Geospatial Software Design

An attempt to predict land cover change using Google Earth Engine

Andry Rajaoberison, MEM'19

In short here is my project:

project/final

```
Imports (3 entries)
 var belo: Polygon, 4 vertices
 var training: Table users/rajaoberison/training_belo
 var map_center: Point (44.44, -19.85)

// Attempted Land Cover Change Prediction by Andry Rajaoberison
// NB: This code is to provide insights on how to do prediction analysis or randow-walk
// using Google Earth Engine. No accuracy assessment were conducted and the training
// for classification were based on obsevation of high resolution Google Earth Imagery.


// SCALE OF THE STUDY
var scale = 30; // meters



/** DEFINING FUNCTIONS **/
/*** OTSU FUNCTION ***/
// https://medium.com/google-earth/otsus-method-for-image-segmentation-f5c48f405e
var otsu = function(histogram) {⟺};


/*** RANDOM FOREST CLASSIFIER GIVEN TRAINING REGIONS ***/
var RFclassifier = function(image, training0, training1, trainingbands, scale){⟺};


/*** LANDSAT 5 IMAGE CLASSIFIER ***/
var l5classifier = function(year, aoi, training_region, scale){⟺};


/*** TRANSITION MATRIX CALCULATOR ***/
var transition_matrix = function(before_image, current_image, year, aoi, scale){⟺};


/*** RANDOM WALK FUNCTION ***/
// This requires a transition matrix which is calculated above.
// For each of the pixels, the current state is given by the rows of the average matrix
// Then, the next state of the land cover is given by the result of product of
// current state * average transition matrix (within the timeframe)
// As the current state is a 1D array (vector), the product will occur for each column
// of the average matrix, whcih means, we have to get it's transposed version
// Here's the function for all of that
var random_walk = function(current_cover, bandNameOfClasses, average_matrix){⟺};



/** MAIN CODE **/
/*** LAND COVER CLASSIFICATION GIVEN THE REGION OF STUDY ***/⟺


/*** COMPUTING TRANSITION MATRIX ***/⟺


/*** RANDOM WALKING ***/
// First we need the average of the transition matrices
/**** AVERAGE TRANSITION MATRIX ****/⟺

/**** SIMULATION FROM 2008 to 2010 ****/⟺



/** PRESENTATION **/⟺
```

What it’s trying to achieve is random walk simulation within land cover types, using average transition matrix from multi-year land cover change. The goal is to predict the next state of the land cover based on the current state.

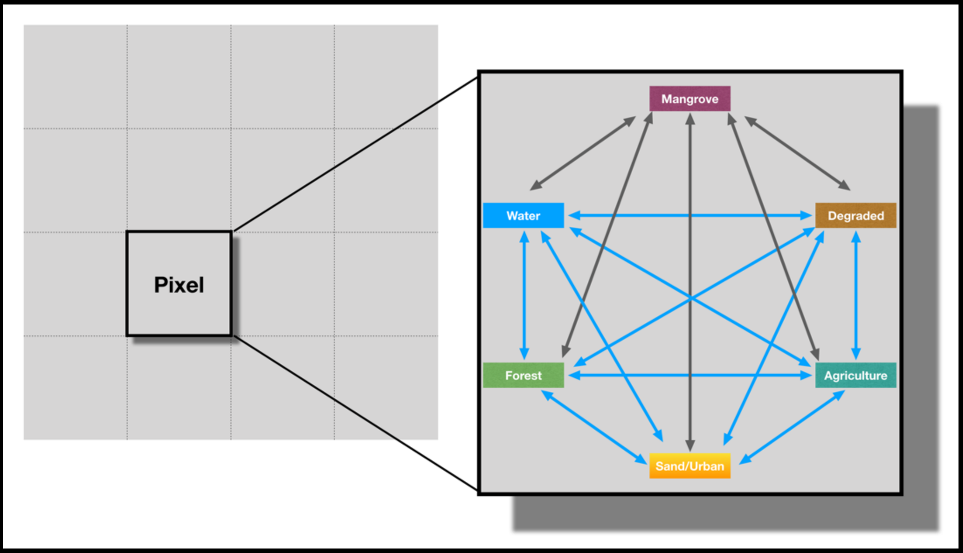
The script works in terms of changing pixel/land cover values based on the transition matrix. However, (crucial) spatial information related to mangrove change is not yet included.

Follow this link for more information and animation:
https://github.com/rajaoberison/LandcoverPrediction

Introduction

In this project, I'm trying to predict landcover change using simple random walk within landcover pixels. The platform used was Google Earth Engine, and one of the main challenges was to incorporate pixel location information in the script. (And I'm still working on that part actually). This script example is specifically designed for mangrove cover change.

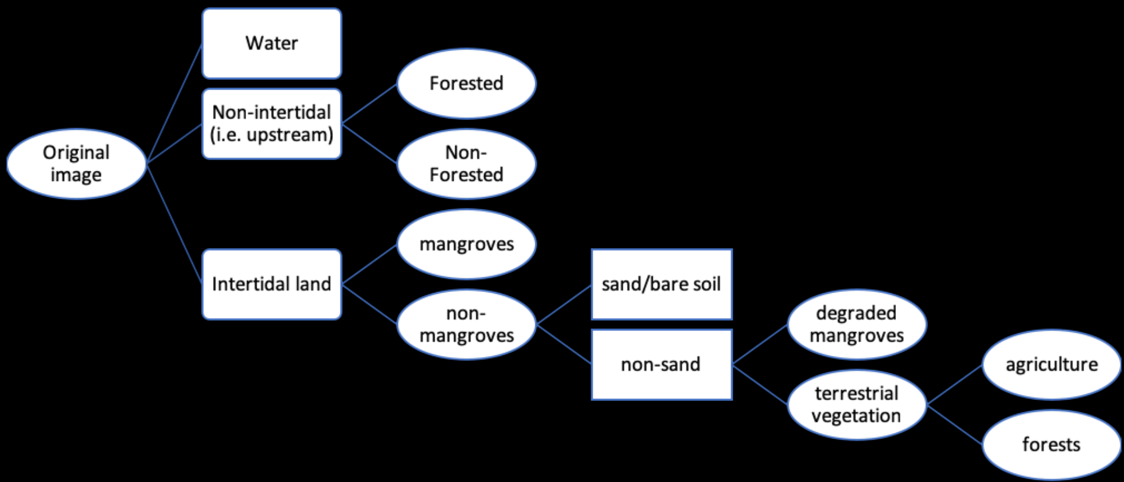
Mangroves are trees and shrubs that inhabit the interface between land and sea of the tropics and subtropics. Their natural distribution is limited, globally, by temperature (20°C winter isotherm of seawater), and, regionally and locally, by rainfall, tidal inundation, and freshwater inflow bringing nutrients and silt (Kathiresan and Bingham, 2001; Alongi and Brinkman, 2011). Additionally, mangroves are abundant in zones of small topographical gradients, well-drained soils, and large tidal amplitudes; but they do poorly in stagnant water (Gopal and Krishnamurthy, 1993; Van Loon et al., 2016).



Based on the assumption that only mangroves can tolerate intertidal areas, my method will assume that there are 6 types of landcover that each pixel can convert into, namely: mangroves, degraded mangroves, terrestrial forest, farming, sand / bare soil / urban, and water. Each state will convert to another state based on the probability transition matrix that will be calculated based on landcover classification, frequency of storm, upstream deforestation rate, proximity to human population and restoration project. (NB: In the example simulation, I did not include water yet but just the 5 “land” covers.)

Land Cover Classification

Getting accurate landcover class for the study area is crucial for this analysis, so I developed a code for landcover classification, which uses Landsat 5, elevation subset, Otsu segmentation, and random forest to produce binary class at each step.



This land cover classification allowed me to produce a transition matrix of with probabilities of the conversion of each pixel from one state to another. This information is not enough however, for the prediction analysis, because factors such as storm frequency, anthropogenic pressures, and upstream forest cover are not yet taken into account. I will try to calculate this probability using Bayesian inference.

Example Simulation

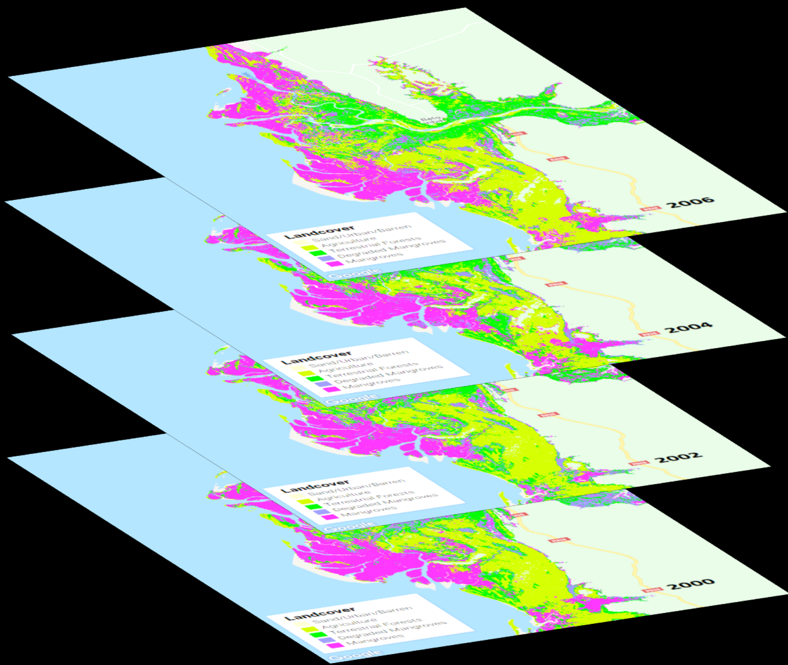
But first let's simulate a simple random walk using the classes and the transition matrix obtained from the classification. By choosing a study region in Belo-sur-Tsiribihina, Madagascar, and a timeframe of 2000 to 2010 (with a two-year intervall), I obtained the following outputs. For this example, water was not yet included but just the land covers.

In short, the script looks like this:

```
// Attempted Land Cover Change Prediction by Andry Rajaoberison
// NB: This code is to provide insights on how to do prediction analysis or random-walk
// using Google Earth Engine. No accuracy assessment was conducted and the training
// for classification were based on observation of high-resolution Google Earth Imagery.


// SCALE OF THE STUDY
var scale = 30; // meters

/** DEFINING FUNCTIONS **/
/*** OTSU FUNCTION ***/
// https://medium.com/google-earth/otsus-method-for-image-segmentation-f5c48f405e
var otsu = function(histogram) {<==>};
```

```
/*** RANDOM FOREST CLASSIFIER GIVEN TRAINING REGIONS ***/
var RFclassifier = function(image, training0, training1, trainingbands, scale){<==>};


/*** LANDSAT 5 IMAGE CLASSIFIER ***/
var l5classifier = function(year, aoi, training_region, scale){<==>};


/*** TRANSITION MATRIX CALCULATOR ***/
var transition_matrix = function(before_image, current_image, year, aoi, scale){<==>};


/*** RANDOM WALK FUNCTION ***/
// This requires a transition matrix which is calculated above.
// For each of the pixels, the current state is given by the rows of the average matrix
// Then, the next state of the land cover is given by the result of product of
// current state * average transition matrix (within the timeframe)
// As the current state is a 1D array (vector), the product will occur for each column
// of the average matrix, whcih means, we have to get it's transposed version
// Here's the function for all of that
var random_walk = function(current_cover, bandNameOfClasses, average_matrix){<==>};


/** MAIN CODE **/
/*** LAND COVER CLASSIFICATION GIVEN THE REGION OF STUDY ***/<==>

/*** COMPUTING TRANSITION MATRIX ***/<==>


/*** RANDOM WALKING ***/ <==>
// First we need the average of the transition matrices
/**** AVERAGE TRANSITION MATRIX ****/<==>

/**** SIMULATION FROM 2008 to 2010 ****/<==>



/** PRESENTATION **/<==>
```

Here are some typical process and outputs:




```
List (5 elements)
  0: List (5 elements)
    0: 0.2419990247581154
    1: 0.6358339264988899
    2: 0.0891839349642396
    3: 0.00624382309615612
    4: 0.026739284687209874
  1: List (5 elements)
    0: 0.0041831182898022234
    1: 0.20265485160052776
    2: 0.6472322568297386
    3: 0.10657323710620403
    4: 0.039356546476483345
  2: List (5 elements)
    0: 0.00027762347599491477
    1: 0.022220771992579103
    2: 0.2718224301934242
    3: 0.4094647541642189
    4: 0.29621441662311554
  3: List (5 elements)
    0: 0
    1: 0.07641559187322855
    2: 0.09800009615719318
    3: 0.21377254277467728
    4: 0.6118117719888687
  4: List (5 elements)
    0: 0
    1: 0.15008085640147328
    2: 0.02753212465904653
    3: 0.2191981626674533
    4: 0.6031888499855995
```


Probability transition matrix

0.24	0.63	0.09	0.01	0.03
0.01	0.20	0.65	0.11	0.04
0.00	0.02	0.27	0.41	0.29
0	0.08	0.10	0.21	0.61
0	0.15	0.03	0.22	0.60

0.24	0.63	0.09	0.01	0.03

X

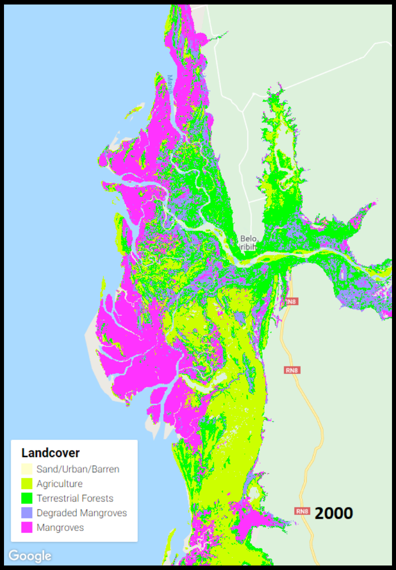
Current state: ex: For state 0 (i.e. sand)

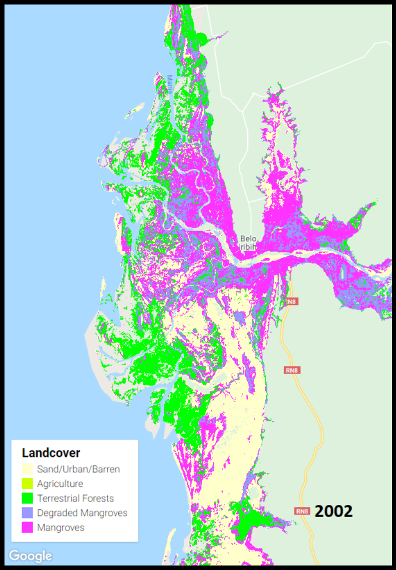
The next state will be 2 (i.e. Forest)

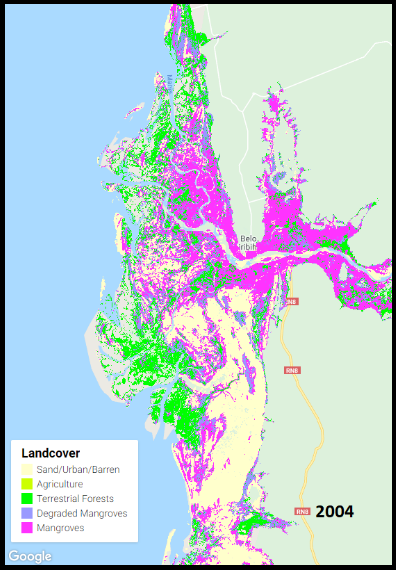
0.06	0.28	$\boxed{0.45}$	0.11	0.08

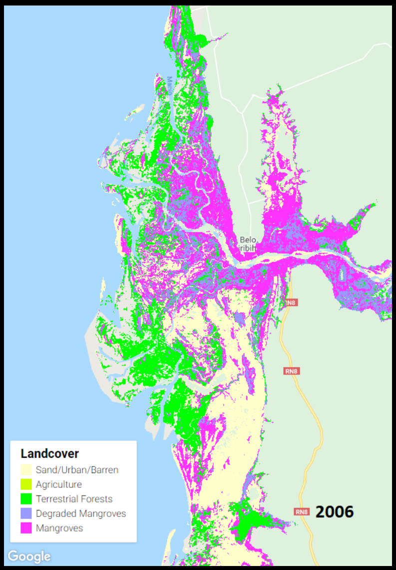
For each pixel, their current state is their row on the transition matrix. So, we multiply that with the transition matrix and get the state with the highest probability to be the next state.

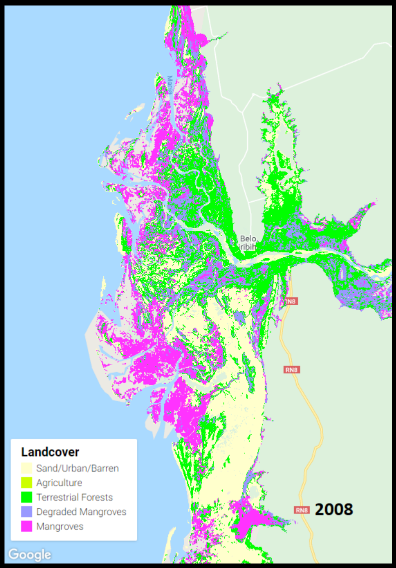
Simulation from 2000 to 2012 (based on year 2000 as the know state):

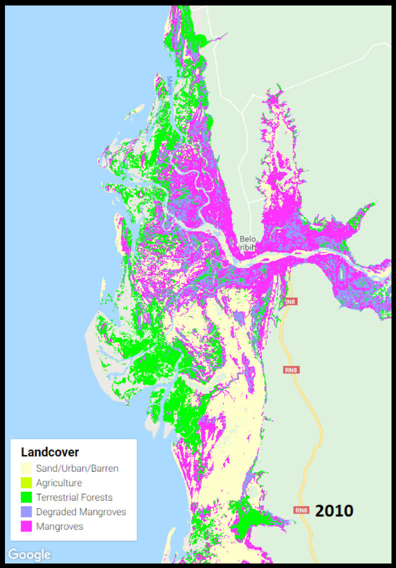


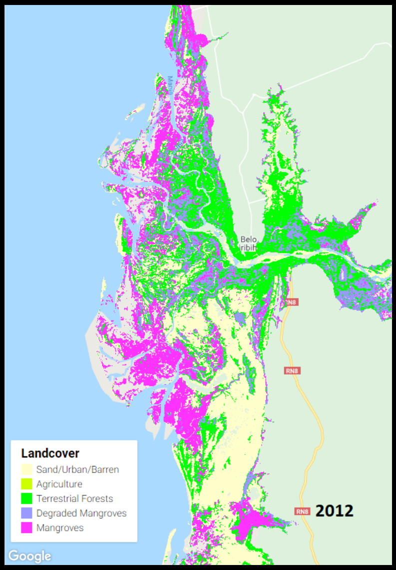




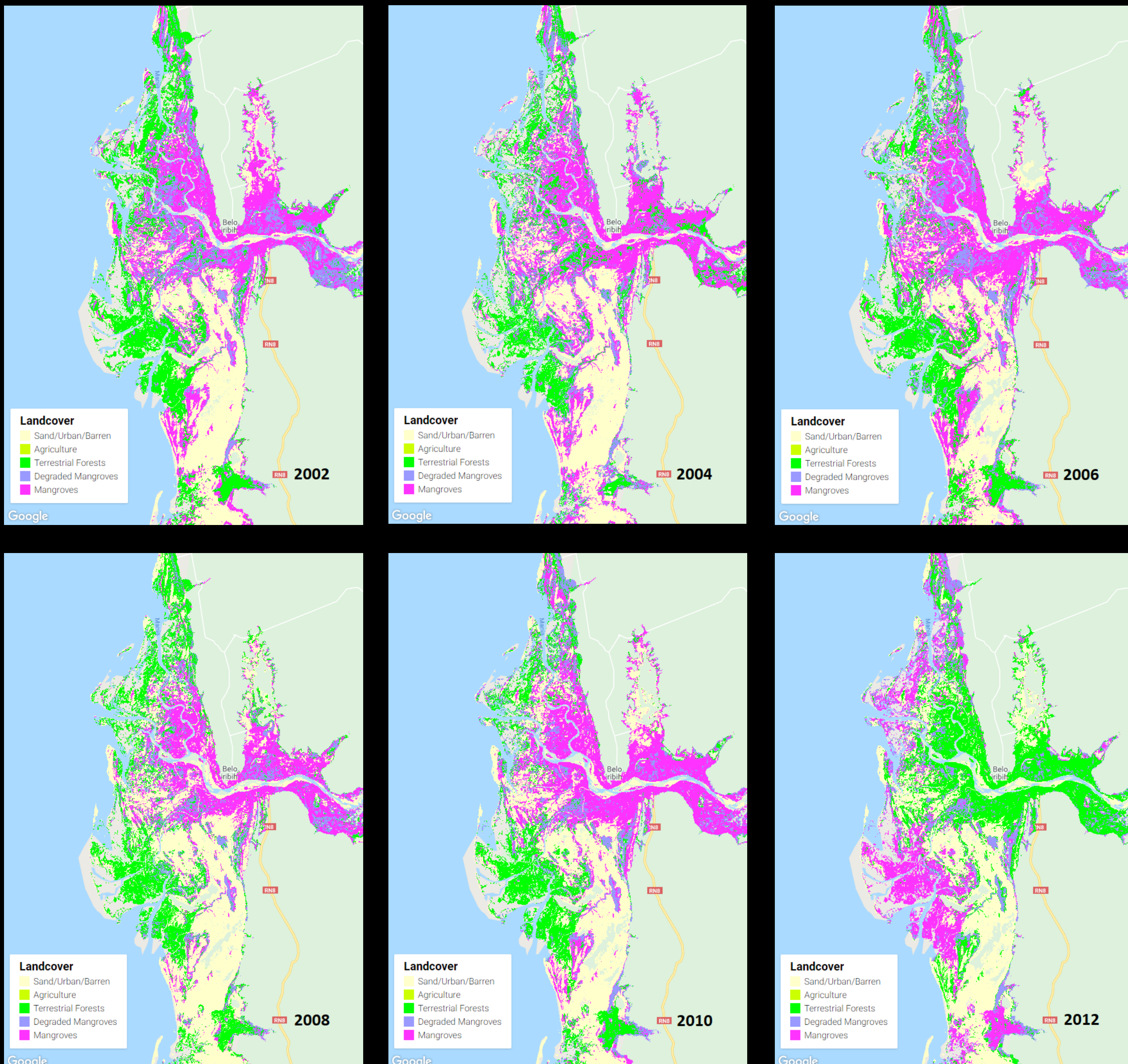








Each land covers predicted from their previous years:



As we can see, what the script does is: it will assign for all landcovers of type "a" to some new land cover of type "b". So it will convert everything, all mangroves to some land cover, all terrestrial forests to some land cover, and so on. While visually, it produces results a little off from the actual land cover, the scripts still provide insights into when where the mangroves vulnerable within the timeframe of study.

If you look closely at the far-right simulation, mangroves are completely lost at the early 2000 but then come back around 2010, I think this is because of the rate of mangrove loss higher in the early 2000 and slower around 2010. Obviously, a way to correct this script is to incorporate some spatial information in the calculation of the probabilities, such as proximity of the land cover to population centers, proximity to coastline, frequency of storms, and upstream land cover (all of which may affect mangrove change).
The next step of this script will try to incorporate this information.

Next Steps

For the next steps, the updating of the landcover class (the random walk) will go by pixels and not by land cover type. This is something, I still find challenging to implement on Earth Engine as I haven't mastered some of its capabilities yet.

The full script of where I am right now can be found below or at:

https://code.earthengine.google.com/f17bb611b56c7332a8f9b3f4ad6efef6

```
// Attempted Land Cover Change Prediction by Andry Rajaoberison
// NB: This code is to provide insights on how to do prediction analysis or randow-walk
// using Google Earth Engine. No accuracy assessment were conducted and the training
// for classification were based on obsevation of high resolution Google Earth Imagery.


// SCALE OF THE STUDY
var scale = 30; // meters



/** DEFINING FUNCTIONS **/
/*** OTSU FUNCTION ***/
// https://medium.com/google-earth/otsus-method-for-image-segmentation-f5c48f405e
var otsu = function(histogram) {
  var counts = ee.Array(ee.Dictionary(histogram).get('histogram'));
  var means = ee.Array(ee.Dictionary(histogram).get('bucketMeans'));
  var size = means.length().get([0]);
  var total = counts.reduce(ee.Reducer.sum(), [0]).get([0]);
  var sum = means.multiply(counts).reduce(ee.Reducer.sum(), [0]).get([0]);
  var mean = sum.divide(total);

  var indices = ee.List.sequence(1, size);

  // Compute between sum of squares, where each mean partitions the data.
  var bss = indices.map(function(i) {
    var aCounts = counts.slice(0, 0, i);
    var aCount = aCounts.reduce(ee.Reducer.sum(), [0]).get([0]);
    var aMeans = means.slice(0, 0, i);
    var aMean = aMeans.multiply(aCounts)
        .reduce(ee.Reducer.sum(), [0]).get([0])
        .divide(aCount);
    var bCount = total.subtract(aCount);
    var bMean = sum.subtract(aCount.multiply(aMean)).divide(bCount);
    return aCount.multiply(aMean.subtract(mean).pow(2)).add(
           bCount.multiply(bMean.subtract(mean).pow(2)));
  });

  //print(ui.Chart.array.values(ee.Array(bss), 0, means));

  // Return the mean value corresponding to the maximum BSS.
  return means.sort(bss).get([-1]);
};


/*** RANDOM FOREST CLASSIFIER GIVEN TRAINING REGIONS ***/
var RFclassifier = function(image, training0, training1, trainingbands, scale){
  //*** IMAGE CLASSIFICATION FUNCTION ***/
  // CLASSIFICATION
  //Create random points inside polygons
  //Take a random sample inside the polygons for training
```

```
  var mang_tpts0 = ee.FeatureCollection.randomPoints(training0, 2000, 0);
  var notmang_tpts0 = ee.FeatureCollection.randomPoints(training1, 2000, 0);

  //Take a random sample inside the polygons for validation
  var mang_vpts = ee.FeatureCollection.randomPoints(training0, 600, 1);
  var notmang_vpts = ee.FeatureCollection.randomPoints(training1, 600, 1);

  //ADD CLASS FIELD
  //add class field for mangrove points
  var addField = function(training0) {
    //var addclass = ee.Number(mangroves.get('landcover'));
    return training0.set({'landcover': 1});
  };

  var mang_tpts = mang_tpts0.map(addField);
  //var mang_vpts = mang_vpts.map(addField);

  //add class field for notmangrove points
  var addField2 = function(training1) {
    //var addclass = ee.Number(notmangroves.get('landcover'));
    return training1.set({'landcover': 0});
  };
  var notmang_tpts = notmang_tpts0.map(addField2);
  //var notmang_vpts = notmang_vpts.map(addField2);

  //Merging random points
  var trainingpts = mang_tpts.merge(notmang_tpts);
  //var validpts = mang_vpts.merge(notmang_vpts);

  // Train the classifier
  // Sample the input imagery to get a FeatureCollection of training data.
  var training = image.sampleRegions({
  collection: trainingpts,
  properties: ['landcover'],
  scale: scale
  });

  // Make a random forest classifier and train it.
  var classifier = ee.Classifier.randomForest(10)
      .train(training, 'landcover', trainingbands);

  // Classify the input imagery.
  var classified = image.select(trainingbands).classify(classifier).rename('class');

return classified;
};


/*** LANDSAT 5 IMAGE CLASSIFIER ***/
var l5classifier = function(year, aoi, training_region, scale){
```

```

/**** IMAGE PREPARATION ****/
// Get Landsat Images
// Year is defined as the Tropical cyclone season
// https://en.wikipedia.org/wiki/2018%E2%80%9319_South-West_Indian_Ocean_cyclone_season
var year_0 = year - 1;
var raw = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')
    .filterDate(year_0+'-05-01', year+'-10-31').filterBounds(aoi)
    .filter(ee.Filter.lte('CLOUD_COVER_LAND', 10));

Map.centerObject(map_center, 11);
var visImage = {bands: ['B4', 'B5', 'B1'], min: 140, max: 4300};
//Map.addLayer(raw, visImage , 'raw '+year, false);

/***** USE THE CLOUD REMOVAL SCRIPT FROM GEE EXAMPLES *****/
// This example demonstrates the use of the Landsat 4, 5 or 7
// surface reflectance QA band to mask clouds.

var cloudMaskL457 = function(image) {
  var qa = image.select('pixel_qa');
  // If the cloud bit (5) is set and the cloud confidence (7) is high
  // or the cloud shadow bit is set (3), then it's a bad pixel.
  var cloud = qa.bitwiseAnd(1 << 5)
          .and(qa.bitwiseAnd(1 << 7))
          .or(qa.bitwiseAnd(1 << 3));
  // Remove edge pixels that don't occur in all bands
  var mask2 = image.mask().reduce(ee.Reducer.min());
  return image.updateMask(cloud.not()).updateMask(mask2);
};

// Map the function over the collection, take the median, and clip.

var cloudRemoved = raw
    .map(cloudMaskL457)
    .median();

//Map.addLayer(cloudRemoved, visImage, 'cloud removed '+ year, false);

/***** REMOVING WATER *****/
// OTSU THRESHOLDING TECHNIQUE
// Also water classification: water vs. non-water

// Compute the histogram of the NIR band.  The mean and variance are only FYI.
var histogram = cloudRemoved.select('B4').reduceRegion({
  reducer: ee.Reducer.histogram(255, 2)
      .combine('mean', null, true)
      .combine('variance', null, true),
  geometry: aoi,
  scale: scale,
  bestEffort: false
});
```

```
  //print(histogram);

  // Chart the histogram
  //print(Chart.image.histogram(cloudRemoved.select('B4'), aoi, 30));

  var threshold = otsu(histogram.get('B4_histogram'));
  //print('threshold '+year+': '+ threshold.getInfo());

  var waterMask = cloudRemoved.select('B4').gt(threshold);

  var waterMasked = cloudRemoved.mask(waterMask);
  //Map.addLayer(waterMasked, visImage, 'water masked '+year, false);

  /***** SUBSETTING BY INTERTIDAL / MANGROVE AREAS *****/
  // IMPORT GIRI (2011) AS REFERENCE
  var giri = ee.ImageCollection('LANDSAT/MANGROVE_FORESTS').filterBounds(aoi);
  //Map.addLayer(giri, {color:'grey'}, 'Giri 2000', false);

  // BASED ON DEM VARIANCE
  var dem = ee.Image('JAXA/ALOS/AW3D30_V1_1').clip(aoi).select('AVE');
  var demPalette = ['blue', 'lightBlue', 'darkGreen', 'brown', 'white'];

  // Not suitable for mangroves, if elevation below 30m
  //var below30m = dem.lte(30);

  //Map.addLayer(dem, {min:0, max:30, palette: demPalette}, 'JAXA_DEM', false);

  // Actual Interdidal zones using tide data
  // https://www.tideschart.com/Madagascar/Diana/Nosy-Be/
  var intertidal =
cloudRemoved.updateMask(ee.ImageCollection([giri.mosaic().focal_mode(10).toInt(),
                                       dem.mask(dem.lte(10)).rename('1').toInt()]).mosaic()
                                       .updateMask(waterMask)).clip(aoi);

  //Map.addLayer(intertidal, visImage, 'inter '+year, false);

  /**** TRAINING DATA ****/
  // Training polygons
  var sand = training_region.filter(ee.Filter.eq('landcover', '0'));
  var mangroves = training_region.filter(ee.Filter.eq('landcover', '1'));
  var deg_mangroves = training_region.filter(ee.Filter.eq('landcover', '2'));
  var forest = training_region.filter(ee.Filter.eq('landcover', '3'));
  var agri = training_region.filter(ee.Filter.eq('landcover', '4'));

  var notmangroves = sand.merge(deg_mangroves).merge(forest).merge(agri);
  var notsand = deg_mangroves.merge(forest).merge(agri);
  var terrestrial_veg = forest.merge(agri);

  /**** IMAGE ANALYSIS ****/
  /***** MAPPING MANGROVES *****/
  var final = intertidal;
```

```
// WATER AND VEGETATION INDEXES
// NDVI
var ndvi = final.normalizedDifference(['B4', 'B3']).rename('ndvi');
var vegPalette = ['blue', 'white', 'darkgreen'];
//Map.addLayer(ndvi, {min:-0.1, max:0.5, palette: vegPalette}, 'ndvi '+year, false);

// EVI
var evi0 = final.expression
  ('2.5 * ((NIR - RED) / (NIR + 6 * RED - 7.5 * BLUE + 1))',
    {
      'NIR': final.select('B4'),
      'RED': final.select('B3'),
      'BLUE': final.select('B1')
    }
  );
var evi = evi0.select('constant').rename('evi');

// Ref: Bo-cai Gao, 1996, NDWI-A normalized difference water index for remote sensing of
vegetation
// liquid water from space,Remote Sensing of Environment, Volume 58, Issue 3, Pages 257-266,
//https://doi.org/10.1016/S0034-4257(96)00067-3.
// http://www.sciencedirect.com/science/article/pii/S0034425796000673)
// NDWI
var ndwi = final.normalizedDifference(['B4', 'B5']).rename('ndwi');
//Map.addLayer(ndwi, {min:-0.3, max:0.6, palette: waterPalette}, 'ndwi '+year, false);

// Ref: Hanqiu Xu (2006) Modification of normalised difference water index (NDWI)
// to enhance open water features in remotely sensed imagery, International Journal of Remote
// Sensing, 27:14, 3025-3033, DOI: 10.1080/01431160600589179
// MNDWI
var mndwi = final.normalizedDifference(['B2', 'B5']).rename('mndwi');
//Map.addLayer(mndwi, {min:-0.3, max:0.6, palette: waterPalette}, 'ndwi '+year, false);

// Band ratios: reference Green, E.P.; Clark, C.D.; Mumby, P.J.; Edwards, A.J.;
// Ellis, A.C. Remote sensing techniques for mangrove mapping. Int. J. Remote
// Sens. 1998, 19, 935-956.

// Band swir/nir ratio (band 5:4 for Landsat5, 6:5 for Landsat 8)
var ratio54 = final.select('B5').divide(final.select('B4')).rename('ratio54');
//Map.addLayer(mndwi, {min:-1, max:1, palette: waterPalette}, 'ndwi '+year, false);

// Band Red:SWIR ratio (band 3:5 for landsat 5, 4:6 for landsat 8)
var ratio35 = final.select('B3').divide(final.select('B5')).rename('ratio35');
//Map.addLayer(mndwi, {min:-1, max:1, palette: waterPalette}, 'ndwi '+year, false);


// Prep for classification: stack all bands, indicies, ratios
var final_stack = final
  .addBands(ndvi)
  .addBands(ndwi)
  .addBands(mndwi)
```

```
  .addBands(evi)
  .addBands(ratio54)
  .addBands(ratio35);

//var bands = final_stack.bandNames();
//print('Band names: ', bands);

var trainingbands = ee.List(['B1','B2','B3','B4','B5','B7','ndvi', 'ndwi',
                              'mndwi','evi','ratio54','ratio35']);

var classified = RFclassifier(final_stack, mangroves, notmangroves, trainingbands, scale);

// Extract Mangroves
// Create a binary mask from classification
var mangrove_mask = classified.select('class').eq(1);
var classified_mangrove = classified.updateMask(mangrove_mask);
//Map.addLayer(classified_mangrove, {palette: 'purple'}, 'mangroves ' + year, false);

/***** MAPPING TERRESTRIAL LAND *****/
var notmangrove_mask = classified.select('class').eq(0);
var notmangrove_zones = intertidal.updateMask(notmangrove_mask);

//Map.addLayer(notmangrove_zones, visImage, 'non mangroves '+year, false);

// TASSELED CAP TRANSFORMATION
// Define an Array of Tasseled Cap coefficients.
var coefficients = ee.Array([
  [0.3037, 0.2793, 0.4743, 0.5585, 0.5082, 0.1863],
  [-0.2848, -0.2435, -0.5436, 0.7243, 0.0840, -0.1800],
  [0.1509, 0.1973, 0.3279, 0.3406, -0.7112, -0.4572],
  [-0.8242, 0.0849, 0.4392, -0.0580, 0.2012, -0.2768],
  [-0.3280, 0.0549, 0.1075, 0.1855, -0.4357, 0.8085],
  [0.1084, -0.9022, 0.4120, 0.0573, -0.0251, 0.0238]
]);

// Make an Array Image, with a 1-D Array per pixel.
var arrayImage1D = notmangrove_zones.select(['B1', 'B2', 'B3', 'B4', 'B5', 'B7']).toArray();

// Make an Array Image with a 2-D Array per pixel, 6x1.
var arrayImage2D = arrayImage1D.toArray(1);

// Do a matrix multiplication: 6x6 times 6x1.
var tasseled = ee.Image(coefficients)
  .matrixMultiply(arrayImage2D)
  // Get rid of the extra dimensions.
  .arrayProject([0])
  .arrayFlatten(
    [['brightness', 'greenness', 'wetness', 'fourth', 'fifth', 'sixth']]);

// Display the first three bands of the result and the input imagery.
var vizParams = {
```

```
    bands: ['brightness', 'greenness', 'wetness'],
    min: -0.1, max: [0.5, 0.1, 0.1]
  };

  //Map.addLayer(tasseled, vizParams, 'components');

  var terr = notmangrove_zones.addBands(tasseled.select('brightness'))
    .addBands(tasseled.select('greenness'))
    .addBands(tasseled.select('wetness'));

  var trainingbands_2 = ee.List(['B1','B2','B3','B4','B5','B7','brightness',
                                 'greenness','wetness']);

  var classified_2 = RFclassifier(terr, sand, notsand, trainingbands_2, scale);

  var sand_mask = classified_2.select('class').eq(1);
  var classified_sand = classified_2.updateMask(sand_mask);

  //Map.addLayer(classified_sand, {palette: 'orange'}, 'sand ' + year, false);

  /***** MAPPING TERRESTRIAL VEGETATION *****/
  // We don't have multiseries options so we'll use indexes
  // https://medium.com/regen-network/remote-sensing-indices-389153e3d947
  var green_mask = classified_2.select('class').eq(0);
  var green_zones = intertidal.updateMask(green_mask);

  var avi = green_zones.expression
    ('cbrt((B4 + 1) * (256 - B3) * (B4 - B3))',
      {
        'B4': green_zones.select('B4'),
        'B3': green_zones.select('B3')
      }
    );
  avi = avi.rename('avi');

  var bi = green_zones.expression
    ('((B4 + B2) - B3)/((B4 + B2) + B3)',
      {
        'B4': green_zones.select('B4'),
        'B3': green_zones.select('B3'),
        'B2': green_zones.select('B2')
      }
    );
  bi = bi.rename('bi');

  var si = green_zones.expression
    ('sqrt((256 - B2) * (256 - B3))',
      {
        'B2': green_zones.select('B2'),
        'B3': green_zones.select('B3')
      }
```

```
    );
  si = si.rename('si');

  var terr_forest = green_zones.addBands(avi)
    .addBands(bi).addBands(si);

  var trainingbands_3 = ee.List(['avi', 'bi', 'si']);

  var classified_3 = RFclassifier(terr_forest, deg_mangroves, terrestrial_veg, trainingbands_3,
scale);

  var deg_mask = classified_3.select('class').eq(1);
  var classified_deg = classified_3.updateMask(deg_mask);

  //Map.addLayer(classified_deg, {palette: 'grey'}, 'degmang ' + year, false);

  /***** MAPPING AGRI vs. FOREST *****/
  var green2_mask = classified_3.select('class').eq(0);
  var green2_zones = terr_forest.updateMask(green2_mask);

  var classified_4 = RFclassifier(green2_zones, forest, agri, trainingbands_3, scale);

  var forest_mask = classified_4.select('class').eq(1);
  var ag_mask = classified_4.select('class').eq(0);

  var classified_forest = classified_4.updateMask(forest_mask);
  var classified_agri = classified_4.updateMask(ag_mask);

  //Map.addLayer(classified_forest, {palette: 'green'}, 'forest ' + year, false);
  //Map.addLayer(classified_agri, {palette: 'D6E744'}, 'agri ' + year, false);

  var all = ee.ImageCollection.fromImages([
    classified_mangrove.select('class').rename(year.toString()).multiply(5).toInt(),
    classified_sand.select('class').rename(year.toString()).multiply(1).toInt(),
    classified_deg.select('class').rename(year.toString()).multiply(4).toInt(),
    classified_forest.select('class').rename(year.toString()).multiply(3).toInt(),
    classified_agri.select('class').rename(year.toString()).add(2).toInt()
    ]);

  //Map.addLayer(all, {min:0, max:4, palette: ['white', 'red']}, 'all ' + year, true);

return all.mosaic();

};


/*** TRANSITION MATRIX CALCULATOR ***/
var transition_matrix = function(before_image, current_image, year, aoi, scale){

  // Let's remap the pixels into transition states
  var remap = before_image.remap([1,2,3,4,5], [10,20,30,40,50], null, year);
```

```
  var before_to_current = remap.add(current_image).toUint8();
  // These transition states are:
ee.List([11,12,13,14,15,21,22,23,24,25,31,32,33,34,35,41,42,43,44,45,51,52,53,54,55])

  // Now for the actual transition
  var transition_histogram = before_to_current.select('remapped').reduceRegion({reducer:
ee.Reducer.histogram(), geometry: aoi, scale: scale});

  var transition_probabilities = function(histogram){
    var counts =
ee.Array(ee.Dictionary(ee.Dictionary(transition_histogram).get('remapped')).get('histogram'));
    var from_class_0 = counts.slice(0, 0, 5).toList();
    var sum_from_class_0 = from_class_0.reduce(ee.Reducer.sum());
    var from_class_1 = counts.slice(0, 10, 15).toList();
    var sum_from_class_1 = from_class_1.reduce(ee.Reducer.sum());
    var from_class_2 = counts.slice(0, 20, 25).toList();
    var sum_from_class_2 = from_class_2.reduce(ee.Reducer.sum());
    var from_class_3 = counts.slice(0, 30, 35).toList();
    var sum_from_class_3 = from_class_3.reduce(ee.Reducer.sum());
    var from_class_4 = counts.slice(0, 40, 45).toList();
    var sum_from_class_4 = from_class_4.reduce(ee.Reducer.sum());
    return ee.Array([
      from_class_0.map(function(i){
      return ee.Number(i).divide(sum_from_class_0)}),
      from_class_1.map(function(i){
      return ee.Number(i).divide(sum_from_class_1)}),
      from_class_2.map(function(i){
      return ee.Number(i).divide(sum_from_class_2)}),
      from_class_3.map(function(i){
      return ee.Number(i).divide(sum_from_class_3)}),
      from_class_4.map(function(i){
      return ee.Number(i).divide(sum_from_class_4)})]);
  };

  return transition_probabilities(transition_histogram);

};


/*** RANDOM WALK FUNCTION ***/
// This requires a transition matrix which is calculated above.
// For each of the pixels, the current state is given by the rows of the average matrix
// Then, the next state of the land cover is given by the result of product of
// current state * average transition matrix (within the timeframe)
// As the current state is a 1D array (vector), the product will occur for each column
// of the average matrix, whcih means, we have to get it's transposed version
// Here's the function for all of that
var random_walk = function(current_cover, bandNameOfClasses, average_matrix){

  // Define the classes. Here we have 5 classes.
  var class_list = ee.List.sequence(0,4);
```

```
  var average_matrix_flatten = ee.List(average_matrix.toList().flatten());

  // Function for new class identification
  var new_image_class = class_list.map(function(i){

    // Current states
    var current_state = ee.Array(average_matrix_flatten.slice(ee.Number(0).add(i),
ee.Number(5).add(i)));
    // Transposed transition matrix
    var average_matrix_bycolumns = average_matrix.matrixTranspose().toList().flatten();
    // Getting the corresponding arrays for multiplication
    var trans0 = ee.Array(average_matrix_bycolumns.slice(0,5));
    var trans1 = ee.Array(average_matrix_bycolumns.slice(5,10));
    var trans2 = ee.Array(average_matrix_bycolumns.slice(10,15));
    var trans3 = ee.Array(average_matrix_bycolumns.slice(15,20));
    var trans4 = ee.Array(average_matrix_bycolumns.slice(20,25));
    // Array multiplication and summing
    var new0 = current_state.multiply(trans0).reduce(ee.Reducer.sum(), [0]);
    var new1 = current_state.multiply(trans1).reduce(ee.Reducer.sum(), [0]);
    var new2 = current_state.multiply(trans2).reduce(ee.Reducer.sum(), [0]);
    var new3 = current_state.multiply(trans3).reduce(ee.Reducer.sum(), [0]);
    var new4 = current_state.multiply(trans4).reduce(ee.Reducer.sum(), [0]);

    // Get the probabilities of the new states
    var new_state = ee.Array.cat([new0, new1, new2, new3, new4]);
    // Get the maximum probability
    var max_for_new_state = new_state.reduce(ee.Reducer.max(), [0]).get([0]).format('%.5f');

    // The value is in long float values so let's convert to string for better indexation
    var new_state_string = ee.List([new0.get([0]).format('%.5f'), new1.get([0]).format('%.5f'),
          new2.get([0]).format('%.5f'), new3.get([0]).format('%.5f'),
new4.get([0]).format('%.5f')]);

    // Get the new class
    var new_class = new_state_string.indexOf(max_for_new_state);

    // And remap the image
    return ee.Image(current_cover.remap([i], [new_class], null,
bandNameOfClasses).rename("new_class")).toUint8();
  });

  // Return the mosaic of all classes
  return ee.ImageCollection.fromImages(new_image_class).mosaic();

};



/** MAIN CODE **/
/*** LAND COVER CLASSIFICATION GIVEN THE REGION OF STUDY ***/
// From 2000 to 2008 with a two-year intervall
```

```
var cover_2000 = l5classifier(2000, belo, training, scale);
var cover_2002 = l5classifier(2002, belo, training, scale);
var cover_2004 = l5classifier(2004, belo, training, scale);
var cover_2006 = l5classifier(2006, belo, training, scale);
var cover_2008 = l5classifier(2008, belo, training, scale);

// For visualization
var classesViz = {min:0, max:4, palette: ['FFFFCC','CCFF00','00FF00','9999FF','FF33FF']};
Map.addLayer(cover_2000, classesViz, '2000', false);
Map.addLayer(cover_2002, classesViz, '2002', false);
Map.addLayer(cover_2004, classesViz, '2004', false);
Map.addLayer(cover_2006, classesViz, '2006', false);
Map.addLayer(cover_2008, classesViz, '2008', false);


/*** COMPUTING TRANSITION MATRIX ***/
var from00to02 = transition_matrix(cover_2000, cover_2002, "2000", belo, scale);
var from02to04 = transition_matrix(cover_2002, cover_2004, "2002", belo, scale);
var from04to06 = transition_matrix(cover_2004, cover_2006, "2004", belo, scale);
var from06to08 = transition_matrix(cover_2006, cover_2008, "2006", belo, scale);


/*** RANDOM WALKING ***/
// First we need the average of the transition matrices
/**** AVERAGE TRANSITION MATRIX ****/
// Flattening to easily get the average with reducer
var all_matrix = ee.Array([
  from00to02.toList().flatten(), from02to04.toList().flatten(),
  from04to06.toList().flatten(), from06to08.toList().flatten()]);

var average_matrix = all_matrix.reduce(ee.Reducer.mean(), [0]);

// Now, unflatten them
average_matrix = ee.List(average_matrix.toList().get(0));

average_matrix = ee.Array([
  average_matrix.slice(0,5), average_matrix.slice(5,10),
  average_matrix.slice(10,15), average_matrix.slice(15,20),
  average_matrix.slice(20,25)]);

print(average_matrix);

/**** SIMULATION FROM 2008 to 2010 ****/
var walk_to_2010 = random_walk(cover_2008, "2008", average_matrix);
Map.addLayer(walk_to_2010, classesViz, '2010_walk', true);
var sim = random_walk(cover_2000, "2000", average_matrix);

// FOR THE ACTUAL 2010 cover
var cover_2010 = l5classifier(2010, belo, training, scale);
Map.addLayer(cover_2010, classesViz, '2010_actual', true);
```

```
/** PRESENTATION **/
// https://mygeoblog.com/2016/12/09/add-a-legend-to-to-your-gee-map/
// set position of panel
var legend = ui.Panel({
  style: {
    position: 'bottom-left',
    padding: '8px 15px'
  }
});

// Create legend title
var legendTitle = ui.Label({
  value: 'Landcover',
  style: {
    fontWeight: 'bold',
    fontSize: '18px',
    margin: '0 0 4px 0',
    padding: '0'
    }
});

// Add the title to the panel
legend.add(legendTitle);

// Creates and styles 1 row of the legend.
var makeRow = function(color, name) {

      // Create the label that is actually the colored box.
      var colorBox = ui.Label({
        style: {
          backgroundColor: '#' + color,
          // Use padding to give the box height and width.
          padding: '8px',
          margin: '0 0 4px 0'
        }
      });

      // Create the label filled with the description text.
      var description = ui.Label({
        value: name,
        style: {margin: '0 0 4px 6px'}
      });

      // return the panel
      return ui.Panel({
        widgets: [colorBox, description],
        layout: ui.Panel.Layout.Flow('horizontal')
      });
```

```
};

//  Palette with the colors
var palette =['FFFFCC','CCFF00','00FF00','9999FF','FF33FF'];

// name of the legend
var names = ['Sand/Urban/Barren','Agriculture','Terrestrial Forests', 'Degraded Mangroves',
'Mangroves'];

// Add color and and names
for (var i = 0; i < 5; i++) {
  legend.add(makeRow(palette[i], names[i]));
  }

// add legend to map (alternatively you can also print the legend to the console)
Map.add(legend);
```