Appendix A. Supplementary Material

Combined Landsat and L-band SAR data for land cover classification and change detection in dynamic tropical landscapes

Jose Don T. De Alban *, Grant M. Connette, Patrick Oswald, Edward L. Webb *

* Corresponding authors: dondealban@gmail.com (J. D. De Alban); ted.webb@nus.edu.sg (E. L. Webb)

Contents

1. Scripts for Landsat image compositing

Links to the original Google Earth Engine image compositing scripts

a. Modifed script by M. Gonzales Roglich:

https://code.earthengine.google.com/c506e197ad85f01d42b22abba9b70ae0

b. Original codes by C. Stam & I. Housman:

https://ee-api.appspot.com/a432bf20510f37ce13e847e4394aee77

Modified script for 1995 Landsat image composite

```
/*
 * This script executes cloud and shadow masking in Landsat Top-of-Atmosphere images and exports the resulting
 * composite image for subsequent use in image classification.
 *
 * The script is based on the modifications by Mariano Gonzalez Roglich (Conservation International) on the Fine Forest
 * Monitoring Workbench Prototype 0.0.1 by Juan Doblas (Instituto Socioambiental), and mainly the  RSAC Temporal Dark
 * Outlier Mask (TDOM) Compositing Script originally developed by Carson Stam and Ian Housman (Red Castle Resources,
 * Inc.).
 *
 * Links to scripts:
 * Modifed script by M. Gonzales Roglich: https://code.earthengine.google.com/c506e197ad85f01d42b22abba9b70ae0
 * Original codes by C. Stam & I. Housman: https://ee-api.appspot.com/a432bf20510f37ce13e847e4394aee77
 *
 */

/*******************************
  USER DEFINITIONS
*******************************/

// Define area of interest
var region_name = "Tanintharyi";
var fc = ee.Geometry.Rectangle(97.0,16.0, 100.0,9.0);

Map.addLayer(fc, {color: '000000'}, 'region', false);
   var saName = region_name;
var crs = 'EPSG:4326'; // CRS = WGS84 (or use 'EPSG:32647' for WGS84/UTM Zone 47N)

//Compositing Parameters
var years = [1994];//Specify years
var compositingPeriod = 2;//Number of years to include in the composite
var startJulian = 0;//Start julian date- supports wrapping
var endJulian = 365;//End julian date

var shadowLookBack = 0;//Number of years to look back for cloud shadow masking
var shadowLookForward = 0;//Number of years to look forward for cloud shadow masking

//More user inputs
var cloudThresh = 10;//Specify the cloud threshold- lower number = more clouds are excluded
var cloudExpandIterations = 0;

var possibleSensors = ee.List(['L5']); //Specify which sensors to pull from- supports L5,L7,L8
```

```
var reduceOrTakeNewestPixel = 1;   // 1 - Will compute median (or other reducer) of the masked collection 0 - Will take newest pixel (still experimental)
var reducer = ee.Reducer.percentile([50]);//Reducer for compositing

var shadowSumBands = ee.List(['swir1','swir2']);//Bands for shadow masking. Removed NIR to avoid commision on deforested areas

var zShadowThresh = -1; //Z score in which shadows are expected to be below
var shadowExpandIterations = 0;

//Names of collections to look in //Add _L1T for L1T imagery //TOA is computed on both the L1G or L1T
var collection_dict = {L5: 'LT5_L1T', L7: 'LE7_L1T', L8: 'LC8_L1T'};

//Band combinations for each sensor corresponding to final selected corresponding bands
var sensor_band_dict = ee.Dictionary({L5 : ee.List([0,1,2,3,4,5,6]),
                                      L7 : ee.List([0,1,2,3,4,5,7]),
                                      L8 : ee.List([1,2,3,4,5,9,6])});
var spacecraft_dict = {'Landsat5': 'L5', 'Landsat7': 'L7', 'Landsat8': 'L8'};

//End user inputs

////////////////////////////////////////////////////////////////////
//Finds the minimum bounding rectangle from the study area
var region = fc.bounds().getInfo().coordinates[0];

var allImages = ee.ImageCollection([ee.Image(1).set('system:time_start',new Date('1/1/1940'))]);
////////////////////////////////////////////////////////////////////
//Choose which visualization parameter to use
var vizParams = {'min': 0.05,'max': [0.3,0.4,0.4],   'bands':'swir1,nir,red', 'gamma':1.6};

////////////////////////////////////////////////////////////////////
var bandNames = ee.List(['blue','green','red','nir','swir1','temp','swir2']);

////////////////////////////////////////////////////////////////////
var bandNumbers = [0,1,2,3,4,5,6];

////////////////////////////////////////////////////////////////////
//Function to mask clouds, ensure data exists in every band
function maskCloudsAndSuch(img){
  //Bust clouds
  var cs = ee.Algorithms.Landsat.simpleCloudScore(img).select(['cloud']).gt(cloudThresh);
  if (cloudExpandIterations>0) {cs = cs.focal_max(1,'circle','pixels',cloudExpandIterations)}
  //Make sure all or no bands have data
  var numberBandsHaveData = img.mask().reduce(ee.Reducer.sum());
  var allOrNoBandsHaveData = numberBandsHaveData.eq(0).or(numberBandsHaveData.gte(7));
  var allBandsHaveData = allOrNoBandsHaveData;
  //Make sure no band is just under zero
  var allBandsGT = img.reduce(ee.Reducer.min()).gt(-0.001);
  return img.mask(img.mask().and(cs.not()).and(allBandsHaveData).and(allBandsGT));
}
////////////////////////////////////////////////////////////////////
function addShadowSum(img){
    return img.addBands(img.select(shadowSumBands).reduce(ee.Reducer.sum()).select([0],['shadowSum']));
}

////////////////////////////////////////////////////////////////////
function zShadowMask(img,meanShadowDark,stdShadowDark){
  var imgDark = img.select(['shadowSum']);
  var shadowZ = imgDark.subtract(meanShadowDark).divide(stdShadowDark);
  var shadows = shadowZ.lt(zShadowThresh);
  if (shadowExpandIterations>0) {shadows = shadows.focal_max(1,'circle','pixels',shadowExpandIterations)}
  return img.mask(img.mask().and(shadows.not()));
```

```
}
////////////////////////////////////////////////////////////////////
//Function to handle empty collections that will cause subsequent processes to fail
//If the collection is empty, will fill it with an empty image
function fillEmptyCollections(inCollection,dummyImage){
  var dummyCollection = ee.ImageCollection([dummyImage.mask(ee.Image(0))]);
  var imageCount =  inCollection.toList(1).length();
  return ee.ImageCollection(ee.Algorithms.If(imageCount.gt(0),inCollection,dummyCollection));
}
////////////////////////////////////////////////////////////////////
//Wrapper function to get composite
var metadataFC = '';
function getImage(year,compositingPeriod, startJulian,endJulian,shadowLookBack,shadowLookForward){
  //Define dates
  var y1Image = year;
  var y2Image = year + compositingPeriod-1;

  var startDate = ee.Date.fromYMD(ee.Number(year),1,1).advance(startJulian,'day');
  var endDate =
ee.Date.fromYMD(ee.Number(year).add(ee.Number(compositingPeriod)).subtract(ee.Number(1)),1,1).advance(endJulian,'day');
  print(startDate,endDate);
  var shadowStartDate = startDate.advance(ee.Number(-1).multiply(ee.Number(shadowLookBack)),'year');
  var shadowEndDate = endDate.advance(ee.Number(shadowLookForward),'year');

  //Helper function to get images from a specified sensor
  function getCollection(sensor,startDate,endDate,startJulian,endJulian){
  var collectionName = collection_dict[sensor];

  //Start with an un-date-confined collection of images
  var WOD = ee.ImageCollection(collectionName)
        .filterBounds(fc)
        .map(ee.Algorithms.Landsat.TOA);

  //Pop off an image to serve as a template if there are no images in the date range
  var dummy = ee.Image(WOD.first());

  //Filter by the dates
  var ls = WOD
        .filterDate(startDate,endDate)
        .filter(ee.Filter.calendarRange(startJulian,endJulian));
  print(ls);

  //Fill the collection if it's empty
  ls = fillEmptyCollections(ls,dummy);

  //Clean the collection up- clouds, fringes....
  ls = ls.map(maskCloudsAndSuch)
        .select(sensor_band_dict.get(sensor),bandNames);
    return ls;
 }

  //Get the images for composite and shadow model
  var l5s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L5'),getCollection('L5',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L5',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));
  var l7s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L7'),getCollection('L7',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L7',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));
  var l8s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L8'),getCollection('L8',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L8',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));
```

```
  //Merge the collections
  var ls = ee.ImageCollection(l5s.merge(l7s).merge(l8s)).map(addShadowSum);
  print(ls);
  var shadowImageCount = ls.toList(100000,0).length();
  //Pop off an image to fill after date constriction for image range
  var dummy = ee.Image(ls.first());

  //Compute stats for dark pixels for shadow masking
  var meanShadowSum = ls.select(['shadowSum']).mean();
  var stdShadowSum = ls.select(['shadowSum']).reduce(ee.Reducer.stdDev());

  //Constrain collection to just the image date range
  var ls = ls.filterDate(startDate,endDate);
  var compositeImageCount = ls.toList(100000,0).length();
  //Fill it in case it's null
  ls = fillEmptyCollections(ls,dummy);

  //Apply z score shadow method
  ls = ls.map(function(img){return zShadowMask(img,meanShadowSum,stdShadowSum)});

  // Flattening of the collection is made by reducing it or by picking the last recorded (and non-masked) pixel
  if (reduceOrTakeNewestPixel){
    var composite = ls.reduce(reducer).select(bandNumbers,bandNames)}
  else {
    var ls_sorted = ls.sort('system:time_start', false);    //Order adquisition by date
    var composite= ls_sorted.mosaic();}

  var sDate = new Date(year,1,1);

  composite = composite.set({'system:time_start':sDate.valueOf()});
  composite = composite.addBands(ee.Image(year).select([0],['year']).float());

//  allImages = ee.ImageCollection(allImages.merge(ls))

  var fullName = saName+'_'+y1Image.toString()+'_' +y2Image.toString()+'_'+startJulian.toString()+'_'+endJulian.toString();
  var toExport = composite.select(['blue','green','red','nir','swir1','swir2']).multiply(10000).int16().clip(fc);

   //Set up metadata
   var f = ee.Feature(toExport.geometry())
       .set('bandNames',toExport.bandNames())
       .set('DateStart',startDate)
       .set('DateEnd',endDate)
       .set('JulianStart',startJulian)
       .set('JulianEnd',endJulian)
       .set('CloudThresh',cloudThresh)
       .set('system:index',fullName)
       .set('CompositingMethod','Percentile')
       .set('CompositingParameters', '50')
       .set('Sensors',possibleSensors)
       .set('bufferCloudShadow',true)
       .set('imageCountComposite',compositeImageCount)
       .set('imageCountShadow',shadowImageCount)
       .set('crs',crs)
       .set('CloudShadowStart',shadowStartDate)
       .set('CloudShadowEnd',shadowEndDate)
       .set('Z_ShadowThresh',zShadowThresh);

  //Append the metadata if it already exists
   f = ee.FeatureCollection([f]);
   if(metadataFC === ''){metadataFC = f}
    else{metadataFC = metadataFC.merge(f)};
```

```
  //Add to map and export composite
  Map.addLayer(composite,vizParams,year.toString() + '_composite', false);
  //Export.image(toExport,fullName,{'scale':30,'maxPixels':1e13,'crs':crs,'region': region});
  //Export.image.toAsset(toExport,fullName,{'scale':30,'maxPixels':1e13,'crs':crs,'region': fc});
  return composite;
}

//Compositing function call
var composites = ee.ImageCollection(years.map(function(yr){return
getImage(yr,compositingPeriod,startJulian,endJulian,shadowLookBack,shadowLookForward)}));
Export.table(metadataFC, saName+'_Composites_Metadata');

print('Old Composite:', composites);

Map.addLayer(ee.Image().paint(fc,1,1), {'palette': '00FFFF'},'Study Area', false);
//Map.centerObject(fc,8);

/*******************************
 DEFINE EXTENT AND VIEW
*******************************/

// Set center of map view
Map.setCenter(98.64212,12.40975, 7); // Zoom in and center display to Tanintharyi Region, Myanmar

// Define and display box extents covering Tanintharyi region
var box = fc; // Complete region

// Split box extents into regions for exporting
var boxTOP = ee.Geometry.Rectangle(97.0,16.0, 100.0,13.0); // Top region for export
var boxMID = ee.Geometry.Rectangle(97.0,13.0, 100.0,11.0); // Middle region for export
var boxLOW = ee.Geometry.Rectangle(97.0,11.0, 100.0, 9.0); // Lower region for export

/*******************************
 LOAD DATASETS
*******************************/

// Rename band filenames in metadata from the output composite
var composites = ee.ImageCollection(composites).select([0,1,2,3,4,5,6,7],['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'YR']);

// Mosaic the old composite (as image collection) into a new composite (as image)
var landsat1995 = composites.mosaic();
print('New Full Composite:', landsat1995);

/*******************************
 CLIP DATASETS
*******************************/

// Clip Landsat image composite within box extents
var clip1995 = landsat1995.clip(box);

// Split/clip image composite into regions for exporting
var clip1995top = landsat1995.clip(boxTOP); // Top region for export
var clip1995mid = landsat1995.clip(boxMID); // Middle region for export
var clip1995low = landsat1995.clip(boxLOW); // Lower region for export

/*******************************
 DISPLAY RGB COMPOSITES
*******************************/

// Display full image composite
Map.addLayer(clip1995, {bands: ['B5', 'B4', 'B3'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 1995 Landsat', false);
```

```
// Display split image composites
Map.addLayer(clip1995top, {bands: ['B5', 'B4', 'B3'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 1995 Top', true);
Map.addLayer(clip1995mid, {bands: ['B5', 'B4', 'B3'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 1995 Mid', true);
Map.addLayer(clip1995low, {bands: ['B5', 'B4', 'B3'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 1995 Low', true);

/*******************************
  PROCESS IMAGE ASSET
*******************************/

// Export full image composite to drive
// Export.image(clip1995, 'Tanintharyi_1995_Landsat', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': box}); // OK
// NOTE: image exports are split automatically due to cap on file size on export

// Export split image composite to drive
Export.image(clip1995top, 'TNI_1995_Landsat_TOP', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxTOP});
Export.image(clip1995mid, 'TNI_1995_Landsat_MID', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxMID});
Export.image(clip1995low, 'TNI_1995_Landsat_LOW', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxLOW});
```

Modified script for 2015 Landsat image composite

```
/*
 * This script executes cloud and shadow masking in Landsat Top-of-Atmosphere images and exports the resulting
 * composite image for subsequent use in image classification.
 *
 * The script is based on the modifications by Mariano Gonzalez Roglich (Conservation International) on the Fine Forest
 * Monitoring Workbench Prototype 0.0.1 by Juan Doblas (Instituto Socioambiental), and mainly the  RSAC Temporal Dark
 * Outlier Mask (TDOM) Compositing Script originally developed by Carson Stam and Ian Housman (Red Castle Resources,
 * Inc.).
 *
 * Links to scripts:
 * Modifed script by M. Gonzales Roglich: https://code.earthengine.google.com/c506e197ad85f01d42b22abba9b70ae0
 * Original codes by C. Stam & I. Housman: https://ee-api.appspot.com/a432bf20510f37ce13e847e4394aee77
 *
 */

/*******************************
  USER DEFINITIONS
*******************************/

// Define area of interest
var region_name = "Tanintharyi";
var fc = ee.Geometry.Rectangle(97.0,16.0, 100.0,9.0);

Map.addLayer(fc, {color: '000000'}, 'region', false);
   var saName = region_name;
var crs = 'EPSG:4326'; // CRS = WGS84 (or use 'EPSG:32647' for WGS84/UTM Zone 47N)

//Compositing Parameters
var years = [2015];//Specify years
var compositingPeriod = 2;//Number of years to include in the composite
var startJulian = 0;//Start julian date- supports wrapping
var endJulian = 365;//End julian date

var shadowLookBack = 0;//Number of years to look back for cloud shadow masking
var shadowLookForward = 0;//Number of years to look forward for cloud shadow masking

//More user inputs
var cloudThresh = 10;//Specify the cloud threshold- lower number = more clouds are excluded
var cloudExpandIterations = 0;

var possibleSensors = ee.List(['L8']); //Specify which sensors to pull from- supports L5,L7,L8
var reduceOrTakeNewestPixel = 1;   // 1 - Will compute median (or other reducer) of the masked collection 0 - Will take newest
pixel (still experimental)
var reducer = ee.Reducer.percentile([50]);//Reducer for compositing

var shadowSumBands = ee.List(['swir1','swir2']);//Bands for shadow masking. Removed NIR to avoid commision on deforested
areas

var zShadowThresh = -1; //Z score in which shadows are expected to be below
var shadowExpandIterations = 0;

//Names of collections to look in //Add _L1T for L1T imagery //TOA is computed on both the L1G or L1T
var collection_dict = {L5: 'LT5_L1T', L7: 'LE7_L1T', L8: 'LC8_L1T'};

//Band combinations for each sensor corresponding to final selected corresponding bands
var sensor_band_dict = ee.Dictionary({L5 : ee.List([0,1,2,3,4,5,6]),
                                      L7 : ee.List([0,1,2,3,4,5,7]),
                                      L8 : ee.List([1,2,3,4,5,9,6])});
var spacecraft_dict = {'Landsat5': 'L5', 'Landsat7': 'L7', 'Landsat8': 'L8'};
```

```
//End user inputs

////////////////////////////////////////////////////////////////////
//Finds the minimum bounding rectangle from the study area
var region = fc.bounds().getInfo().coordinates[0];

var allImages = ee.ImageCollection([ee.Image(1).set('system:time_start',new Date('1/1/1940'))]);
////////////////////////////////////////////////////////////////////
//Choose which visualization parameter to use
var vizParams = {'min': 0.05,'max': [0.3,0.4,0.4],   'bands':'swir1,nir,red', 'gamma':1.6};

////////////////////////////////////////////////////////////////////
var bandNames = ee.List(['blue','green','red','nir','swir1','temp','swir2']);

////////////////////////////////////////////////////////////////////
var bandNumbers = [0,1,2,3,4,5,6];

////////////////////////////////////////////////////////////////////
//Function to mask clouds, ensure data exists in every band
function maskCloudsAndSuch(img){
  //Bust clouds
  var cs = ee.Algorithms.Landsat.simpleCloudScore(img).select(['cloud']).gt(cloudThresh);
  if (cloudExpandIterations>0) {cs = cs.focal_max(1,'circle','pixels',cloudExpandIterations)}
  //Make sure all or no bands have data
  var numberBandsHaveData = img.mask().reduce(ee.Reducer.sum());
  var allOrNoBandsHaveData = numberBandsHaveData.eq(0).or(numberBandsHaveData.gte(7));
  var allBandsHaveData = allOrNoBandsHaveData;
  //Make sure no band is just under zero
  var allBandsGT = img.reduce(ee.Reducer.min()).gt(-0.001);
  return img.mask(img.mask().and(cs.not()).and(allBandsHaveData).and(allBandsGT));
}
////////////////////////////////////////////////////////////////////
function addShadowSum(img){
    return img.addBands(img.select(shadowSumBands).reduce(ee.Reducer.sum()).select([0],['shadowSum']));
}
////////////////////////////////////////////////////////////////////
function zShadowMask(img,meanShadowDark,stdShadowDark){
  var imgDark = img.select(['shadowSum']);
  var shadowZ = imgDark.subtract(meanShadowDark).divide(stdShadowDark);
  var shadows = shadowZ.lt(zShadowThresh);
  if (shadowExpandIterations>0) {shadows = shadows.focal_max(1,'circle','pixels',shadowExpandIterations)}
  return img.mask(img.mask().and(shadows.not()));
}
////////////////////////////////////////////////////////////////////
//Function to handle empty collections that will cause subsequent processes to fail
//If the collection is empty, will fill it with an empty image
function fillEmptyCollections(inCollection,dummyImage){
  var dummyCollection = ee.ImageCollection([dummyImage.mask(ee.Image(0))]);
  var imageCount =  inCollection.toList(1).length();
  return ee.ImageCollection(ee.Algorithms.If(imageCount.gt(0),inCollection,dummyCollection));
}
////////////////////////////////////////////////////////////////////
//Wrapper function to get composite
var metadataFC = '';
function getImage(year,compositingPeriod, startJulian,endJulian,shadowLookBack,shadowLookForward){
  //Define dates
  var y1Image = year;
  var y2Image = year + compositingPeriod-1;

  var startDate = ee.Date.fromYMD(ee.Number(year),1,1).advance(startJulian,'day');
```

```
  var endDate =
ee.Date.fromYMD(ee.Number(year).add(ee.Number(compositingPeriod)).subtract(ee.Number(1)),1,1).advance(endJulian,'day');
  print(startDate,endDate);
  var shadowStartDate = startDate.advance(ee.Number(-1).multiply(ee.Number(shadowLookBack)),'year');
  var shadowEndDate = endDate.advance(ee.Number(shadowLookForward),'year');

  //Helper function to get images from a specified sensor
  function getCollection(sensor,startDate,endDate,startJulian,endJulian){
  var collectionName = collection_dict[sensor];

  //Start with an un-date-confined collection of images
  var WOD = ee.ImageCollection(collectionName)
        .filterBounds(fc)
        .map(ee.Algorithms.Landsat.TOA);

  //Pop off an image to serve as a template if there are no images in the date range
  var dummy = ee.Image(WOD.first());

  //Filter by the dates
  var ls = WOD
        .filterDate(startDate,endDate)
        .filter(ee.Filter.calendarRange(startJulian,endJulian));
  print(ls);

  //Fill the collection if it's empty
  ls = fillEmptyCollections(ls,dummy);

  //Clean the collection up- clouds, fringes....
  ls = ls.map(maskCloudsAndSuch)
        .select(sensor_band_dict.get(sensor),bandNames);
    return ls;
  }

  //Get the images for composite and shadow model
  var l5s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L5'),getCollection('L5',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L5',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));
  var l7s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L7'),getCollection('L7',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L7',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));
  var l8s =
ee.ImageCollection(ee.Algorithms.If(possibleSensors.contains('L8'),getCollection('L8',shadowStartDate,shadowEndDate,startJuli
an,endJulian),getCollection('L8',ee.Date('1000-01-01'),ee.Date('1001-01-01'),0,365)));

  //Merge the collections
  var ls = ee.ImageCollection(l5s.merge(l7s).merge(l8s)).map(addShadowSum);
  print(ls);
  var shadowImageCount = ls.toList(100000,0).length();
  //Pop off an image to fill after date constriction for image range
  var dummy = ee.Image(ls.first());

  //Compute stats for dark pixels for shadow masking
  var meanShadowSum = ls.select(['shadowSum']).mean();
  var stdShadowSum = ls.select(['shadowSum']).reduce(ee.Reducer.stdDev());

  //Constrain collection to just the image date range
  var ls = ls.filterDate(startDate,endDate);
  var compositeImageCount = ls.toList(100000,0).length();
  //Fill it in case it's null
  ls = fillEmptyCollections(ls,dummy);

  //Apply z score shadow method
```

```
  ls = ls.map(function(img){return zShadowMask(img,meanShadowSum,stdShadowSum)});

  // Flattening of the collection is made by reducing it or by picking the last recorded (and non-masked) pixel
  if (reduceOrTakeNewestPixel){
    var composite = ls.reduce(reducer).select(bandNumbers,bandNames)}
  else {
    var ls_sorted = ls.sort('system:time_start', false);    //Order adquisition by date
    var composite= ls_sorted.mosaic();}

  var sDate = new Date(year,1,1);

  composite = composite.set({'system:time_start':sDate.valueOf()});
  composite = composite.addBands(ee.Image(year).select([0],['year']).float());

//  allImages = ee.ImageCollection(allImages.merge(ls))

  var fullName = saName+'_'+y1Image.toString()+'_' +y2Image.toString()+'_'+startJulian.toString()+'_'+endJulian.toString();
  var toExport = composite.select(['blue','green','red','nir','swir1','swir2']).multiply(10000).int16().clip(fc);

   //Set up metadata
   var f = ee.Feature(toExport.geometry())
       .set('bandNames',toExport.bandNames())
       .set('DateStart',startDate)
       .set('DateEnd',endDate)
       .set('JulianStart',startJulian)
       .set('JulianEnd',endJulian)
       .set('CloudThresh',cloudThresh)
       .set('system:index',fullName)
       .set('CompositingMethod','Percentile')
       .set('CompositingParameters', '50')
       .set('Sensors',possibleSensors)
       .set('bufferCloudShadow',true)
       .set('imageCountComposite',compositeImageCount)
       .set('imageCountShadow',shadowImageCount)
       .set('crs',crs)
       .set('CloudShadowStart',shadowStartDate)
       .set('CloudShadowEnd',shadowEndDate)
       .set('Z_ShadowThresh',zShadowThresh);

  //Append the metadata if it already exists
   f = ee.FeatureCollection([f]);
   if(metadataFC === ''){metadataFC = f}
    else{metadataFC = metadataFC.merge(f)};

   //Add to map and export composite
   Map.addLayer(composite,vizParams,year.toString() + '_composite', false);
   //Export.image(toExport,fullName,{'scale':30,'maxPixels':1e13,'crs':crs,'region': region});
   //Export.image.toAsset(toExport,fullName,{'scale':30,'maxPixels':1e13,'crs':crs,'region': fc});
   return composite;
}

//Compositing function call
var composites = ee.ImageCollection(years.map(function(yr){return
getImage(yr,compositingPeriod,startJulian,endJulian,shadowLookBack,shadowLookForward)}));
Export.table(metadataFC, saName+'_Composites_Metadata');

print('Old Composite:', composites);

Map.addLayer(ee.Image().paint(fc,1,1), {'palette': '00FFFF'},'Study Area', false);
//Map.centerObject(fc,8);
```

```
/*******************************
  DEFINE EXTENT AND VIEW
*******************************/

// Set center of map view
Map.setCenter(98.64212,12.40975, 7); // Zoom in and center display to Tanintharyi Region, Myanmar

// Define and display box extents covering Tanintharyi region
var box = fc; // Complete region

// Split box extents into regions for exporting
var boxTOP = ee.Geometry.Rectangle(97.0,16.0, 100.0,13.0); // Top region for export
var boxMID = ee.Geometry.Rectangle(97.0,13.0, 100.0,11.0); // Middle region for export
var boxLOW = ee.Geometry.Rectangle(97.0,11.0, 100.0, 9.0); // Lower region for export

/*******************************
  LOAD DATASETS
*******************************/

// Rename band filenames in metadata from the output composite
var composites = ee.ImageCollection(composites).select([0,1,2,3,4,5,6,7],['B2', 'B3', 'B4', 'B5', 'B6', 'B10', 'B7', 'YR']);

// Mosaic the old composite (as image collection) into a new composite (as image)
var landsat2015 = composites.mosaic();
print('New Full Composite:', landsat2015);

/*******************************
  CLIP DATASETS
*******************************/

// Clip Landsat image composite within box extents
var clip2015 = landsat2015.clip(box);

// Split/clip image composite into regions for exporting
var clip2015top = landsat2015.clip(boxTOP); // Top region for export
var clip2015mid = landsat2015.clip(boxMID); // Middle region for export
var clip2015low = landsat2015.clip(boxLOW); // Lower region for export

/*******************************
  DISPLAY RGB COMPOSITES
*******************************/

// Display full image composite
Map.addLayer(clip2015, {bands: ['B6', 'B5', 'B4'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 2015 Landsat', false);

// Display split image composites
Map.addLayer(clip2015top, {bands: ['B6', 'B5', 'B4'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 2015 Top', true);
Map.addLayer(clip2015mid, {bands: ['B6', 'B5', 'B4'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 2015 Mid', true);
Map.addLayer(clip2015low, {bands: ['B6', 'B5', 'B4'], min: 0.05, max: [0.3,0.4,0.4], gamma: 1.6}, 'RGB 2015 Low', true);

/*******************************
  PROCESS IMAGE ASSET
*******************************/

// Export full image composite to drive
// Export.image(clip2015, 'Tanintharyi_2015_Landsat', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': box}); // OK
// NOTE: image exports are split automatically due to cap on file size on export

// Export split image composite to drive
Export.image(clip2015top, 'TNI_2015_Landsat_TOP', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxTOP});
Export.image(clip2015mid, 'TNI_2015_Landsat_MID', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxMID});
Export.image(clip2015low, 'TNI_2015_Landsat_LOW', {'scale':30, 'maxPixels':1e13, 'crs':crs, 'region': boxLOW});
```

2. Scripts for image pre-processing, classification, and accuracy assessment

Script for Set A 1995

```
/*
 * This script executes an image classification procedure on a 1995 image stack consisting of the following datasets:
 *
 * Landsat-5, comprised of 7 bands and 5 indices (total = 12)
 * JERS-1, comprised of 1 polarisation and 8 GLCM texture measures (total = 9)
 *
 * Note that the Landsat images consist of a cloud-masked 1994-1995 composite. The JERS-1 mosaic tiles were pre-
 * processed using ESA SNAP Toolbox software, which included mosaicking individual tiles into a regional mosaic, speckle
 * filtering, and calculating normalised radar cross-section.
 *
 * The GLCM texture measures were computed from the HH sigma0 channel of the JERS-1 images (to match with HH channel
 * of PALSAR data. For the single polarisation channel, 8 texture measures were computed using a 3x3 kernel (note:
 * kernel size=1 was used to facilitate computation in Google Earth Engine). The list of texture measures that were
 * computed include:
 *
 * ASM: angular second moment
 * CONTRAST: contrast
 * CORR: correlation
 * DISS: dissimilarity
 * ENT: entropy
 * IDM: inverse difference moment (or homogeneity)
 * SAVG: sum average
 * VAR: variance
 *
 * A total of 21 layers were classified using Random Forest (RF) algorithm with 9 land cover types found in
 * Tanintharyi, Myanmar. Accuracy assessments were done, of which overall accuracy, the error matrix, Kappa statistic,
 * consumer's and producer's accuracies, and F1 score were computed.
 *
 */

/*******************************
  DEFINE RANDOM SEED
*******************************/

var seed = 2018;


/*******************************
  DEFINE EXTENT AND VIEW
*******************************/

// Set center of map view
Map.setCenter(98.64212,12.40975, 11); // Zoom in and center display to Tanintharyi Region, Myanmar

// Define and display box extents covering Tanintharyi region
var box = ee.Geometry.Rectangle(97.0,16.0, 100.0,9.0);


/*******************************
  LOAD DATASETS
*******************************/

// LANDSAT

// Load Landsat image assets
var tni1995top = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_1995_Landsat_TOP');
var tni1995mid = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_1995_Landsat_MID');
```

```
var tni1995low = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_1995_Landsat_LOW');

// Mosaic Landsat image assets
var composite1995 = ee.ImageCollection([tni1995top, tni1995mid, tni1995low]).mosaic()
                              .select([0,1,2,3,4,5,6,7],['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'YR']);

// Calculate indices from Landsat bands
var ndvi1995 = composite1995.normalizedDifference(['B4', 'B3']); // Normalised Difference Vegetation Index (NDVI)
var lswi1995 = composite1995.normalizedDifference(['B4', 'B5']); // Land Surface Water Index (LSWI)
var ndti1995 = composite1995.normalizedDifference(['B5', 'B7']); // Normalised Difference Till Index (NDTI)
var stvi1995 = composite1995.expression('((b("B5") - b("B3")) / (b("B5") + b("B3") + 0.1)) * (1.1 - (b("B7") / 2))'); // Soil-
Adjusted Total Vegetation Index (SATVI)
var evi1995  = composite1995.expression('2.5 * ((b("B4") - b("B3")) / (b("B4") + 6 * b("B3") - 7.5 * b("B1") + 1))'); // Enhanced
Vegetation Index

// JERS-1

// Load JERS-1 image assets
var hh1995r = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_1995_JERS1_HH'); // 19955 HH polarisation Raw
var hh1995s = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_1995_JERS1_HH_Sigma0'); // 1995 HH polarisation Sigma0

// Rescale floating point to integer
var scaledhh1995 = hh1995s.expression('1000*b("b1")').int32();

// Rename rescaled Sigma0 channels
scaledhh1995 = scaledhh1995.rename('HH');

// Calculate GLCM texture measures
var textureMeasures = ['HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];
var glcm1995 = scaledhh1995.glcmTexture({size: 1, average: true }).select(textureMeasures);  // 3x3 kernel


/*******************************
  CREATE COMPOSITE STACK
*******************************/

// Rename band filenames in metadata
ndvi1995 = ndvi1995.rename('NDVI');
lswi1995 = lswi1995.rename('LSWI');
ndti1995 = ndti1995.rename('NDTI');
stvi1995 = stvi1995.rename('SATVI');
evi1995 = evi1995.rename('EVI');
hh1995s = hh1995s.rename('HH');

// Create image collection from images
var stackCombined1995 =
composite1995.addBands(ndvi1995).addBands(lswi1995).addBands(ndti1995).addBands(stvi1995).addBands(evi1995)
                    .addBands(hh1995s).addBands(glcm1995);
var bandsCombined = ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                'HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];

var stackLandsat1995 =
composite1995.addBands(ndvi1995).addBands(lswi1995).addBands(ndti1995).addBands(stvi1995).addBands(evi1995);
var bandsLandsat = ['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',];

var stackSAR1995 = hh1995s.addBands(glcm1995);
var bandsSAR = ['HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];


/*******************************
  DEFINE REGIONS OF INTEREST
*******************************/
```

```
// Merge regions of interest
var tanintharyiROI = Forest.merge(Mangrove).merge(PalmOilMature).merge(RubberMature).merge(ShrubOrchard)
                        .merge(RicePaddy).merge(BuiltUp).merge(BareSoil).merge(Water);

// Initialise random column and values for ROI feature collection
tanintharyiROI = tanintharyiROI.randomColumn('random1', seed);

var train = tanintharyiROI.filter(ee.Filter.lte('random1', 0.7));
var test  = tanintharyiROI.filter(ee.Filter.gt('random1', 0.7));

Map.addLayer(train, {'color': '000000'}, 'ROI Train', true);
Map.addLayer(test,  {'color': 'FF0000'}, 'ROI Test', true);

// Initialise random column and values for ROI feature collection
train = train.randomColumn('random', seed);
test  = test.randomColumn('random', seed);

// Create training ROIs from the image dataset
var roiTrainCombined = stackCombined1995.select(bandsCombined).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainLandsat = stackLandsat1995.select(bandsLandsat).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainSAR = stackSAR1995.select(bandsSAR).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 25
});

// Create testing ROIs from the image dataset
var roiTestCombined = stackCombined1995.select(bandsCombined).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestLandsat = stackLandsat1995.select(bandsLandsat).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestSAR = stackSAR1995.select(bandsSAR).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 25
});

// Partition the regions of interest into training and testing areas
var trainingCombined = roiTrainCombined.filter(ee.Filter.lte('random', 0.7));
var trainingLandsat = roiTrainLandsat.filter(ee.Filter.lte('random', 0.7));
var trainingSAR = roiTrainSAR.filter(ee.Filter.lte('random', 0.7));
var testingCombined = roiTestCombined.filter(ee.Filter.lte('random', 0.7));
var testingLandsat = roiTestLandsat.filter(ee.Filter.lte('random', 0.7));
var testingSAR = roiTestSAR.filter(ee.Filter.lte('random', 0.7));

// Print number of regions of interest for training and testing at the console
print('Training, Combined, n =', trainingCombined.aggregate_count('.all'));
```

```
print('Testing, Combined, n =', testingCombined.aggregate_count('.all'));
print('Training, Landsat, n =', trainingLandsat.aggregate_count('.all'));
print('Testing, Landsat, n =', testingLandsat.aggregate_count('.all'));
print('Training, SAR, n =', trainingSAR.aggregate_count('.all'));
print('Testing, SAR, n =', testingSAR.aggregate_count('.all'));


/*******************************
  EXECUTE CLASSIFICATION
*******************************/

// COMBINED LANDSAT+SAR

// Classification using Random Forest algorithm
var classifierCombined = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingCombined.select(['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                     'HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                                     'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsCombined
});

// Classify the validation data
var validationCombined = testingCombined.classify(classifierCombined);

// Calculate accuracy metrics
var emC = validationCombined.errorMatrix('ClassID', 'classification'); // Error matrix
var oaC = emC.accuracy(); // Overall accuracy
var ksC = emC.kappa(); // Kappa statistic
var uaC = emC.consumersAccuracy().project([1]); // Consumer's accuracy
var paC = emC.producersAccuracy().project([0]); // Producer's accuracy
var f1C = (uaC.multiply(paC).multiply(2.0)).divide(uaC.add(paC)); // F1-statistic

print('Error Matrix, Combined: ', emC);
print('Overall Accuracy, Combined: ', oaC);
print('Kappa Statistic, Combined: ', ksC);
print('User\'s Accuracy (rows), Combined:', uaC);
print('Producer\'s Accuracy (cols), Combined:', paC);
print('F1 Score, Combined: ', f1C);

// Classify the image Random Forest algorithm
var classifiedCombined = stackCombined1995.select(bandsCombined).classify(classifierCombined);


// LANDSAT ONLY

// Classification using Random Forest algorithm
var classifierLandsat = ee.Classifier.randomForest(100,0,10,0.5,false,2017).train({
  features: trainingLandsat.select(['B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                    'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsLandsat
});

// Classify the validation data
var validationLandsat = testingLandsat.classify(classifierLandsat);

// Calculate accuracy metrics
var emL = validationLandsat.errorMatrix('ClassID', 'classification'); // Error matrix
var oaL = emL.accuracy(); // Overall accuracy
var ksL = emL.kappa(); // Kappa statistic
var uaL = emL.consumersAccuracy().project([1]); // Consumer's accuracy
```

```
var paL = emL.producersAccuracy().project([0]); // Producer's accuracy
var f1L = (uaL.multiply(paL).multiply(2.0)).divide(uaL.add(paL)); // F1-statistic

print('Error Matrix, Landsat: ', emL);
print('Overall Accuracy, Landsat: ', oaL);
print('Kappa Statistic, Landsat: ', ksL);
print('User\'s Accuracy (rows), Landsat:', uaL);
print('Producer\'s Accuracy (cols), Landsat:', paL);
print('F1 Score, Landsat: ', f1L);

// Classify the image Random Forest algorithm
var classifiedLandsat = stackLandsat1995.select(bandsLandsat).classify(classifierLandsat);


// SAR ONLY

// Classification using Random Forest algorithm
var classifierSAR = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingSAR.select(['HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                          'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsSAR
});

// Classify the validation data
var validationSAR = testingSAR.classify(classifierSAR);

// Calculate accuracy metrics
var emS = validationSAR.errorMatrix('ClassID', 'classification'); // Error matrix
var oaS = emS.accuracy(); // Overall accuracy
var ksS = emS.kappa(); // Kappa statistic
var uaS = emS.consumersAccuracy().project([1]); // Consumer's accuracy
var paS = emS.producersAccuracy().project([0]); // Producer's accuracy
var f1S = (uaS.multiply(paS).multiply(2.0)).divide(uaS.add(paS)); // F1-statistic

print('Error Matrix, SAR: ', emS);
print('Overall Accuracy, SAR: ', oaS);
print('Kappa Statistic, SAR: ', ksS);
print('User\'s Accuracy (rows), SAR:', uaS);
print('Producer\'s Accuracy (cols), SAR:', paS);
print('F1 Score, SAR: ', f1S);

// Classify the image Random Forest algorithm
var classifiedSAR = stackSAR1995.select(bandsSAR).classify(classifierSAR);


/*******************************
  FILTER CLASSIFICATION
*******************************/

// Perform a mode filter on the classified image
var filteredCombined = classifiedCombined.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
var filteredLandsat = classifiedLandsat.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
var filteredSAR = classifiedSAR.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
```

```
});


/*******************************
  DISPLAY RGB COMPOSITES
*******************************/

Map.addLayer(composite1995, {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.3}, 'RGB 1995 Landsat', false);
Map.addLayer(hh1995s, {min: -18.8356, max: -3.64452}, 'HH 1995 JERS-1', false);


/*******************************
  DISPLAY CLASSIFICATION
*******************************/

// Create a palette for displaying the classified images
var palette = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];

// Display classified image
Map.addLayer(classifiedCombined, {min: 0, max: 8, palette: palette}, '1995 Classification, Combined', true);
Map.addLayer(classifiedLandsat,  {min: 0, max: 8, palette: palette}, '1995 Classification, Landsat',  true);
Map.addLayer(classifiedSAR,      {min: 0, max: 8, palette: palette}, '1995 Classification, JERS-1',  true);

// Display classified mode image
Map.addLayer(filteredCombined,  {min: 0, max: 8, palette: palette}, '1995 Mode, Combined', false);
Map.addLayer(filteredLandsat,   {min: 0, max: 8, palette: palette}, '1995 Mode, Landsat', false);
Map.addLayer(filteredSAR,       {min: 0, max: 8, palette: palette}, '1995 Mode, JERS-1', false);

// Define classification legend
var colors = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];
var names = ["Forest",
             "Mangrove",
             "Oil palm (mature)",
             "Rubber (mature)",
             "Shrub/orchard",
             "Rice paddy",
             "Built-up area",
             "Bare soil/ground",
             "Water"];
var legend = ui.Panel({style: {position: 'bottom-left'}});
legend.add(ui.Label({
  value: "Land Cover Classification",
  style: {
    fontWeight: 'bold',
    fontSize: '16px',
    margin: '0 0 4px 0',
    padding: '0px'
  }
}));

// Iterate classification legend entries
var entry;
for (var x = 0; x<9; x++){
  entry = [
    ui.Label({style:{color:colors[x],margin: '0 0 4px 0'}, value: '\u2588'}),
    ui.Label({
      value: names[x],
      style: {
        margin: '0 0 4px 4px'
      }
    })
  ];
```

```
  legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));
}

// Display classification legend
Map.add(legend);


/*******************************
  EXPORT CLASSIFIED IMAGES
*******************************/

// Classified images for Combined Landsat+SAR

Export.image.toDrive({
  image: classifiedCombined.uint8(),
  description: 'Classification_SetA_1995_LJ_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredCombined.uint8(),
  description: 'Classification_Mode_SetA_1995_LJ_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for Landsat only

Export.image.toDrive({
  image: classifiedLandsat.uint8(),
  description: 'Classification_SetA_1995_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredLandsat.uint8(),
  description: 'Classification_Mode_SetA_1995_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for SAR only

Export.image.toDrive({
  image: classifiedSAR.uint8(),
  description: 'Classification_SetA_1995_J_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
});
```

```
Export.image.toDrive({
  image: filteredSAR.uint8(),
  description: 'Classification_Mode_SetA_1995_J_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
  });


/******************************
  EXPORT TABLES
******************************/

// Export computed statistics for all regions of interest as a csv file

// For combined Landsat+SAR
Export.table.toDrive(trainingCombined, 'Training_SetA_1995_LJ_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationCombined, 'Validation_SetA_1995_LJ_30m_RF', 'Google Earth Engine');

// For Landsat only
Export.table.toDrive(trainingLandsat, 'Training_SetA_1995_L_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationLandsat, 'Validation_SetA_1995_L_30m_RF', 'Google Earth Engine');

// For SAR only
Export.table.toDrive(trainingSAR, 'Training_SetA_1995_J_25m_RF', 'Google Earth Engine');
Export.table.toDrive(validationSAR, 'Validation_SetA_1995_J_25m_RF', 'Google Earth Engine');
```

Script for Set A 2015

```
/*
 * This script executes an image classification procedure on a 2015 image stack consisting of the following datasets:
 *
 * Landsat-8, comprised of 7 bands and 5 indices (total = 12)
 * ALOS/PALSAR-2, comprised of 1 polarisation and 8 GLCM texture measures (total = 9)
 *
 * Note that the Landsat images consist of a cloud-masked 2015-2016 composite. The PALSAR mosaic tiles were pre-
 * processed using ESA SNAP Toolbox software, which included mosaicking individual tiles into a regional mosaic, speckle
 * filtering, and calculating normalised radar cross-section.
 *
 * The GLCM texture measures were computed from the HH sigma0 channel of the PALSAR images (to match with HH channel
 * of JERS-1 data. For the single polarisation channel, 8 texture measures were computed using a 3x3 kernel (note:
 * kernel size=1 was used to facilitate computation in Google Earth Engine). The list of texture measures that were
 * computed include:
 *
 * ASM: angular second moment
 * CONTRAST: contrast
 * CORR: correlation
 * DISS: dissimilarity
 * ENT: entropy
 * IDM: inverse difference moment (or homogeneity)
 * SAVG: sum average
 * VAR: variance
 *
 * In addition, indices from the raw dual-polarised PALSAR images were computed including: HH/HV ratio, HV/HH ratio,
 * average, difference, normalised difference index, and NL index.
 *
 * A total of 21 layers were classified using Random Forest (RF) algorithm with 9 land cover types found in
 * Tanintharyi, Myanmar. Accuracy assessments were done, of which overall accuracy, the error matrix, Kappa statistic,
 * consumer's and producer's accuracies, and F1 score were computed.
 *
 */

/*******************************
  DEFINE RANDOM SEED
*******************************/

var seed = 2018;


/*******************************
  DEFINE EXTENT AND VIEW
*******************************/

// Set center of map view
Map.setCenter(98.64212,12.40975, 11); // Zoom in and center display to Tanintharyi Region, Myanmar

// Define and display box extents covering Tanintharyi region
var box = ee.Geometry.Rectangle(97.0,16.0, 100.0,9.0);


/*******************************
  LOAD DATASETS
*******************************/

// LANDSAT

// Load Landsat image assets
var tni2015top = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_TOP');
```

```
var tni2015mid = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_MID');
var tni2015low = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_LOW');

// Mosaic Landsat image assets
var composite2015 = ee.ImageCollection([tni2015top, tni2015mid, tni2015low]).mosaic()
                              .select([0,1,2,3,4,5,6,7],['B2', 'B3', 'B4', 'B5', 'B6', 'B10', 'B7', 'YR']);

// Calculate indices from Landsat bands
var ndvi2015 = composite2015.normalizedDifference(['B5', 'B4']); // Normalised Difference Vegetation Index (NDVI)
var lswi2015 = composite2015.normalizedDifference(['B5', 'B6']); // Land Surface Water Index (LSWI)
var ndti2015 = composite2015.normalizedDifference(['B6', 'B7']); // Normalised Difference Till Index (NDTI)
var stvi2015 = composite2015.expression('((b("B6") - b("B4")) / (b("B6") + b("B4") + 0.1)) * (1.1 - (b("B7") / 2))'); // Soil-
Adjusted Total Vegetation Index (SATVI)
var evi2015  = composite2015.expression('2.5 * ((b("B5") - b("B4")) / (b("B5") + 6 * b("B4") - 7.5 * b("B2") + 1))'); // Enhanced
Vegetation Index

// ALOS/PALSAR

// Load PALSAR image assets
var hh2015r = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HH'); // 2015 HH polarisation Raw
var hh2015s = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HH_Sigma0'); // 2015 HH polarisation
Sigma0

// Rescale floating point to integer
var scaledhh2015 = hh2015s.expression('1000*b("b1")').int32();

// Rename rescaled Sigma0 channels
scaledhh2015 = scaledhh2015.rename('HH');

// Calculate GLCM texture measures
var textureMeasures = ['HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];
var glcm2015 = scaledhh2015.glcmTexture({size: 1, average: true }).select(textureMeasures);  // 3x3 kernel


/*******************************
  CREATE COMPOSITE STACK
*******************************/

// Rename band filenames in metadata
ndvi2015 = ndvi2015.rename('NDVI');
lswi2015 = lswi2015.rename('LSWI');
ndti2015 = ndti2015.rename('NDTI');
stvi2015 = stvi2015.rename('SATVI');
evi2015 = evi2015.rename('EVI');
hh2015s = hh2015s.rename('HH');


// Create image collection from images
var stackCombined2015 =
composite2015.addBands(ndvi2015).addBands(lswi2015).addBands(ndti2015).addBands(stvi2015).addBands(evi2015)
                              .addBands(hh2015s).addBands(glcm2015);
var bandsCombined = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                     'HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];

var stackLandsat2015 =
composite2015.addBands(ndvi2015).addBands(lswi2015).addBands(ndti2015).addBands(stvi2015).addBands(evi2015);
var bandsLandsat = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI'];

var stackSAR2015 = hh2015s.addBands(glcm2015);
var bandsSAR = ['HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss'];
```

```
/*******************************
  DEFINE REGIONS OF INTEREST
*******************************/

// Merge regions of interest
var tanintharyiROI = Forest.merge(Mangrove).merge(PalmOilMature).merge(RubberMature).merge(ShrubOrchard)
                     .merge(RicePaddy).merge(BuiltUp).merge(BareSoil).merge(Water);

// Initialise random column and values for ROI feature collection
tanintharyiROI = tanintharyiROI.randomColumn('random1', seed);

var train = tanintharyiROI.filter(ee.Filter.lte('random1', 0.7));
var test  = tanintharyiROI.filter(ee.Filter.gt('random1', 0.7));

Map.addLayer(train, {'color': '000000'}, 'ROI Train', true);
Map.addLayer(test,  {'color': 'FF0000'}, 'ROI Test', true);

// Initialise random column and values for ROI feature collection
train = train.randomColumn('random', seed);
test  = test.randomColumn('random', seed);

// Create training ROIs from the image dataset
var roiTrainCombined = stackCombined2015.select(bandsCombined).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainLandsat = stackLandsat2015.select(bandsLandsat).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainSAR = stackSAR2015.select(bandsSAR).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 25
});

// Create testing ROIs from the image dataset
var roiTestCombined = stackCombined2015.select(bandsCombined).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestLandsat = stackLandsat2015.select(bandsLandsat).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestSAR = stackSAR2015.select(bandsSAR).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 25
});

// Partition the regions of interest into training and testing areas
var trainingCombined = roiTrainCombined.filter(ee.Filter.lte('random', 0.7));
var trainingLandsat = roiTrainLandsat.filter(ee.Filter.lte('random', 0.7));
var trainingSAR = roiTrainSAR.filter(ee.Filter.lte('random', 0.7));
var testingCombined = roiTestCombined.filter(ee.Filter.lte('random', 0.7));
var testingLandsat = roiTestLandsat.filter(ee.Filter.lte('random', 0.7));
var testingSAR = roiTestSAR.filter(ee.Filter.lte('random', 0.7));
```

```
// Print number of regions of interest for training and testing at the console
print('Training, Combined, n =', trainingCombined.aggregate_count('.all'));
print('Testing, Combined, n =', testingCombined.aggregate_count('.all'));
print('Training, Landsat, n =', trainingLandsat.aggregate_count('.all'));
print('Testing, Landsat, n =', testingLandsat.aggregate_count('.all'));
print('Training, SAR, n =', trainingSAR.aggregate_count('.all'));
print('Testing, SAR, n =', testingSAR.aggregate_count('.all'));


/*******************************
  EXECUTE CLASSIFICATION
*******************************/

// COMBINED LANDSAT+SAR

// Classification using Random Forest algorithm
var classifierCombined = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingCombined.select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                     'HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                                     'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsCombined
});

// Classify the validation data
var validationCombined = testingCombined.classify(classifierCombined);

// Calculate accuracy metrics
var emC = validationCombined.errorMatrix('ClassID', 'classification'); // Error matrix
var oaC = emC.accuracy(); // Overall accuracy
var ksC = emC.kappa(); // Kappa statistic
var uaC = emC.consumersAccuracy().project([1]); // Consumer's accuracy
var paC = emC.producersAccuracy().project([0]); // Producer's accuracy
var f1C = (uaC.multiply(paC).multiply(2.0)).divide(uaC.add(paC)); // F1-statistic

print('Error Matrix, Combined: ', emC);
print('Overall Accuracy, Combined: ', oaC);
print('Kappa Statistic, Combined: ', ksC);
print('User\'s Accuracy (rows), Combined:', uaC);
print('Producer\'s Accuracy (cols), Combined:', paC);
print('F1 Score, Combined: ', f1C);

// Classify the image Random Forest algorithm
var classifiedCombined = stackCombined2015.select(bandsCombined).classify(classifierCombined);


// LANDSAT ONLY

// Classification using Random Forest algorithm
var classifierLandsat = ee.Classifier.randomForest(100,0,10,0.5,false,2017).train({
  features: trainingLandsat.select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                    'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsLandsat
});

// Classify the validation data
var validationLandsat = testingLandsat.classify(classifierLandsat);

// Calculate accuracy metrics
var emL = validationLandsat.errorMatrix('ClassID', 'classification'); // Error matrix
```

```
var oaL = emL.accuracy(); // Overall accuracy
var ksL = emL.kappa(); // Kappa statistic
var uaL = emL.consumersAccuracy().project([1]); // Consumer's accuracy
var paL = emL.producersAccuracy().project([0]); // Producer's accuracy
var f1L = (uaL.multiply(paL).multiply(2.0)).divide(uaL.add(paL)); // F1-statistic

print('Error Matrix, Landsat: ', emL);
print('Overall Accuracy, Landsat: ', oaL);
print('Kappa Statistic, Landsat: ', ksL);
print('User\'s Accuracy (rows), Landsat:', uaL);
print('Producer\'s Accuracy (cols), Landsat:', paL);
print('F1 Score, Landsat: ', f1L);

// Classify the image Random Forest algorithm
var classifiedLandsat = stackLandsat2015.select(bandsLandsat).classify(classifierLandsat);


// SAR ONLY

// Classification using Random Forest algorithm
var classifierSAR = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingSAR.select(['HH', 'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                               'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsSAR
});

// Classify the validation data
var validationSAR = testingSAR.classify(classifierSAR);

// Calculate accuracy metrics
var emS = validationSAR.errorMatrix('ClassID', 'classification'); // Error matrix
var oaS = emS.accuracy(); // Overall accuracy
var ksS = emS.kappa(); // Kappa statistic
var uaS = emS.consumersAccuracy().project([1]); // Consumer's accuracy
var paS = emS.producersAccuracy().project([0]); // Producer's accuracy
var f1S = (uaS.multiply(paS).multiply(2.0)).divide(uaS.add(paS)); // F1-statistic

print('Error Matrix, SAR: ', emS);
print('Overall Accuracy, SAR: ', oaS);
print('Kappa Statistic, SAR: ', ksS);
print('User\'s Accuracy (rows), SAR:', uaS);
print('Producer\'s Accuracy (cols), SAR:', paS);
print('F1 Score, SAR: ', f1S);

// Classify the image Random Forest algorithm
var classifiedSAR = stackSAR2015.select(bandsSAR).classify(classifierSAR);


/*******************************
  FILTER CLASSIFICATION
*******************************/

// Perform a mode filter on the classified image
var filteredCombined = classifiedCombined.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
var filteredLandsat = classifiedLandsat.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
```

```
var filteredSAR = classifiedSAR.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});


/*******************************
  DISPLAY RGB COMPOSITES
*******************************/

Map.addLayer(composite2015, {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.3}, 'RGB 2015 Landsat', false);
Map.addLayer(hh2015s, {min: -26.0033, max: -4.57395}, 'HH 2015 PALSAR', false);

/*******************************
  DISPLAY CLASSIFICATION
*******************************/

// Create a palette for displaying the classified images
var palette = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];

// Display classified image
Map.addLayer(classifiedCombined, {min: 0, max: 8, palette: palette}, '2015 Classification, Combined', true);
Map.addLayer(classifiedLandsat,  {min: 0, max: 8, palette: palette}, '2015 Classification, Landsat',  true);
Map.addLayer(classifiedSAR,      {min: 0, max: 8, palette: palette}, '2015 Classification, PALSAR',  true);

// Display classified mode image
Map.addLayer(filteredCombined,  {min: 0, max: 8, palette: palette}, '2015 Mode, Combined', false);
Map.addLayer(filteredLandsat,   {min: 0, max: 8, palette: palette}, '2015 Mode, Landsat', false);
Map.addLayer(filteredSAR,       {min: 0, max: 8, palette: palette}, '2015 Mode, PALSAR', false);

// Define classification legend
var colors = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];
var names = ["Forest",
             "Mangrove",
             "Oil palm (mature)",
             "Rubber (mature)",
             "Shrub/orchard",
             "Rice paddy",
             "Built-up area",
             "Bare soil/ground",
             "Water"];
var legend = ui.Panel({style: {position: 'bottom-left'}});
legend.add(ui.Label({
  value: "Land Cover Classification",
  style: {
    fontWeight: 'bold',
    fontSize: '16px',
    margin: '0 0 4px 0',
    padding: '0px'
  }
}));

// Iterate classification legend entries
var entry;
for (var x = 0; x<9; x++){
  entry = [
    ui.Label({style:{color:colors[x],margin: '0 0 4px 0'}, value: '\u2588'}),
    ui.Label({
      value: names[x],
      style: {
        margin: '0 0 4px 4px'
      }
```

```
    })
  ];
  legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));
}

// Display classification legend
Map.add(legend);


/*******************************
  EXPORT CLASSIFIED IMAGES
*******************************/

// Classified images for Combined Landsat+SAR

Export.image.toDrive({
  image: classifiedCombined.uint8(),
  description: 'Classification_SetA_2015_LP_30m_RF_9CL',
  folder:  'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredCombined.uint8(),
  description: 'Classification_Mode_SetA_2015_LP_30m_RF_9CL',
  folder:  'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for Landsat only

Export.image.toDrive({
  image: classifiedLandsat.uint8(),
  description: 'Classification_SetA_2015_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredLandsat.uint8(),
  description: 'Classification_Mode_SetA_2015_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for SAR only

Export.image.toDrive({
  image: classifiedSAR.uint8(),
  description: 'Classification_SetA_2015_P_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
```

```
});

Export.image.toDrive({
  image: filteredSAR.uint8(),
  description: 'Classification_Mode_SetA_2015_P_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
  });


/*******************************
  EXPORT TABLES
*******************************/

// Export computed statistics for all regions of interest as a csv file

// For combined Landsat+SAR
Export.table.toDrive(trainingCombined, 'Training_SetA_2015_LP_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationCombined, 'Validation_SetA_2015_LP_30m_RF', 'Google Earth Engine');

// For Landsat only
Export.table.toDrive(trainingLandsat, 'Training_SetA_2015_L_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationLandsat, 'Validation_SetA_2015_L_30m_RF', 'Google Earth Engine');

// For SAR only
Export.table.toDrive(trainingSAR, 'Training_SetA_2015_P_25m_RF', 'Google Earth Engine');
Export.table.toDrive(validationSAR, 'Validation_SetA_2015_P_25m_RF', 'Google Earth Engine');
```

Script for Set B 2015

```
/*
 * This script executes an image classification procedure on a 2015 image stack consisting of the following datasets:
 *
 * Landsat-8, comprised of 7 bands and 5 indices (total = 12)
 * ALOS/PALSAR-2, comprised of 2 polarisations, 6 indices, and 16 GLCM texture measures (total = 24)
 *
 * Note that the Landsat images consist of a cloud-masked 2015-2016 composite. The PALSAR mosaic tiles were pre-
 * processed using ESA SNAP Toolbox software, which included mosaicking individual tiles into a regional mosaic, speckle
 * filtering, and calculating normalised radar cross-section.
 *
 * The GLCM texture measures were computed from the HH and HV sigma0 channels of the PALSAR images. For each
 * channel, 8 texture measures were computed comprising a total of 16 texture measures using a 3x3 kernel (note: kernel size=1
 * was used to facilitate computation in Google Earth Engine). The list of texture measures that were computed include:
 *
 * ASM: angular second moment
 * CONTRAST: contrast
 * CORR: correlation
 * DISS: dissimilarity
 * ENT: entropy
 * IDM: inverse difference moment (or homogeneity)
 * SAVG: sum average
 * VAR: variance
 *
 * In addition, indices from the raw dual-polarised PALSAR images were computed including: HH/HV ratio, HV/HH ratio,
 * average, difference, normalised difference index, and NL index.
 *
 * A total of 36 layers were classified using Random Forest (RF) algorithm with 9 land cover types found in
 * Tanintharyi, Myanmar. Accuracy assessments were done, of which overall accuracy, the error matrix, Kappa statistic,
 * consumer's and producer's accuracies, and F1 score were computed.
 *
 */

/*******************************
  DEFINE RANDOM SEED
*******************************/

var seed = 2018;

/*******************************
  DEFINE EXTENT AND VIEW
*******************************/

// Set center of map view
Map.setCenter(98.64212,12.40975, 11); // Zoom in and center display to Tanintharyi Region, Myanmar

// Define and display box extents covering Tanintharyi region
var box = ee.Geometry.Rectangle(97.0,16.0, 100.0,9.0);


/*******************************
  LOAD DATASETS
*******************************/

// LANDSAT

// Load Landsat image assets
var tni2015top = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_TOP');
var tni2015mid = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_MID');
var tni2015low = ee.Image('users/dondealban/Tanintharyi/Landsat_Composite_2/TNI_2015_Landsat_LOW');
```

```
// Mosaic Landsat image assets
var composite2015 = ee.ImageCollection([tni2015top, tni2015mid, tni2015low]).mosaic()
                              .select([0,1,2,3,4,5,6,7],['B2', 'B3', 'B4', 'B5', 'B6', 'B10', 'B7', 'YR']);

// Calculate indices from Landsat bands
var ndvi2015 = composite2015.normalizedDifference(['B5', 'B4']); // Normalised Difference Vegetation Index (NDVI)
var lswi2015 = composite2015.normalizedDifference(['B5', 'B6']); // Land Surface Water Index (LSWI)
var ndti2015 = composite2015.normalizedDifference(['B6', 'B7']); // Normalised Difference Till Index (NDTI)
var stvi2015 = composite2015.expression('((b("B6") - b("B4")) / (b("B6") + b("B4") + 0.1)) * (1.1 - (b("B7") / 2))'); // Soil-
Adjusted Total Vegetation Index (SATVI)
var evi2015  = composite2015.expression('2.5 * ((b("B5") - b("B4")) / (b("B5") + 6 * b("B4") - 7.5 * b("B2") + 1))'); // Enhanced
Vegetation Index

// ALOS/PALSAR

// Load PALSAR image assets
var hh2015r = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HH'); // 2015 HH polarisation Raw
var hv2015r = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HV'); // 2015 HV polarisation Raw
var hh2015s = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HH_Sigma0'); // 2015 HH polarisation
Sigma0
var hv2015s = ee.Image('users/dondealban/Tanintharyi/Tanintharyi_2015_PALSAR_HV_Sigma0'); // 2015 HV polarisation
Sigma0

// Generate indices using raw channels
var rt2015a = hh2015r.divide(hv2015r);
var rt2015b = hv2015r.divide(hh2015r);
var ave2015 = (hh2015r.add(hv2015r)).divide(2);
var dif2015 = hh2015r.subtract(hv2015r);
var ndi2015 = (hh2015r.subtract(hv2015r)).divide(hh2015r.add(hv2015r));
var nli2015 = (hh2015r.multiply(hv2015r)).divide(hh2015r.add(hv2015r));

// Rescale floating point to integer
var scaledhh2015 = hh2015s.expression('1000*b("b1")').int32();
var scaledhv2015 = hh2015s.expression('1000*b("b1")').int32();

// Rename rescaled Sigma0 channels
scaledhh2015 = scaledhh2015.rename('HH');
scaledhv2015 = scaledhv2015.rename('HV');

// Stack rescaled Sigma0 channels
var dual2015 = scaledhh2015.addBands(scaledhv2015);

// Calculate GLCM texture measures
var textureMeasures = ['HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                       'HV_asm', 'HV_contrast', 'HV_corr', 'HV_var', 'HV_idm', 'HV_savg', 'HV_ent', 'HV_diss'];
var glcm2015 = dual2015.glcmTexture({size: 1, average: true }).select(textureMeasures);  // 3x3 kernel


/*******************************
  CREATE COMPOSITE STACK
*******************************/

// Rename band filenames in metadata
ndvi2015 = ndvi2015.rename('NDVI');
lswi2015 = lswi2015.rename('LSWI');
ndti2015 = ndti2015.rename('NDTI');
stvi2015 = stvi2015.rename('SATVI');
evi2015 = evi2015.rename('EVI');
hh2015s = hh2015s.rename('HH');
hv2015s = hv2015s.rename('HV');
rt2015a = rt2015a.rename('RT1');
```

```
rt2015b = rt2015b.rename('RT2');
ave2015 = ave2015.rename('AVE');
dif2015 = dif2015.rename('DIF');
ndi2015 = ndi2015.rename('NDI');
nli2015 = nli2015.rename('NLI');

// Create image collection from images
var stackCombined2015 =
composite2015.addBands(ndvi2015).addBands(lswi2015).addBands(ndti2015).addBands(stvi2015).addBands(evi2015)
                             .addBands(hh2015s).addBands(hv2015s).addBands(rt2015a).addBands(rt2015b)
                             .addBands(ave2015).addBands(dif2015).addBands(ndi2015).addBands(nli2015).addBands(glcm2015);
var bandsCombined = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                'HH', 'HV', 'RT1', 'RT2', 'AVE', 'DIF', 'NDI', 'NLI',
                'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                'HV_asm', 'HV_contrast', 'HV_corr', 'HV_var', 'HV_idm', 'HV_savg', 'HV_ent', 'HV_diss'];

var stackLandsat2015 =
composite2015.addBands(ndvi2015).addBands(lswi2015).addBands(ndti2015).addBands(stvi2015).addBands(evi2015);
var bandsLandsat = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI'];

var stackSAR2015 = hh2015s.addBands(hv2015s).addBands(rt2015a).addBands(rt2015b)
                    .addBands(ave2015).addBands(dif2015).addBands(ndi2015).addBands(nli2015).addBands(glcm2015);
var bandsSAR = ['HH', 'HV', 'RT1', 'RT2', 'AVE', 'DIF', 'NDI', 'NLI',
            'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
            'HV_asm', 'HV_contrast', 'HV_corr', 'HV_var', 'HV_idm', 'HV_savg', 'HV_ent', 'HV_diss'];


/*******************************
  DEFINE REGIONS OF INTEREST
*******************************/

// Merge regions of interest
var tanintharyiROI = Forest.merge(Mangrove).merge(PalmOilMature).merge(RubberMature).merge(ShrubOrchard)
                    .merge(RicePaddy).merge(BuiltUp).merge(BareSoil).merge(Water);

// Initialise random column and values for ROI feature collection
tanintharyiROI = tanintharyiROI.randomColumn('random1', seed);

var train = tanintharyiROI.filter(ee.Filter.lte('random1', 0.7));
var test  = tanintharyiROI.filter(ee.Filter.gt('random1', 0.7));

Map.addLayer(train, {'color': '000000'}, 'ROI Train', true);
Map.addLayer(test,  {'color': 'FF0000'}, 'ROI Test', true);

// Initialise random column and values for ROI feature collection
train = train.randomColumn('random', seed);
test  = test.randomColumn('random', seed);

// Create training ROIs from the image dataset
var roiTrainCombined = stackCombined2015.select(bandsCombined).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainLandsat = stackLandsat2015.select(bandsLandsat).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTrainSAR = stackSAR2015.select(bandsSAR).sampleRegions({
        collection: train,
        properties: ['ClassID', 'random'],
```

```
        scale: 25
});

// Create testing ROIs from the image dataset
var roiTestCombined = stackCombined2015.select(bandsCombined).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestLandsat = stackLandsat2015.select(bandsLandsat).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 30
});
var roiTestSAR = stackSAR2015.select(bandsSAR).sampleRegions({
        collection: test,
        properties: ['ClassID', 'random'],
        scale: 25
});

// Partition the regions of interest into training and testing areas
var trainingCombined = roiTrainCombined.filter(ee.Filter.lte('random', 0.7));
var trainingLandsat = roiTrainLandsat.filter(ee.Filter.lte('random', 0.7));
var trainingSAR = roiTrainSAR.filter(ee.Filter.lte('random', 0.7));
var testingCombined = roiTestCombined.filter(ee.Filter.lte('random', 0.7));
var testingLandsat = roiTestLandsat.filter(ee.Filter.lte('random', 0.7));
var testingSAR = roiTestSAR.filter(ee.Filter.lte('random', 0.7));

// Print number of regions of interest for training and testing at the console
print('Training, Combined, n =', trainingCombined.aggregate_count('.all'));
print('Testing, Combined, n =', testingCombined.aggregate_count('.all'));
print('Training, Landsat, n =', trainingLandsat.aggregate_count('.all'));
print('Testing, Landsat, n =', testingLandsat.aggregate_count('.all'));
print('Training, SAR, n =', trainingSAR.aggregate_count('.all'));
print('Testing, SAR, n =', testingSAR.aggregate_count('.all'));


/*******************************
  EXECUTE CLASSIFICATION
*******************************/

// COMBINED LANDSAT+SAR

// Classification using Random Forest algorithm
var classifierCombined = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingCombined.select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                     'HH', 'HV', 'RT1', 'RT2', 'AVE', 'DIF', 'NDI', 'NLI',
                                     'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                                     'HV_asm', 'HV_contrast', 'HV_corr', 'HV_var', 'HV_idm', 'HV_savg', 'HV_ent', 'HV_diss',
                                     'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsCombined
});

// Classify the validation data
var validationCombined = testingCombined.classify(classifierCombined);

// Calculate accuracy metrics
var emC = validationCombined.errorMatrix('ClassID', 'classification'); // Error matrix
var oaC = emC.accuracy(); // Overall accuracy
var ksC = emC.kappa(); // Kappa statistic
var uaC = emC.consumersAccuracy().project([1]); // Consumer's accuracy
```

```
var paC = emC.producersAccuracy().project([0]); // Producer's accuracy
var f1C = (uaC.multiply(paC).multiply(2.0)).divide(uaC.add(paC)); // F1-statistic

print('Error Matrix, Combined: ', emC);
print('Overall Accuracy, Combined: ', oaC);
print('Kappa Statistic, Combined: ', ksC);
print('User\'s Accuracy (rows), Combined:', uaC);
print('Producer\'s Accuracy (cols), Combined:', paC);
print('F1 Score, Combined: ', f1C);

// Classify the image Random Forest algorithm
var classifiedCombined = stackCombined2015.select(bandsCombined).classify(classifierCombined);


// LANDSAT ONLY

// Classification using Random Forest algorithm
var classifierLandsat = ee.Classifier.randomForest(100,0,10,0.5,false,2017).train({
  features: trainingLandsat.select(['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B10', 'NDVI', 'LSWI', 'NDTI', 'SATVI', 'EVI',
                                    'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsLandsat
});

// Classify the validation data
var validationLandsat = testingLandsat.classify(classifierLandsat);

// Calculate accuracy metrics
var emL = validationLandsat.errorMatrix('ClassID', 'classification'); // Error matrix
var oaL = emL.accuracy(); // Overall accuracycl
var ksL = emL.kappa(); // Kappa statistic
var uaL = emL.consumersAccuracy().project([1]); // Consumer's accuracy
var paL = emL.producersAccuracy().project([0]); // Producer's accuracy
var f1L = (uaL.multiply(paL).multiply(2.0)).divide(uaL.add(paL)); // F1-statistic

print('Error Matrix, Landsat: ', emL);
print('Overall Accuracy, Landsat: ', oaL);
print('Kappa Statistic, Landsat: ', ksL);
print('User\'s Accuracy (rows), Landsat:', uaL);
print('Producer\'s Accuracy (cols), Landsat:', paL);
print('F1 Score, Landsat: ', f1L);

// Classify the image Random Forest algorithm
var classifiedLandsat = stackLandsat2015.select(bandsLandsat).classify(classifierLandsat);


// SAR ONLY

// Classification using Random Forest algorithm
var classifierSAR = ee.Classifier.randomForest(100,0,10,0.5,false,seed).train({
  features: trainingSAR.select(['HH', 'HV', 'RT1', 'RT2', 'AVE', 'DIF', 'NDI', 'NLI',
                                'HH_asm', 'HH_contrast', 'HH_corr', 'HH_var', 'HH_idm', 'HH_savg', 'HH_ent', 'HH_diss',
                                'HV_asm', 'HV_contrast', 'HV_corr', 'HV_var', 'HV_idm', 'HV_savg', 'HV_ent', 'HV_diss',
                                'ClassID']),
  classProperty: 'ClassID',
  inputProperties: bandsSAR
});

// Classify the validation data
var validationSAR = testingSAR.classify(classifierSAR);

// Calculate accuracy metrics
```

```
var emS = validationSAR.errorMatrix('ClassID', 'classification'); // Error matrix
var oaS = emS.accuracy(); // Overall accuracy
var ksS = emS.kappa(); // Kappa statistic
var uaS = emS.consumersAccuracy().project([1]); // Consumer's accuracy
var paS = emS.producersAccuracy().project([0]); // Producer's accuracy
var f1S = (uaS.multiply(paS).multiply(2.0)).divide(uaS.add(paS)); // F1-statistic

print('Error Matrix, SAR: ', emS);
print('Overall Accuracy, SAR: ', oaS);
print('Kappa Statistic, SAR: ', ksS);
print('User\'s Accuracy (rows), SAR:', uaS);
print('Producer\'s Accuracy (cols), SAR:', paS);
print('F1 Score, SAR: ', f1S);

// Classify the image Random Forest algorithm
var classifiedSAR = stackSAR2015.select(bandsSAR).classify(classifierSAR);


/*******************************
  FILTER CLASSIFICATION
*******************************/

// Perform a mode filter on the classified image
var filteredCombined = classifiedCombined.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
var filteredLandsat = classifiedLandsat.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});
var filteredSAR = classifiedSAR.reduceNeighborhood({
  reducer: ee.Reducer.mode(),
  kernel: ee.Kernel.square(1),
});


/*******************************
  DISPLAY RGB COMPOSITES
*******************************/

Map.addLayer(composite2015, {bands: ['B5', 'B4', 'B3'], min: 0, max: 0.3}, 'RGB 2015 Landsat', false);
Map.addLayer(hh2015s.addBands(hv2015s).addBands(rt2015a), {min: [-26.0033, -36.6691, 1], max: [-4.57395, -10.4865, 9]},
'RGB 2015 PALSAR', false);


/*******************************
  DISPLAY CLASSIFICATION
*******************************/

// Create a palette for displaying the classified images
var palette = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];

// Display classified image
Map.addLayer(classifiedCombined, {min: 0, max: 8, palette: palette}, '2015 Classification, Combined', true);
Map.addLayer(classifiedLandsat,  {min: 0, max: 8, palette: palette}, '2015 Classification, Landsat',  true);
Map.addLayer(classifiedSAR,      {min: 0, max: 8, palette: palette}, '2015 Classification, PALSAR',  true);

// Display classified mode image
Map.addLayer(filteredCombined,  {min: 0, max: 8, palette: palette}, '2015 Mode, Combined', false);
Map.addLayer(filteredLandsat,   {min: 0, max: 8, palette: palette}, '2015 Mode, Landsat', false);
Map.addLayer(filteredSAR,       {min: 0, max: 8, palette: palette}, '2015 Mode, PALSAR', false);
```

```
// Define classification legend
var colors = ['246a24', '6666ff', 'ff8000', 'ff00ff', 'ccff66', 'a65400', 'ff0000', 'ffff66', '66ccff'];
var names = ["Forest",
             "Mangrove",
             "Oil palm (mature)",
             "Rubber (mature)",
             "Shrub/orchard",
             "Rice paddy",
             "Built-up area",
             "Bare soil/ground",
             "Water"];
var legend = ui.Panel({style: {position: 'bottom-left'}});
legend.add(ui.Label({
  value: "Land Cover Classification",
  style: {
    fontWeight: 'bold',
    fontSize: '16px',
    margin: '0 0 4px 0',
    padding: '0px'
  }
}));

// Iterate classification legend entries
var entry;
for (var x = 0; x<9; x++){
  entry = [
    ui.Label({style:{color:colors[x],margin: '0 0 4px 0'}, value: '■'}),
    ui.Label({
      value: names[x],
      style: {
        margin: '0 0 4px 4px'
      }
    })
  ];
  legend.add(ui.Panel(entry, ui.Panel.Layout.Flow('horizontal')));
}

// Display classification legend
Map.add(legend);


/*********************************
  EXPORT CLASSIFIED IMAGES
*********************************/

// Classified images for Combined Landsat+SAR

Export.image.toDrive({
  image: classifiedCombined.uint8(),
  description: 'Classification_SetB_2015_LP_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredCombined.uint8(),
  description: 'Classification_Mode_SetB_2015_LP_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
```

```
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for Landsat only

Export.image.toDrive({
  image: classifiedLandsat.uint8(),
  description: 'Classification_SetB_2015_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

Export.image.toDrive({
  image: filteredLandsat.uint8(),
  description: 'Classification_Mode_SetB_2015_L_30m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 30,
  maxPixels: 300000000,
});

// Classified images for SAR only

Export.image.toDrive({
  image: classifiedSAR.uint8(),
  description: 'Classification_SetB_2015_P_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
});

Export.image.toDrive({
  image: filteredSAR.uint8(),
  description: 'Classification_Mode_SetB_2015_P_25m_RF_9CL',
  folder: 'Google Earth Engine',
  region: box,
  scale: 25,
  maxPixels: 500000000,
});


/*******************************
  EXPORT TABLES
*******************************/

// Export computed statistics for all regions of interest as a csv file

// For combined Landsat+SAR
Export.table.toDrive(trainingCombined, 'Training_SetB_2015_LP_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationCombined, 'Validation_SetB_2015_LP_30m_RF', 'Google Earth Engine');

// For Landsat only
Export.table.toDrive(trainingLandsat, 'Training_SetB_2015_L_30m_RF', 'Google Earth Engine');
Export.table.toDrive(validationLandsat, 'Validation_SetB_2015_L_30m_RF', 'Google Earth Engine');

// For SAR only
Export.table.toDrive(trainingSAR, 'Training_SetB_2015_P_25m_RF', 'Google Earth Engine');
Export.table.toDrive(validationSAR, 'Validation_SetB_2015_P_25m_RF', 'Google Earth Engine');
```

3. Box-whisker plots of reflectance/backscatter values for each land cover type and decision trees used for delineation of 1995 ROI polygons

We extracted the image statistics based from the 2015 Landsat and SAR image stacks using the corresponding 2015 regions of interest. We produced box-whisker plots from the image statistics for each of the 36 predictor variables. Using a visual assessment of the plots, we chose predictor variables from either optical or SAR layers that we deemed capable of discriminating Oil Palm Mature, Rubber Mature, and Shrub/Orchard classes. The following predictor variables were chosen: (1) HH-polarisation channel to extract FOR, MNG, OPM, RBM, and SHB and exclude other classes (Figure S1); (2) Band 4 (RED) to extract (a) FOR, (b) MNG, OPM, RBM, and (c) SHB (Figure S2); (3) Band 5 (NIR) to extract (a) MNG, and (b) OPM, RBM (Figure S3); and (4) Band 6 (SWIR1) to extract (a) OPM and (b) RBM (Figure S4).

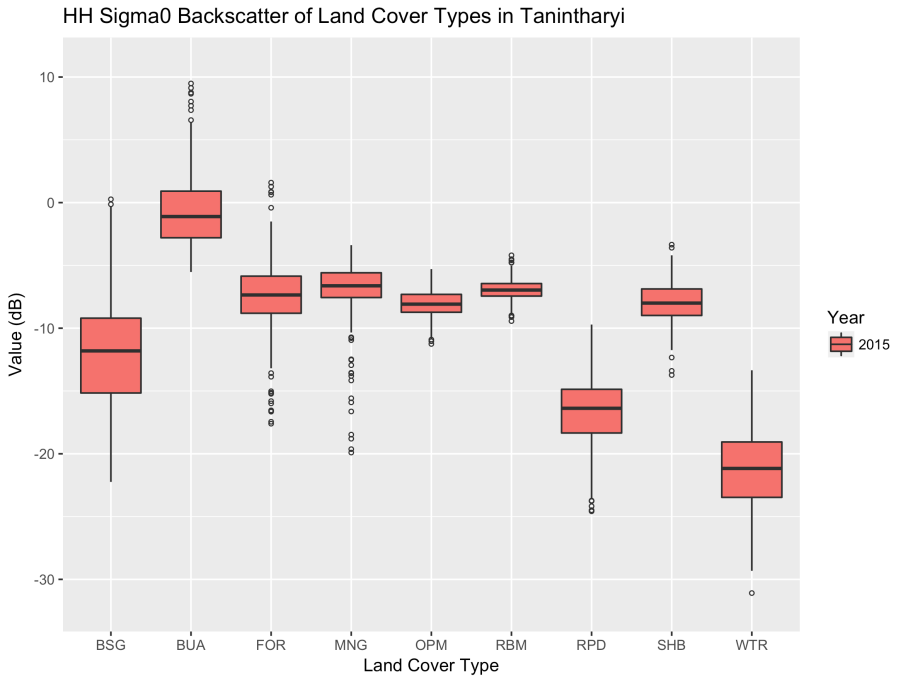

Figure S1. Box-whisker plot for HH-polarisation channel to extract FOR, MNG, OPM, RBM, and SHB and exclude other classes.

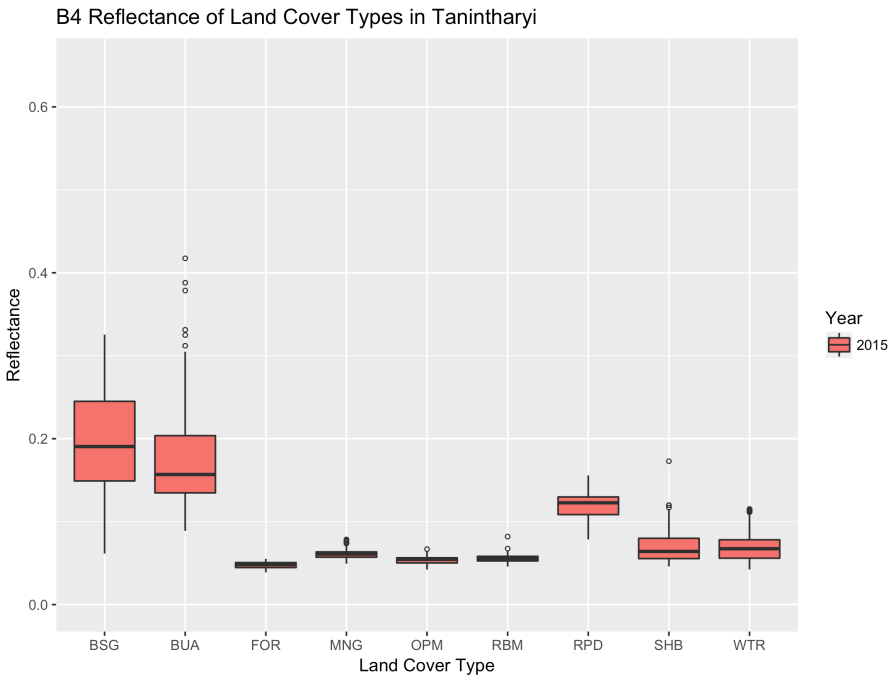

Figure S2. Box-whisker plot for Band 4 (RED) to extract (a) FOR, (b) MNG, OPM, RBM, and (c) SHB classes.

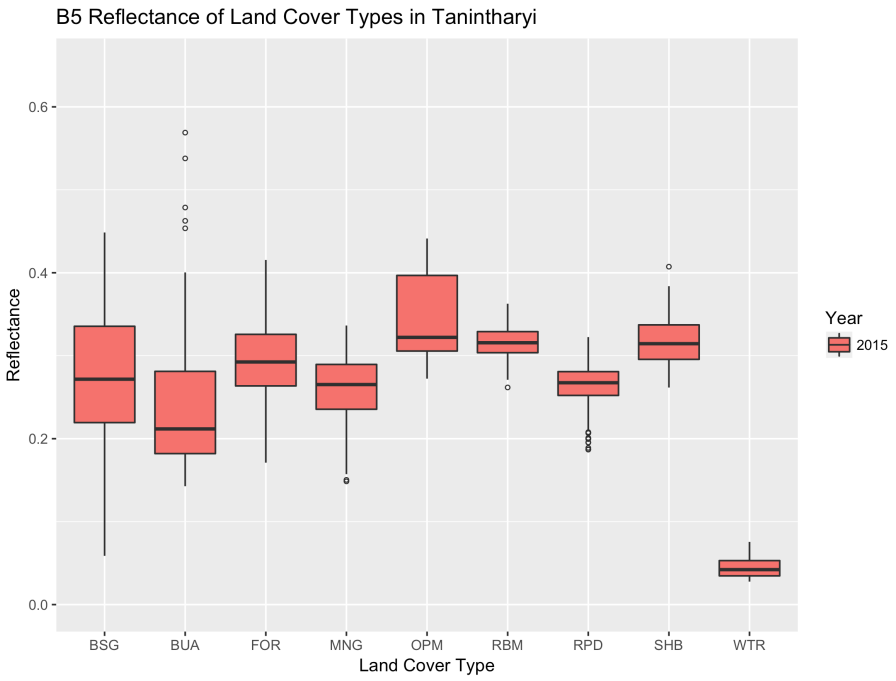

Figure S3. Box-whisker plot for Band 5 (NIR) to extract (a) MNG, and (b) OPM, RBM classes.

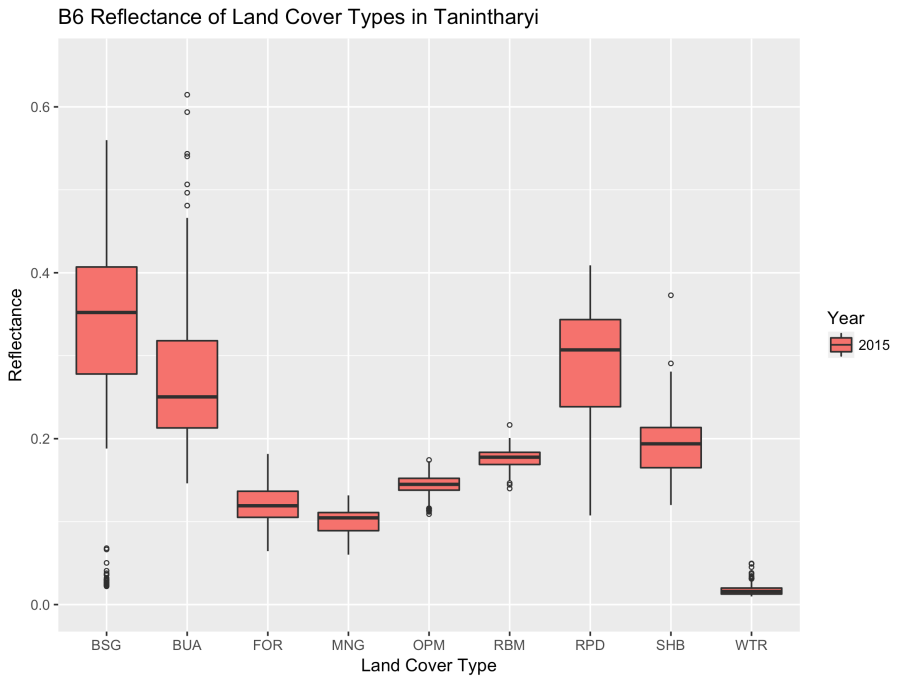

Figure S4. Box-whisker plot for Band 6 (SWIR1) to extract (a) OPM and (b) RBM classes.

After selecting the predictor variables, we subsequently implemented a decision tree classifier to determine thresholds for separating classes in each of the selected predictors. The masks were generated using the corresponding thresholds for each predictor.

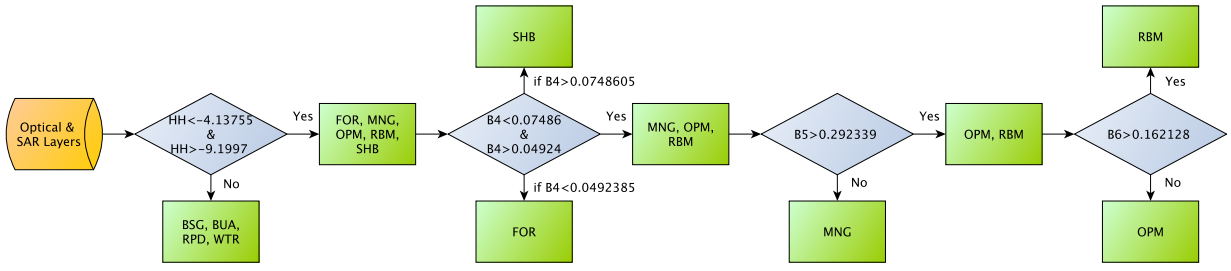

Figure S5. The decision rules showing thresholds for separating classes in each of the selected predictors.

4. User's and producer's accuracies for all classification results

Table S1. User's (UA) and producer's (PA) accuracies of land cover classes from the classification results of Landsat-only, SAR-only, and Landsat + SAR data expressed in terms of unbiased estimates with 95% confidence intervals.

Land Cover	Set A				Set B	
	1995		2015		2015	
	UA	PA	UA	PA	UA	PA
Landsat-only						
Forest	0.9892 ± 0.0000	0.9344 ± 0.0000	1.0000 ± 0.0000	0.9583 ± 0.0000	1.0000 ± 0.0000	0.9583 ± 0.0000
Mangrove	0.5934 ± 0.0002	1.0000 ± 0.0003	0.9403 ± 0.0002	1.0000 ± 0.0003	0.9403 ± 0.0002	1.0000 ± 0.0003
Oil Palm Mature	1.0000 ± 0.0000	0.9136 ± 0.0000	0.6552 ± 0.0002	0.8364 ± 0.0000	0.6552 ± 0.0002	0.8364 ± 0.0000
Rubber Mature	1.0000 ± 0.0000	1.0000 ± 0.0000	0.6731 ± 0.0000	0.9091 ± 0.0000	0.6731 ± 0.0000	0.9091 ± 0.0000
Shrub/Orchard	0.9879 ± 0.0000	0.6522 ± 0.0001	0.7593 ± 0.0001	0.6522 ± 0.0001	0.7593 ± 0.0001	0.6522 ± 0.0001
Rice Paddy	0.6259 ± 0.0002	0.7879 ± 0.0000	0.9963 ± 0.0000	0.8049 ± 0.0000	0.9963 ± 0.0000	0.8049 ± 0.0000
Built-Up Area	0.5616 ± 0.0004	0.9250 ± 0.0000	0.9005 ± 0.0006	0.5079 ± 0.0001	0.9005 ± 0.0006	0.5079 ± 0.0001
Bare Soil/Ground	0.5865 ± 0.0001	0.8687 ± 0.0000	0.7046 ± 0.0002	0.9630 ± 0.0000	0.7046 ± 0.0003	0.9630 ± 0.0000
Water Body	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
SAR-only						
Forest	0.6557 ± 0.0001	0.3651 ± 0.0000	0.3428 ± 0.0002	0.2273 ± 0.0000	0.7789 ± 0.0001	0.5616 ± 0.0000
Mangrove	0.0062 ± 0.0000	0.2353 ± 0.0004	0.5286 ± 0.0002	0.6769 ± 0.0001	0.5801 ± 0.0002	0.6767 ± 0.0002
Oil Palm Mature	0.3680 ± 0.0002	0.2650 ± 0.0000	0.5740 ± 0.0002	0.3750 ± 0.0000	0.7197 ± 0.0002	0.4831 ± 0.0000
Rubber Mature	0.1582 ± 0.0001	0.2342 ± 0.0000	0.6279 ± 0.0001	0.6174 ± 0.0000	0.6162 ± 0.0000	0.7667 ± 0.0000
Shrub/Orchard	0.0196 ± 0.0001	0.0364 ± 0.0000	0.2039 ± 0.0001	0.3182 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
Rice Paddy	0.7092 ± 0.0002	0.2974 ± 0.0000	0.1810 ± 0.0001	0.4595 ± 0.0000	0.1303 ± 0.0000	0.5370 ± 0.0000
Built-Up Area	0.7742 ± 0.0002	0.8676 ± 0.0000	0.9129 ± 0.0002	0.6667 ± 0.0000	0.9265 ± 0.0002	0.7000 ± 0.0000
Bare Soil/Ground	0.2072 ± 0.0001	0.4237 ± 0.0000	0.2355 ± 0.0001	0.4259 ± 0.0000	0.5219 ± 0.0001	0.5441 ± 0.0000
Water Body	0.9746 ± 0.0000	0.9367 ± 0.0000	0.9774 ± 0.0000	0.7179 ± 0.0000	0.9995 ± 0.0000	0.8555 ± 0.0000

Landsat + SAR						
Forest	0.9898 ± 0.0000	0.9661 ± 0.0000	0.9909 ± 0.0000	0.9375 ± 0.0000	0.9928 ± 0.0000	0.9000 ± 0.0000
Mangrove	0.6616 ± 0.0002	1.0000 ± 0.0003	1.0000 ± 0.0000	1.0000 ± 0.0003	1.0000 ± 0.0000	1.0000 ± 0.0003
Oil Palm Mature	1.0000 ± 0.0000	0.9136 ± 0.0000	0.7012 ± 0.0002	0.8393 ± 0.0000	0.6851 ± 0.0002	0.8545 ± 0.0000
Rubber Mature	1.0000 ± 0.0000	1.0000 ± 0.0000	0.6600 ± 0.0000	0.9184 ± 0.0000	0.6796 ± 0.0000	0.9375 ± 0.0000
Shrub/Orchard	0.9722 ± 0.0001	0.7778 ± 0.0000	0.8282 ± 0.0001	0.7143 ± 0.0001	0.8618 ± 0.0001	0.7714 ± 0.0000
Rice Paddy	0.6905 ± 0.0002	0.7465 ± 0.0001	0.9731 ± 0.0001	0.8553 ± 0.0000	0.9334 ± 0.0001	0.9041 ± 0.0000
Built-Up Area	0.8603 ± 0.0004	0.9348 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
Bare Soil/Ground	0.6963 ± 0.0001	0.9239 ± 0.0000	0.8765 ± 0.0002	0.9518 ± 0.0000	0.9453 ± 0.0002	0.9625 ± 0.0000
Water Body	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000

5. Error matrices for all classification results

Table S2. Error matrices expressed in terms of estimated area proportions of the classification results based on Landsat-only data comparing the classified map (rows) to the reference data (columns). Map marginal proportions denoted by W_i.

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set A 1995											
FOR	0.12111	0.00850	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.12961	37506784
MNG	0.00000	0.02303	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02303	6663330
OPM	0.00132	0.00000	0.01397	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.01529	4424256
RBM	0.00000	0.00000	0.00000	0.02790	0.00000	0.00000	0.00000	0.00000	0.00000	0.02790	8073731
SHB	0.00000	0.00728	0.00000	0.00000	0.10916	0.02183	0.00000	0.02911	0.00000	0.16739	48437465
RPD	0.00000	0.00000	0.00000	0.00000	0.00000	0.03988	0.00000	0.01074	0.00000	0.05061	14645718
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00683	0.00055	0.00000	0.00738	2135976
BSG	0.00000	0.00000	0.00000	0.00000	0.00133	0.00200	0.00533	0.05729	0.00000	0.06595	19084875
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51284	0.51284	148404205
Set A 2015											
FOR	0.12837	0.00000	0.00279	0.00000	0.00279	0.00000	0.00000	0.00000	0.00000	0.13395	38761681
MNG	0.00000	0.02043	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02043	5911082
OPM	0.00000	0.00101	0.04638	0.00202	0.00605	0.00000	0.00000	0.00000	0.00000	0.05545	16047072
RBM	0.00000	0.00000	0.00053	0.04756	0.00423	0.00000	0.00000	0.00000	0.00000	0.05232	15140455
SHB	0.00000	0.00000	0.02109	0.02109	0.07907	0.00000	0.00000	0.00000	0.00000	0.12125	35086313
RPD	0.00000	0.00000	0.00000	0.00000	0.01199	0.06597	0.00000	0.00400	0.00000	0.08196	23716613
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00024	0.00261	0.00228	0.00000	0.00513	1484125
BSG	0.00000	0.00029	0.00000	0.00000	0.00000	0.00000	0.00029	0.01497	0.00000	0.01555	4498902
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51397	0.51397	148730097

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set B 2015											
FOR	0.12837	0.00000	0.00279	0.00000	0.00279	0.00000	0.00000	0.00000	0.00000	0.13395	38761681
MNG	0.00000	0.02043	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02043	5911082
OPM	0.00000	0.00101	0.04638	0.00202	0.00605	0.00000	0.00000	0.00000	0.00000	0.05545	16047072
RBM	0.00000	0.00000	0.00053	0.04756	0.00423	0.00000	0.00000	0.00000	0.00000	0.05232	15140455
SHB	0.00000	0.00000	0.02109	0.02109	0.07907	0.00000	0.00000	0.00000	0.00000	0.12125	35086313
RPD	0.00000	0.00000	0.00000	0.00000	0.01199	0.06597	0.00000	0.00400	0.00000	0.08196	23716613
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00024	0.00261	0.00228	0.00000	0.00513	1484125
BSG	0.00000	0.00029	0.00000	0.00000	0.00000	0.00000	0.00029	0.01497	0.00000	0.01555	4498902
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51397	0.51397	148730097

Table S3. Error matrices expressed in terms of estimated area proportions of the classification results based on SAR-only data comparing the classified map (rows) to the reference data (columns). Map marginal proportions denoted by W_i.

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set A 1995											
FOR	0.05072	0.01103	0.00882	0.02867	0.01323	0.00441	0.00662	0.01544	0.00000	0.13893	57893258
MNG	0.00061	0.00049	0.00049	0.00025	0.00000	0.00012	0.00000	0.00012	0.00000	0.00208	868014
OPM	0.00407	0.02035	0.01798	0.02341	0.00000	0.00000	0.00000	0.00204	0.00000	0.06785	28272057
RBM	0.01091	0.00955	0.01410	0.01182	0.00045	0.00091	0.00000	0.00273	0.00000	0.05047	21031053
SHB	0.00000	0.00559	0.00000	0.00076	0.00051	0.00178	0.00000	0.00534	0.00000	0.01398	5823808
RPD	0.00602	0.02675	0.00602	0.00936	0.00936	0.03879	0.00000	0.03411	0.00000	0.13042	54346074
BUA	0.00115	0.00231	0.00000	0.00000	0.00000	0.00000	0.02268	0.00000	0.00000	0.02614	10891824
BSG	0.00386	0.00338	0.00145	0.00048	0.00241	0.00868	0.00000	0.02412	0.01254	0.05692	23717194
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.03248	0.48074	0.51322	213864005
Set A 2015											
FOR	0.02381	0.00476	0.01429	0.00238	0.01905	0.00952	0.00000	0.03095	0.00000	0.10477	43657422
MNG	0.00455	0.02004	0.00000	0.00205	0.00182	0.00000	0.00114	0.00000	0.00000	0.02960	12335217
OPM	0.02155	0.00517	0.04137	0.02155	0.01120	0.00000	0.00000	0.00948	0.00000	0.11032	45973025
RBM	0.00052	0.00364	0.01301	0.04788	0.01249	0.00000	0.00000	0.00000	0.00000	0.07754	32311672
SHB	0.00824	0.00330	0.00165	0.00165	0.01154	0.00000	0.00000	0.00989	0.00000	0.03627	15112107
RPD	0.00000	0.00000	0.00000	0.00000	0.00000	0.02911	0.00000	0.02654	0.00771	0.06336	26401148
BUA	0.00373	0.00099	0.00000	0.00075	0.00050	0.00000	0.01194	0.00000	0.00000	0.01790	7460886
BSG	0.00706	0.00000	0.00176	0.00000	0.00000	0.04587	0.00000	0.04058	0.00000	0.09527	39698660
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.07630	0.00000	0.05484	0.33383	0.46497	193757150

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set B 2015											
FOR	0.08587	0.00000	0.00628	0.01257	0.03770	0.00000	0.00000	0.01047	0.00000	0.15289	63711699
MNG	0.00081	0.01818	0.00101	0.00182	0.00384	0.00000	0.00081	0.00040	0.00000	0.02687	11195164
OPM	0.01537	0.00405	0.04612	0.01699	0.01052	0.00000	0.00000	0.00243	0.00000	0.09547	39784433
RBM	0.00304	0.00121	0.00911	0.05586	0.00364	0.00000	0.00000	0.00000	0.00000	0.07286	30360421
SHB	0.00329	0.00000	0.00000	0.00165	0.00000	0.00000	0.00000	0.00329	0.00000	0.00823	3430510
RPD	0.00000	0.00000	0.00000	0.00000	0.00000	0.01207	0.00000	0.01020	0.00021	0.02248	9367960
BUA	0.00187	0.00166	0.00000	0.00021	0.00062	0.00000	0.01018	0.00000	0.00000	0.01454	6059498
BSG	0.00000	0.00624	0.00156	0.00156	0.00468	0.03430	0.00000	0.05769	0.00000	0.10602	44179431
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.04630	0.00000	0.02604	0.42829	0.50063	208618171

Table S4. Error matrices expressed in terms of estimated area proportions of the classification results based on Landsat + SAR data comparing the classified map (rows) to the reference data (columns). Map marginal proportions denoted by W_i.

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set A 1995											
FOR	0.12806	0.00225	0.00000	0.00000	0.00225	0.00000	0.00000	0.00000	0.00000	0.13256	38358504
MNG	0.00000	0.02184	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02184	6321300
OPM	0.00132	0.00000	0.01397	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.01529	4424448
RBM	0.00000	0.00000	0.00000	0.02603	0.00000	0.00000	0.00000	0.00000	0.00000	0.02603	7531073
SHB	0.00000	0.00893	0.00000	0.00000	0.12497	0.01785	0.00000	0.00893	0.00000	0.16068	46495934
RPD	0.00000	0.00000	0.00000	0.00000	0.00000	0.04426	0.00000	0.01503	0.00000	0.05929	17155955
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00815	0.00057	0.00000	0.00872	2522512
BSG	0.00000	0.00000	0.00000	0.00000	0.00132	0.00198	0.00132	0.05623	0.00000	0.06086	17610223
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51475	0.51475	148956391
Set A 2015											
FOR	0.12121	0.00000	0.00269	0.00000	0.00539	0.00000	0.00000	0.00000	0.00000	0.12929	37412431
MNG	0.00000	0.02241	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02241	6485188
OPM	0.00111	0.00000	0.05218	0.00222	0.00666	0.00000	0.00000	0.00000	0.00000	0.06217	17991296
RBM	0.00000	0.00000	0.00000	0.04225	0.00376	0.00000	0.00000	0.00000	0.00000	0.04601	13313086
SHB	0.00000	0.00000	0.01954	0.01954	0.09771	0.00000	0.00000	0.00000	0.00000	0.13680	39586395
RPD	0.00000	0.00000	0.00000	0.00000	0.00447	0.04837	0.00000	0.00372	0.00000	0.05656	16367511
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00539	0.00000	0.00000	0.00539	1559134
BSG	0.00000	0.00000	0.00000	0.00000	0.00000	0.00134	0.00000	0.02640	0.00000	0.02774	8026234
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51364	0.51364	148635065

	FOR	MNG	OPM	RBM	SHB	RPD	BUA	BSG	WTR	W_i	Total Pixels
Set B 2015											
FOR	0.12371	0.00000	0.00275	0.00000	0.01100	0.00000	0.00000	0.00000	0.00000	0.13746	39777108
MNG	0.00000	0.02156	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02156	6240241
OPM	0.00090	0.00000	0.04245	0.00181	0.00452	0.00000	0.00000	0.00000	0.00000	0.04967	14374168
RBM	0.00000	0.00000	0.00000	0.03937	0.00262	0.00000	0.00000	0.00000	0.00000	0.04200	12153042
SHB	0.00000	0.00000	0.01676	0.01676	0.11313	0.00000	0.00000	0.00000	0.00000	0.14665	42436264
RPD	0.00000	0.00000	0.00000	0.00000	0.00000	0.02802	0.00000	0.00297	0.00000	0.03100	8969589
BUA	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00448	0.00000	0.00000	0.00448	1296011
BSG	0.00000	0.00000	0.00000	0.00000	0.00000	0.00200	0.00000	0.05134	0.00000	0.05334	15434167
WTR	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.51385	0.51385	148695750