MONTE: Microclimate Observatory Network for Terrestrial Ecosystems

FOXNet: Forest Observation and eXperimental Network

FMNet: Forest Microclimate Network

ForMON: Forest Microclimate Observatory Network

EMO: Ecosystem Microclimate Observatory

EMONet: Ecosystem Microclimate Observation Network  
MicroNet: Microclimatology Network

FM2: Forest Microclimate Monitoring Network

FM2 Project: Forest Microclimate Monitoring Project

MINet: Microclimate Investigation Network

Forest Microclimate Research Network

EMM-Net: Ecosystem Microclimate Monitoring Network

EMNet – Ecosystem Microclimate Network

Forest MicroNet

Forest Microclimate Monitoring Project

// Define the regional bounds of animation frames.

var region = ee.Geometry.Polygon(

[[[-86.9, 11.2],

[-81.5, 11.2],

[-86.8, 7.6],

[-81.5, 7.6]]],

null, false

);

col = col.map(function(img) {

var doy = ee.Date(img.get('system:time\_start')).getRelative('day', 'year');

return img.set('doy', doy);

});

var distinctDOY = col.filterDate('2021-1-01', '2021-12-31');

// Define a filter that identifies which images from the complete collection

// match the DOY from the distinct DOY collection.

var filter = ee.Filter.equals({leftField: 'doy', rightField: 'doy'});

// Define a join.

var join = ee.Join.saveAll('doy\_matches');

// Apply the join and convert the resulting FeatureCollection to an

// ImageCollection.

var joinCol = ee.ImageCollection(join.apply(distinctDOY, col, filter));

// Apply median reduction among matching DOY collections.

var comp = joinCol.map(function(img) {

var doyCol = ee.ImageCollection.fromImages(

img.get('doy\_matches')

);

return doyCol.reduce(ee.Reducer.median());

});

var multicomp = comp.toBands()

print(comp, "comp");

// try to export

Export.image.toDrive({

image: multicomp,

description: 'compositeMODIS2021',

region: mask,

'maxPixels':1e13

});

// Define RGB visualization parameters.

var visParams = {

min: 0.0,

max: 9000.0,

palette: [

'FFFFFF', 'CE7E45', 'DF923D', 'F1B555', 'FCD163', '99B718', '74A901',

'66A000', '529400', '3E8601', '207401', '056201', '004C00', '023B01',

'012E01', '011D01', '011301'

],

};

// Create RGB visualization images for use as animation frames.

var rgbVis = comp.map(function(img) {

return img.visualize(visParams).clip(mask);

});

// Define GIF visualization parameters.

var gifParams = {

'region': region,

'dimensions': 600,

'crs': 'EPSG:3857',

'framesPerSecond': 10

};

// Print the GIF URL to the console.

print(rgbVis.getVideoThumbURL(gifParams));

// Render the GIF animation in the console.

print(ui.Thumbnail(rgbVis, gifParams));