (change bands for different
visualisations)
 var vizNDWI = {
            bands:['NDWI'],
            min:-1,
            max:1,
            palette:['#af2442','#b71a3c','#bf0f35','#745644','#a77c65','#d6a379','#f9e098',
                  '#97c0b0','#7db2ac','#62a4a7','#5494a1','#56819b','#566f94','#555d8e'
                  ]
 }

 return s2_ndwi.visualize(vizNDWI);
 }

/*
* Set up the maps and control widgets
*/

// Create the left map, and have it display layer 0.
var leftMap = ui.Map();
leftMap.setControlVisibility(false);
```

```
leftMap.setCenter(-0.36, 39.34, 12);
var leftSelector = left_addLayerSelector(leftMap, 1, 'top-left');

// Create the right map, and have it display layer 1.
var rightMap = ui.Map();
rightMap.setControlVisibility(false);
rightMap.setCenter(-0.36, 39.34, 12);
var rightSelector = right_addLayerSelector(rightMap, 1, 'top-right');

// Adds a layer selection widget to the given map, to allow users to change
// which image is displayed in the associated map.
function left_addLayerSelector(mapToChange, defaultValue, position) {
  var label = ui.Label('Choose an image to visualize');
  var label_data = ui.Label({
    value: 'Image´s median composition of Oct/2024',
    style:{fontFamily:'serif',textDecoration:'underline',fontSize:'16px'}

  })

  // This function changes the given map to show the selected image.
  function updateMap(selection) {
    mapToChange.layers().set(0, ui.Map.Layer(left_images[selection]));
  }

  // Configure a selection dropdown to allow the user to choose between images,
  // and set the map to update when a user makes a selection.
  var select = ui.Select({items: Object.keys(left_images), onChange: updateMap});
  select.setValue(Object.keys(left_images)[defaultValue], true);

  var controlPanel =
      ui.Panel({widgets: [label_data,label, select], style: {position: position}});

  mapToChange.add(controlPanel);
}

function right_addLayerSelector(mapToChange, defaultValue, position) {
  var label = ui.Label('Choose an image to visualize');

  var label_data = ui.Label({
    value:'Image at 31/10/2024',
    style:{fontFamily:'serif',textDecoration:'underline',fontSize:'16px'}
    })

  // This function changes the given map to show the selected image.
  function updateMap(selection) {
    mapToChange.layers().set(0, ui.Map.Layer(right_images[selection]));
  }

  // Configure a selection dropdown to allow the user to choose between images,
  // and set the map to update when a user makes a selection.
  var select = ui.Select({items: Object.keys(right_images), onChange: updateMap});
  select.setValue(Object.keys(right_images)[defaultValue], true);

  var controlPanel =
      ui.Panel({widgets: [label_data,label, select], style: {position: position}});
```

```
  mapToChange.add(controlPanel);
}

/*
* Tie everything together
*/

// Create a SplitPanel to hold the adjacent, linked maps.
var splitPanel = ui.SplitPanel({
  firstPanel: leftMap,
  secondPanel: rightMap,
  wipe: true,
  style: {stretch: 'both'}
});

// Set the SplitPanel as the only thing in the UI root.
ui.root.widgets().reset([splitPanel]);
var linker = ui.Map.Linker([leftMap, rightMap]);
leftMap.setCenter(-0.36, 39.34, 12);
```