Publication: Doggart, N., Mugasha, W.A., Mpiri, A., Morgan-Brown, T., Sallu, S.M., Spracklen, D.V., 2023. Agricultural fallows are the main driver of natural forest regeneration in Tanzania. Environmental Research Letters.

Script author: Theron Morgan-Brown

Earth Engine script for review of 1987 – 2021 land cover data

This Google Earth Engine Script is used to view the random sample points using the least cloudy images / data from Landsat 5, 7 and 8, Sentinel 2 and PALSAR 1 and 2. Images are loaded with the oldest images at the top. The images were reviewed visually and the Remote Sensing Sample Points (RSSPs) classified according to the classes described in Supplementary File 1.

// Load sample sites

var table = ee.FeatureCollection('users/S2\_Random\_Sample\_Points\_4000\_plot\_boundaries'); // sample plot boundary table

//var table3 = ee.FeatureCollection('users/S2\_Random\_Sample\_Points\_4000'); // enter sample point table

// Choose the range of months to view Satellite data for

var startMonth = 1;

var endMonth = 6;

// Generate image year lists for each satellite

// and reverse them so that maps will load in reverse

// chronological order

var LT5years = ee.List.sequence(1987, 2012); //Landsat 5

var LT5years = LT5years.reverse();

var LE7years = ee.List.sequence(2000, 2015); //Landsat 7

var LE7years = LE7years.reverse();

var LC8years = ee.List.sequence(2013, 2016); //Landsat 8

var LC8years = LC8years.reverse();

var S2years = ee.List.sequence(2016, 2021); //Sentinel 2

var S2years = S2years.reverse();

var PAL1years = ee.List.sequence(2007, 2010); //PALSAR 1 and 2

var PAL1years = PAL1years.reverse();

var PAL2years = ee.List.sequence(2015, 2018); //PALSAR 1 and 2

var PAL2years = PAL2years.reverse();

var PAL2years2 = ee.List.sequence(2019, 2020); //PALSAR 1 and 2

var PAL2years2 = PAL2years2.reverse();

// Function to generate a list of least cloudy Sentinel-2 tiles

// for the selected years and selected sample point

var funS2list = function(year) {

year = year;

var S = ee.ImageCollection("COPERNICUS/S2")

.filter(ee.Filter.calendarRange(year,year,'year'))

.filter(ee.Filter.calendarRange(startMonth,endMonth,'month'))

.sort('CLOUDY\_PIXEL\_PERCENTAGE', true)

.filterBounds(Map.getCenter());

var mylist= S.toList(3);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of second least cloudy Sentinel-2 tiles

// for the selected years and selected sample point

var funS2list2 = function(year) {

year = year;

var S = ee.ImageCollection("COPERNICUS/S2")

.filter(ee.Filter.calendarRange(year,year,'year'))

.filter(ee.Filter.calendarRange(startMonth,endMonth,'month'))

.sort('CLOUDY\_PIXEL\_PERCENTAGE', true)

.filterBounds(Map.getCenter());

var mylist= S.toList(3);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(1)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of least cloudy Landsat 5 scenes

// for the selected years and selected sample point

var funLT5list = function(year) {

year = year;

var S = ee.ImageCollection("LANDSAT/LT05/C01/T1\_SR")

.filter(ee.Filter.calendarRange(year,year,'year'))

.filter(ee.Filter.calendarRange(startMonth,endMonth,'month'))

.sort('CLOUD\_COVER', true)

.filterBounds(Map.getCenter());

var mylist= S.toList(3);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of least cloudy Landsat 7 scenes

// for the selected years and selected sample point

var funLE7list = function(year) {

year = year;

var S = ee.ImageCollection("LANDSAT/LE07/C01/T1\_SR")

.filter(ee.Filter.calendarRange(year,year,'year'))

.filter(ee.Filter.calendarRange(startMonth,endMonth,'month'))

.sort('CLOUD\_COVER', true)

.filterBounds(Map.getCenter());

var mylist= S.toList(3);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of least cloudy Landsat 8 scenes

// for the selected years and selected sample point

var funLC8list = function(year) {

year = year;

var S = ee.ImageCollection("LANDSAT/LC08/C01/T1\_SR")

.filter(ee.Filter.calendarRange(year,year,'year'))

.filter(ee.Filter.calendarRange(startMonth,endMonth,'month'))

.sort('CLOUD\_COVER', true)

.filterBounds(Map.getCenter());

var mylist= S.toList(3);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of PALSAR 1 scenes

// for the selected years and selected sample point

var funPAL1list = function(year) {

year = year;

var S = ee.ImageCollection('JAXA/ALOS/PALSAR/YEARLY/SAR')

.filter(ee.Filter.calendarRange(year,year,'year'))

.filterBounds(Map.getCenter());

var mylist= S.toList(1);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to generate a list of PALSAR 2 scenes

// from selected years at selected sample point

var funPAL2list = function(year) {

year = year;

var S = ee.ImageCollection('JAXA/ALOS/PALSAR/YEARLY/SAR')

.filter(ee.Filter.calendarRange(year,year,'year'))

.filterBounds(Map.getCenter());

var mylist= S.toList(1);

var s = ee.Algorithms.If(ee.Number(S.size()).gte(1),

ee.Image(mylist.get(0)), //change to 1 or 2 for 2nd or 3rd least cloudy

ee.Image(0));

return s;

};

// Function to add selected Landsat 5 images to map

var addL5Image = function(feature, displacement) {

var image = ee.Image(feature.id);

var sd = image.displace(displacement);

var layer = ui.Map.Layer(sd,{

bands: ['B3', 'B2', 'B1'],

min: [146, 167, 113],

max: [1519, 1387, 1101],

},'updating...',0);

var layerName = ee.String('L5 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected Landsat 7 images to map

var addLE7Image = function(feature, displacement) {

var image = ee.Image(feature.id);

var sd = image.displace(displacement);

var layer = ui.Map.Layer(sd,{

bands: ['B3', 'B2', 'B1'],

min: [146, 167, 113],

max: [1519, 1387, 1101],

},'updating...',0);

var layerName = ee.String('L7 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected Landsat 8 images to map

var addLC8Image = function(feature, displacement) {

var image = ee.Image(feature.id);

var sd = image.displace(displacement);

var layer = ui.Map.Layer(sd,{

bands: ['B4', 'B3', 'B2'],

min: [146, 167, 113],

max: [1519, 1387, 1101],

},'updating...',0);

var layerName = ee.String('L8 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected Sentinel 2 images to map

var addS2Image = function(feature) {

var image = ee.Image(feature.id);

var layer = ui.Map.Layer(image,{

bands: ['B4', 'B3', 'B2'],

min: [357, 567, 755],

max: [1561, 1365, 1326],

},'updating...',0);

var layerName = ee.String('S2 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected PALSAR 1 images to map

var addPAL1Image = function(feature) {

var image = ee.Image(feature.id);

var sarHV = image.select('HV');

var sarHH = image.select('HH');

var sarHHoHV = sarHH.divide(sarHV);

var s = ee.Image.cat([sarHH, sarHV, sarHHoHV]);

var disConst = ee.Image([60,-30]); // adjust x and y displacement values here in meters

var sd = s.displace(disConst);

var layer = ui.Map.Layer(sd,{

bands: ['HH', 'HV', 'HH\_1'],

min: [2443, 1113, 1.55],

max: [7952, 4493, 3.04],

gamma: [1, 1, 1]

},'updating...',0);

var layerName = ee.String('P1 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected PALSAR 2 images to map

var addPAL2Image = function(feature) {

var image = ee.Image(feature.id);

var sarHV = image.select('HV');

var sarHH = image.select('HH');

var sarHHoHV = sarHH.divide(sarHV);

var s = ee.Image.cat([sarHH, sarHV, sarHHoHV]);

var disConst = ee.Image([60,-30]); // adjust x and y displacement values here in meters

var sd = s.displace(disConst);

var layer = ui.Map.Layer(sd,{

bands: ['HH', 'HV', 'HH\_1'],

min: [2443, 1113, 1.55],

max: [7952, 4493, 3.04],

gamma: [1, 1, 1]

},'updating...',0);

var layerName = ee.String('P2 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add selected PALSAR 2 images to map

var addPAL2Image2 = function(feature) {

var image = ee.Image(feature.id);

var sarHV = image.select('HV');

var sarHH = image.select('HH');

var sarHHoHV = sarHH.divide(sarHV);

var s = ee.Image.cat([sarHH, sarHV, sarHHoHV]);

//var disConst = ee.Image([60,-30]); // adjust x and y displacement values here in meters

//var sd = s.displace(disConst);

var layer = ui.Map.Layer(s,{

bands: ['HH', 'HV', 'HH\_1'],

min: [2443, 1113, 1.55],

max: [7952, 4493, 3.04],

gamma: [1, 1, 1]

},'updating...',0);

var layerName = ee.String('P2 ').cat(image.date().format('YYYY-MM-dd'));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Function to add sample plot boundary to map

var addPlotGrid = function(text) {

//var row = ee.Number.parse(text);

var geom = table.filter(ee.Filter.eq('name', text));

var image = ee.Image();

var pixelOut = image.paint(geom,1,1);

var layer = ui.Map.Layer(pixelOut,{'palette': '98ff00'},'updating...');

var layerName = ee.String('Sample #').cat(ee.String(text));

Map.layers().add(layer);

// evaluate layerName asynchroneously (server -> client) and update in the map layer

layerName.evaluate(function(layerName) {

layer.setName(layerName);

});

};

// Generate a text box for the user to enter the sample point to review

var label = ui.Label('Enter point to review in box below and press enter');

print(label);

var textbox = ui.Textbox({

placeholder: 'Enter point to review here...',

onChange: function(text) {

var tablef = table.filter(ee.Filter.eq('name', text));

Map.setOptions('SATELLITE');

Map.clear();

Map.centerObject(tablef, 16); // Centers map on sample point with specified zoom level

//Generate Sentinel-2 Collection for user selected point

var S2list = ee.ImageCollection(S2years.map(funS2list));

var S2\_2016 = S2list.sort("system:time\_start", true).first();

//Generate Sentinel-2 Collection for user selected point

var S2list2 = ee.ImageCollection(S2years.map(funS2list2));

// Combine Sentinel-2 lists and sort by date

S2list = S2list.merge(S2list2);

S2list = S2list.sort("system:time\_start", false).getInfo();

//Generate Landsat 8 Collection for user selected point

var LC8list = ee.ImageCollection(LC8years.map(funLC8list));

var LC8\_2016 = LC8list.first();

LC8list = LC8list.getInfo();

var point = Map.getCenter();

var square = point.buffer(5000).bounds();

// Use bicubic resampling during registration.

var LC8\_2016r = LC8\_2016.resample('bicubic');

var S2\_2016r = S2\_2016.resample('bicubic');

// Choose to register using only the 'R' band.

var LC8\_2016r\_b6 = LC8\_2016r.select('B4').clip(square);

var S2\_2016r\_b6 = S2\_2016r.select('B4').clip(square);

// Determine the displacement by matching only the 'R' bands.

var displacement = LC8\_2016r\_b6.displacement({

referenceImage: S2\_2016r\_b6,

maxOffset: 50.0,

//patchWidth: 100.0

});

// Compute image offset and direction.

//var offset = displacement.select('dx').hypot(displacement.select('dy'));

//var angle = displacement.select('dx').atan2(displacement.select('dy'));

// Display offset distance and angle.

//Map.addLayer(offset, {min:0, max: 20}, 'offset');

//Map.addLayer(angle, {min: -Math.PI, max: Math.PI}, 'angle');

//Generate Landsat 7 Collection for user selected point

//var LE7list = ee.ImageCollection(LE7years.map(funLE7list)).getInfo();

// Generate List of Landsat 7 images for user selected point

var LE7list = LE7years.map(funLE7list);

// Generate Landsat 7 Collection and remove missing years

var LE7listf = ee.ImageCollection(LE7list)

.map(function(image) {

return image.set('count', image.bandNames().length());

})

.filter(ee.Filter.eq('count', 11)).getInfo();

// Generate List of Landsat 5 images for user selected point

var LT5list = LT5years.map(funLT5list);

// Generate Landsat 5 Collection and remove missing years

var LT5listf = ee.ImageCollection(LT5list)

.map(function(image) {

return image.set('count', image.bandNames().length());

})

.filter(ee.Filter.eq('count', 11)).getInfo();

//Generate PALSAR Collection for user selected point

var PAL1list = ee.ImageCollection(PAL1years.map(funPAL1list)).getInfo();

var PAL2list = ee.ImageCollection(PAL2years.map(funPAL2list)).getInfo();

var PAL2list2 = ee.ImageCollection(PAL2years2.map(funPAL2list)).getInfo();

S2list.features.map(addS2Image); // load Sentinel-2 image lists

addPlotGrid(text); // load plot boundary

PAL2list2.features.map(addPAL2Image2); // load PALSAR 2 2019-2020 image lists

PAL2list.features.map(addPAL2Image); // load PALSAR 2 2015-2018 image lists

LC8list.features.map(function(x) { return addLC8Image(x, displacement); }

);

//LC8list.features.map(addLC8Image); // load Landsat 8 image lists

PAL1list.features.map(addPAL1Image); // load PALSAR 1 image lists

LE7listf.features.map(function(x) { return addLE7Image(x, displacement); }

); // load Landsat 5 image lists

LT5listf.features.map(function(x) { return addL5Image(x, displacement); }

); // load Landsat 5 image lists

addPlotGrid(text); // load plot boundary

}

});

print(textbox);