Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

Document S2: Source code used in (a) deriving spectral recovery metrics from Landsat time series using LandTrendr, in Google Earth Engine; and (b) reformatting Google Earth Engine outputs for analysis.

(a)-i Deriving spectral recovery metrics

The following source code is written in the JavaScript programming language and designed for use within the Google Earth Engine (GEE) online API environment. The code instructs GEE to process Landsat 5, 7, and 8 data into an annual composite time series, temporally segment these time series using the LandTrendr algorithms, and derive a series of spectral recovery metrics that are exported to the user's list of Assets. The results of this code are then used as input to the code provided in (a)-ii below.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
//
// CODE:    Generating and exporting (to Asset) post-harvest forest spectral recovery
//          metric layers
//
// AUTHORS: Jennifer N. Hird, Remote Sensing Scientist, Alberta Biodiversity Monitoring
//          Institute | Geospatial Research Coordinator, Geography, University of Calgary
//          (Calgary, AB, Canada)
//          email: jennifer.hird@ucalgary.ca
//          *With contributions from the authors of the LandTrendr GEE scripts (see below)
//
// INPUTS:  Geometry of an area of interest
//
// NOTES/COMMENTS: This code creates spectral recovery metric layers, while a separate
//                 code is used to create per-harvest area summary statistics
//
// LAST UPDATED: 2021-10-01
//
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////

//=== Create geometry object encompassing area of interest
var aoi = ee.Geometry.Polygon(
  [[[-118.4335260235648, 54.19740097006359],
  [-118.4335260235648, 53.644236588674055],
  [-116.10167787903355, 53.644236588674055],
  [-116.10167787903355, 54.19740097006359]]], null, false);

// *** From LandTrendr Scripts ***
//////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////
//#
//#                         LANDTRENDR GREATEST MAGNITUDE DISTURBANCE MAPPING
//#
//////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////////////////////////////

// date: 2018-04-19
// author: Zhiqiang Yang  | zhiqiang.yang@oregonstate.edu
//         Justin Braaten | jstnbraaten@gmail.com
//         Robert Kennedy | rkennedy@coas.oregonstate.edu
// website: https://github.com/eMapR/LT-GEE

//////////////////////////////////////////////////////////////////////////////
//##### INPUTS #####
//////////////////////////////////////////////////////////////////////////////

// ----- SET VISUALIZATION PARAMETERS -----
var imageVisParam = {"opacity":1,"bands":["yr1985","yr2001","yr2018"],
"min":473.9825265803124,"max":915.2623445519782,"gamma":1};

// ----- DEFINE YEARS & DATES FOR LANDSAT IMAGE COLLECTION -----
var startYear  = 1984;
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
var endYear    = 2018;
var startDay   = '06-01';
var endDay     = '09-30';

// ----- DEFINE DISTURBANCE MAPPING FILTER PARAMTERS -----
var treeLoss1  = 175;
var treeLoss20 = 200;
var preVal     = 400;
var mmu        = 15;


// ----- DEFINE FUNCTION TO CALCULATE SPECTRAL INDEX FOR LT SEGMENTATION -----
var segIndex = function(img) {
    var index = img.normalizedDifference(['B4', 'B7'])
                   .multiply(1000)
                   .select([0], ['NBR'])
                   .set('system:time_start', img.get('system:time_start'));
    return index ;
};

var distDir = -1; // define the sign of spectral delta for vegetation loss for the segmentation index - NBR
                  // delta is negetive for vegetation loss, so -1 for NBR, 1 for band 5, -1 for NDVI, etc


//----- DEFINE SEGMENTATION PARAMETERS -----
// *Reference: Kennedy, R. E., Yang, Z., & Cohen, W. B. (2010). Detecting trends in
//  forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr—
//  Temporal segmentation algorithms. Remote Sensing of Environment, 114(12), 2897-2910.
// *https://github.com/eMapR/LT-GEE
var run_params = {
  maxSegments:            6,
  spikeThreshold:         0.9,
  vertexCountOvershoot:   3,
  preventOneYearRecovery: true,
  recoveryThreshold:      0.5,
  pvalThreshold:          0.05,
  bestModelProportion:    0.75,
  minObservationsNeeded:  6
};

//############################################################################################################

//############################################################################################################
//##### ANNUAL SR TIME SERIES COLLECTION BUILDING FUNCTIONS #####
//############################################################################################################

//----- MAKE A DUMMY COLLECTOIN FOR FILLTING MISSING YEARS -----
// Make an image collection from an image with 6 bands all set to 0 and then make them masked values
var dummyCollection = ee.ImageCollection([ee.Image([0,0,0,0,0,0]).mask(ee.Image(0))]);

//------ L8 to L7 HARMONIZATION FUNCTION -----
// Slope and intercept citation: Roy, D.P., Kovalskyy, V., Zhang, H.K., Vermote, E.F., Yan, L., Kumar, S.S,
//  Egorov, A., 2016, Characterization of Landsat-7 to Landsat-8 reflective wavelength and normalized
//  difference vegetation index continuity, Remote Sensing of Environment, 185, 57-70.
//  (http://dx.doi.org/10.1016/j.rse.2015.12.024); Table 2 - reduced major axis (RMA) regression
//  coefficients
var harmonizationRoy = function(oli) {
  var slopes = ee.Image.constant([0.9785, 0.9542, 0.9825, 1.0073, 1.0171, 0.9949]);
  var itcp = ee.Image.constant([-0.0095, -0.0016, -0.0022, -0.0021, -0.0030, 0.0029]);
  var y = oli.select(['B2','B3','B4','B5','B6','B7'],['B1', 'B2', 'B3', 'B4', 'B5', 'B7'])
             .resample('bicubic')
             .subtract(itcp.multiply(10000)).divide(slopes)
             .set('system:time_start', oli.get('system:time_start'));
  return y.toShort();
};

//------ RETRIEVE A SENSOR SR COLLECTION FUNCTION -----
var getSRcollection = function(year, startDay, endDay, sensor, aoi) {
  // Define start and end dates
  var mo1 = ee.Number.parse(startDay.substr(0,2));
  var day1 = ee.Number.parse(startDay.substr(3,2));
  var mo2 = ee.Number.parse(endDay.substr(0,2));
  var day2 = ee.Number.parse(endDay.substr(3,2));
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
  var startDate = ee.Date.fromYMD(year, mo1, day1);
  var endDate = ee.Date.fromYMD(year, mo2, day2);
  // Get a landsat collection for given year, day range, and sensor
  var srCollection = ee.ImageCollection('LANDSAT/'+ sensor + '/C01/T1_SR')
                       .filterBounds(aoi)
                       .filterDate(startDate, endDate);
  // Apply the harmonization function to LC08 (if LC08), subset bands, unmask, and resample
  srCollection = srCollection.map(function(img) {
    var dat = ee.Image(
      ee.Algorithms.If(
        sensor == 'LC08',
        harmonizationRoy(img.unmask()),
        img.select(['B1', 'B2', 'B3', 'B4', 'B5', 'B7'])
           .unmask()
           .resample('bicubic')
           .set('system:time_start', img.get('system:time_start'))
      )
    );
    // Make a cloud, cloud shadow, and snow mask from fmask band
    var qa = img.select('pixel_qa');
    var mask = qa.bitwiseAnd(8).eq(0).and(
               qa.bitwiseAnd(16).eq(0)).and(
               qa.bitwiseAnd(32).eq(0));
    // Apply the mask to the image and return it
    return dat.mask(mask);
  });
  return srCollection;
};

//------ FUNCTION TO COMBINE LT05, LE07, & LC08 COLLECTIONS -----
var getCombinedSRcollection = function(year, startDay, endDay, aoi) {
    var lt5 = getSRcollection(year, startDay, endDay, 'LT05', aoi);
    var le7 = getSRcollection(year, startDay, endDay, 'LE07', aoi);
    var lc8 = getSRcollection(year, startDay, endDay, 'LC08', aoi);
    var mergedCollection = ee.ImageCollection(lt5.merge(le7).merge(lc8));
    return mergedCollection;
};

// *** Original Code ***

//=== Function for computing median per summer season from an image collection
var calcSeasonalMedian = function(imgCollection, yr){
  var sysTime = ee.Image(imgCollection.first()).get('system:time_start');
  var seasMed = imgCollection.median();
  var year = ee.Image(ee.Number(yr)).rename("YEAR");
  return seasMed.set('year', yr).set('system:time_start',sysTime).addBands(year).toFloat();
};

//=== Create sequential list of years
var ltYrs = ee.List.sequence(startYear, endYear);

//=== For each year in sequence, create a combined Landsat surface reflectance image collection,
//    and apply seasonal median function to create a single composite
var buildAnnualMedian = ltYrs.map(function(yr){
  var comboCollec = getCombinedSRcollection(yr, startDay, endDay, aoi);
  var medComp = calcSeasonalMedian(comboCollec, yr);
  return medComp;
});

//=== Define function to calculate a spectral index to segment with LT (SOURCE: LandTrendr segIndex function)
var NBRindex = function(img) {
    var index = img.normalizedDifference(['B4', 'B7'])
                   .multiply(1000).multiply(distDir)
                   .select([0], ['NBR'])
                   .set('system:time_start', img.get('system:time_start'))
                   .set('year', img.get('year'));
    return index ;
};

// *** From LandTrendr Scripts ***

//###################################################################################################
//##### UNPACKING LT-GEE OUTPUT STRUCTURE FUNCTIONS #####
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//#############################################################################################################

// ----- FUNCTION TO EXTRACT VERTICES FROM LT RESULTS AND STACK BANDS -----
var getLTvertStack = function(LTresult) {
  // Creat empty arrays, iterate through maximum number of vertices and fill arrays
  var emptyArray = [];
  var vertLabels = [];
  var iString;
  for(var i=1;i<=run_params.maxSegments+1;i++){
    iString = i.toString();
    vertLabels.push("vert_"+iString);
    emptyArray.push(0);
  }
  // Create empty image for filling holes where vertices is < maximum + 1
  var zeros = ee.Image(ee.Array([emptyArray,
                                 emptyArray,
                                 emptyArray]));
  // Set labes for outupt
  var lbls = [['yrs_','src_','fit_'], vertLabels,];
  // Slice out portion of array relating to vertices (series of 1s and 0s)
  var vmask = LTresult.arraySlice(0,3,4);
  // Compile stack of vertices
  var ltVertStack = LTresult.arrayMask(vmask)
                      .arraySlice(0, 0, 3)
                      .addBands(zeros)
                      .toArray(1)
                      .arraySlice(1, 0, run_params.maxSegments+1)
                      .arrayFlatten(lbls, '');
  return ltVertStack;
};

//#############################################################################################################
//##### GREATEST DISTURBANCE EXTRACTION FUNCTIONS #####
//#############################################################################################################

// ----- FUCNTION TO EXTRACT GREATEST DISTURBANCE BETWEEN VERTICES  -----
var extractDisturbance = function(lt, distDir, params, mmu) {
  // Select only the vertices that represents a change
  var vertexMask = lt.arraySlice(0, 3, 4);
  var vertices = lt.arrayMask(vertexMask);
  // Construct segment start and end point years and index values
  var left = vertices.arraySlice(1, 0, -1);
  var right = vertices.arraySlice(1, 1, null);
  var startYear = left.arraySlice(0, 0, 1);
  var startVal = left.arraySlice(0, 2, 3);
  var endYear = right.arraySlice(0, 0, 1);
  var endVal = right.arraySlice(0, 2, 3);
  var dur = endYear.subtract(startYear);
  var mag = endVal.subtract(startVal);
  // Concatenate segment start year, delta, duration, and starting spectral index value to an array
  var distImg = ee.Image.cat([startYear.add(1), mag, dur, startVal.multiply(distDir)]).toArray(0);
  // Sort the segments in the disturbance attribute image delta by spectral index change delta,
  // and flip the delta around so that the greatest delta segment is first in order
  var distImgSorted = distImg.arraySort(mag.multiply(-1));
  // Slice out the first (greatest) delta
  var tempDistImg = distImgSorted.arraySlice(1, 0, 1).unmask(ee.Image(ee.Array([[0],[0],[0],[0]])));
  // *** Added to LandTrendr original codes *** (2019-01-18): Slice out second-greatest delta from array
  var tempSecMag = distImgSorted.arraySlice(1,1,2).unmask(ee.Image(ee.Array([[0],[0],[0],[0]])));
  // Make an image from the array of attributes for the greatest disturbance - year, magnitude, duration, pre-disturbance values
  var finalDistImg = ee.Image.cat(tempDistImg.arraySlice(0,0,1).arrayProject([1]).arrayFlatten([['yod']]),
                                  tempDistImg.arraySlice(0,1,2).arrayProject([1]).arrayFlatten([['mag']]),
                                  tempDistImg.arraySlice(0,2,3).arrayProject([1]).arrayFlatten([['dur']]),
                                  tempDistImg.arraySlice(0,3,4).arrayProject([1]).arrayFlatten([['preval']]),
                                  tempSecMag.arraySlice(0,0,1).arrayProject([1]).arrayFlatten([['yod2nd']]),
                                  //*** Above line added to original LandTrendr codes (2019-01-18) ***
                                  tempSecMag.arraySlice(0,1,2).arrayProject([1]).arrayFlatten([['mag2nd']]));
                                  //*** Above line added to original LandTrendr codes (2019-01-18) ***
  // Apply the filter mask
  finalDistImg = finalDistImg.int16();
  return finalDistImg;
};

// *** Original Code ***
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//=== Build annual seasonal median composite
var medianSeries = ee.ImageCollection(buildAnnualMedian);

//=== Apply NBRindex function to calculate an NBR composite time series
var NBRMedseries = medianSeries.map(NBRindex);

// *** From LandTrendr Scripts ***

// Add the segmentation index as a second band to the annual collection so that the LT fuction will return a
// "fitted" copy of the annual time series for this index
var indexNameFTV = ee.Image(NBRMedseries.first()).bandNames().getInfo()[0]+'_FTV';
var NBRMedseriesFTV = NBRMedseries.map(function(img) {
  return img.addBands(img.select([0],[indexNameFTV]))
                        //.multiply(distDir))
                        .set('system:time_start', img.get('system:time_start'));
});

// *** Original Code ***

// Create multiband image from NBR series image collection
var NBRImgResult = ee.Image(0);
for (var i = startYear; i <= endYear /*<= NBRseries.size()*/; i++){
  var img = ee.Image(NBRMedseries.filterMetadata('year','equals',i).first());
  var bandName = "yr"+i;
  var newBand = img.select([0]).rename(bandName).toFloat().multiply(distDir);
  var NBRImgResult = NBRImgResult.addBands(newBand);
}
NBRImgResult = NBRImgResult.select(NBRImgResult.bandNames().removeAll(["constant"]));

// *** From LandTrendr Scripts ***

//----- RUN LANDTRENDR -----
run_params.timeSeries = NBRMedseriesFTV;
var lt = ee.Algorithms.TemporalSegmentation.LandTrendr(run_params);

//----- EXTRACT SEGMENTATION-FITTED INDEX STACK -----
// Make array of band names
var years = [];
for (var i = startYear; i <= endYear; ++i) years.push(i.toString());

// Create multiband image of fitted index values
var ltFitStack = lt.select([1])
                   .arrayFlatten([years]);

// *** Original Code ***

//=== Use distDir to switch NBR back to positives (* -1)
var ltNBRFitted = ltFitStack.multiply(ee.Image.constant(distDir));

// *** From LandTrendr Scripts ***

//############################################################################################################
//##### RUN THE GREATEST DISTURBANCE EXTRACT FUCTION #####
//############################################################################################################

// Assemble the disturbance extraction parameters
var distParams = {
  tree_loss1: treeLoss1,
  tree_loss20: treeLoss20,
  pre_val: preVal
};

// Run the dist extract function
var distImg = extractDisturbance(lt.select('LandTrendr'), distDir, distParams);

// *** Original Code ***

//############################################################################################################
//#################### CALCULATE POST-HARVEST/DISTURBANCE METRICS AND INFORMATION ######################
//############################################################################################################

//=== Transform multi-band ltFitStack image into image collection (and convert back to positive values, for NBR)
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
var fitCollection = ee.ImageCollection(ltFitStack.bandNames().map(function(band) {
  var img = ltFitStack.select([band]).multiply(distDir).rename('NBR_fitted');
  var yr = ee.Number.parse(band);
  return img.set('year', yr);
}));

//----------------------------------------
// ------- Pre-Disturbance Mean NBR -------

//=== Create image from 'yod' band of distImg
var yodYrImage = distImg.select(["yod"]);

//=== Function to mask out fitted NBR values that are 5 years before disturbance
var preDist5yr = fitCollection.map(function(image) {
  var currYr = ee.Number.parse(image.get('year'));
  var yrImg = ee.Image.constant(currYr);
  var preDistDates = yrImg.lt(yodYrImage).and(yrImg.gte(yodYrImage.subtract(5)));
  var yearBand = ee.Image.constant(currYr).rename('Year').toFloat();
  return image.addBands(yearBand).updateMask(preDistDates);
});

//=== Use reducer (mean) to calculate pre-disturbance mean NBR, per pixel, for 5 years before harvest
var preDistMean = preDist5yr.reduce(ee.Reducer.mean()).select(['NBR_fitted_mean']);

//---------------------------------------
// ------- Post-Disturbance NBR -------

//=== Function to mask out fitted NBR values that are before the end of disturbance
var postDistColl = fitCollection.map(function(image) {
  var currYr = ee.Number.parse(image.get('year'));
  var yrImg = ee.Image.constant(currYr);
  var postDistDates = yrImg.gte(yodYrImage);
  var yearBand = ee.Image.constant(currYr).rename('Year').toFloat();
   return image.addBands(yearBand).updateMask(postDistDates);
});

//=== Create list of band names from ltFitStack multiband image
var listBands = ltFitStack.bandNames();

//=== Map function over each band (which represents fitted NBR for that year),
//     that masks out pixels for which that year that do not equal the value at
//     the biggest change (i.e., yod, calculated by LandTrendr functions)
var yodMaskedStack = listBands.map(function(band){
    var bandYrImg = ee.Image.constant(ee.Number.parse(band));
    var yodMask = bandYrImg.eq(yodYrImage);
    var relvVals = ltFitStack.select([band]).multiply(distDir).toFloat()
    .mask(yodMask).set('year', ee.Number.parse(band)).rename('yodVal');
    return relvVals;
  });

//=== Convert outputted list to image collection
var yodValColl = ee.ImageCollection(yodMaskedStack);

//=== Combine image collection into one image (each pixel should be masked out in
//     every image in the collection save one - the date of YOD)
var yodValImg = yodValColl.reduce(ee.Reducer.median()).rename('endDistVal').toFloat();

//=== Create empty start image filled with 0s, with correct (empty) bands and (empty) year property
var startImg = ee.List([ee.Image(0).select([0],['NBR_fitted'])
                                   .addBands(ee.Image(0).rename('Year'))
                                   .addBands(ee.Image(0).rename('increaseFlag'))
                                   .set('year',0)]);

//=== Function to pass to Iterate(): create a new band where it is '1' when the current
//    NBR value is higher than the previous NBR value
var flagIncrease = function(image, list) {
  var addImg = ee.Image.constant(1000);
  var currNBR = image.select(['NBR_fitted']).add(addImg);
  var prevImg = ee.Image(ee.List(list).get(-1));
  var prevNBR = prevImg.select(['NBR_fitted']).add(addImg);
  var incrFlag = ee.Image.constant(0).where(currNBR.gt(prevNBR), 1).rename('increaseFlag');
  var outImg = image.addBands(incrFlag).set('system:time_start', image.get('system:time_start'))
                    .set('year', image.get('year'));
```

```
  return ee.List(list).add(outImg);
};

//=== Apply iteration to postDistColl
var incrFlagColl = ee.ImageCollection(ee.List(postDistColl.iterate(flagIncrease, startImg)));

//=== Remove first emtpy image from  collection (e.g. 'startImg')
var newFlagColl = incrFlagColl.filterMetadata('year','not_equals',0);

//=== Create multiband image from incrFlagColl image collection
var incrFlagImg = ee.Image(0);
for (var i = startYear; i <= endYear; i++){
  var img = ee.Image(newFlagColl.filterMetadata('year','equals',i).first());
  var bandName = ee.String(img.get('year'));
  var newBand = img.select(['increaseFlag']).rename(bandName);
  var incrFlagImg = incrFlagImg.addBands(newBand);
}
incrFlagImg = incrFlagImg.select(incrFlagImg.bandNames().removeAll(["constant"]));

//=== Use 'increaseFlag' band to mask out 'Year' band in newFlagColl
var flagCollyrMasked = newFlagColl.map(function(image){
  var yrBand = image.select('Year');
  var incrFlag = image.select(['increaseFlag']);
  var flagMask = incrFlag.eq(1);
  var markedYr = ee.Image.constant(endYear + 1).where(incrFlag.eq(1), yrBand).rename('Year');
  return markedYr;
});

//=== Create multiband image from flagCollyrMasked image collection
var incrFlaggedYrImg = ee.Image(0);
for (var i = startYear; i <= endYear; i++){
  var img = ee.Image(flagCollyrMasked.filterMetadata('year','equals',i).first());
  var bandName = ee.String(img.get('year'));
  var newBand = img.select(['Year']).rename(bandName);
  var incrFlaggedYrImg = incrFlaggedYrImg.addBands(newBand);
}
incrFlaggedYrImg = incrFlaggedYrImg.select(incrFlaggedYrImg.bandNames().removeAll(["constant"]));

//=== Reduce collection, calculating minimum value for each band in each pixel,
//     thereby extracting the earliest year where NBR values begin increasing from previous year
var extractEOC = flagCollyrMasked.select(['Year']).reduce(ee.Reducer.min());

//=== Extract the 'Year_min' band - the earliest year in which NBR increase from previous year
//     is detected, and subtract 1 to get year of 'end of change'
var eocYrImage = extractEOC.select(['Year_min']).subtract(ee.Image.constant(1)).rename('eocYear');

//=== Set any pixels where eoc was calculated as earlier than startYear, to the startYear value
var minYear = ee.Image.constant(startYear);
eocYrImage.where(eocYrImage.lt(startYear), minYear);

//=== Map function over each band (which represents fitted NBR for that year),
//     that masks out pixels for which that year does not equal the value at
//     the end of all harvest change (i.e., eoc)
var eocMaskedStack = listBands.map(function(band){
    var bandYrImg = ee.Image.constant(ee.Number.parse(band));
    var eocMask = bandYrImg.eq(eocYrImage);
    var relvVals = ltFitStack.select([band]).multiply(distDir).toFloat()
    .mask(eocMask).set('year', ee.Number.parse(band)).rename('eocVal');
    return relvVals;
  });

//=== Convert outputted list to image collection
var eocValColl = ee.ImageCollection(eocMaskedStack);

//=== Combine image collection into one image (each pixel should be masked out in
//     every image save one)
var eocValImg = eocValColl.reduce(ee.Reducer.median()).rename('endDistVal').toFloat();

//-------------------------------------------------------------------
// ------- Length of Disturbance Period (Between YOD and EOC) -------

//=== Calculate length of time between start of harvest and start of regeneration
var lenDistb = eocYrImage.subtract(yodYrImage).add(1);
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//-------------------------------------------------------------
// ------- Total Change (Relative to Pre-Distb Mean NBR) -------

//=== Subtract value at end of disturbance from pre-disturbanc mean NBR
var totDistbVal = preDistMean.subtract(eocValImg);

//-----------------------------------------
// ------- 5-Yr Percent Regeneration -------

//=== Function to mask out fitted NBR values that are before end of change
var postEocColl = fitCollection.map(function(image) {
  var currYr = ee.Number.parse(image.get('year'));
  var yrImg = ee.Image.constant(currYr);
  var postEoCDates = yrImg.gte(eocYrImage);
  var yearBand = ee.Image.constant(currYr).rename('Year').toFloat();
  return image.addBands(yearBand).updateMask(postEoCDates);
});

//=== Create image for year 5 years post-EOC
var yr5PostEocImg = eocYrImage.add(ee.Image.constant(5));

//=== Map function over postEocColl to extract 5-year percent regeneration,
//    based on EOC dates and values
var regen5yrColl = postEocColl.map(function(image) {
  var yr5flag = ee.Image(0).where(image.select('Year').eq(yr5PostEocImg), 1);
  var yr5vals = ee.Image(0).where(yr5flag.eq(1), image.select('NBR_fitted'));
  var yr5chgCalc = yr5vals.subtract(eocValImg);
  var yr5chg = yr5chgCalc.where(yr5flag.eq(0), 0);
  var regenPercCalc = yr5chg.divide(totDistbVal).multiply(100);
  regenPercCalc.where(yr5vals.eq(0), 0);
  return regenPercCalc.addBands(yr5vals)
                      .addBands(yr5flag)
                      .addBands(yr5chg)
                      .rename(['yr5regen','yr5val','yr5flag','yr5chg'])
                      .set('Year', image.get('year'));
});

//=== Combine collection bands into single images
var regen5yrPerc = regen5yrColl.select('yr5regen').reduce(ee.Reducer.max()).toFloat();

//-----------------------------------
// ------- Total Regeneration -------

//=== Extract index of last band in ltNBRbands
var ltNBRbands = ltNBRFitted.bandNames();
var lastBandIdx = ltNBRbands.length().subtract(1);
var lastBand = ltNBRbands.get(lastBandIdx);

//=== Extract final band in NBR fitted time series
var endTSval = ltNBRFitted.select([lastBand]);

//=== Calculate total percent regeneration (regen / disturbance)
var totPercRegen = endTSval.subtract(eocValImg).divide(totDistbVal)
                           .multiply(100).rename('percTotRegen');

//-------------------------------------------------
// ------- Years to 80% Regeneration (Y2R80%) -------

//=== Create image of NBR values at 80% recovered
var img80perc = totDistbVal.multiply(0.8).add(eocValImg);

//=== Map function over postEocColl to extract years to 80% spectrally regenerated
var y2r80Coll = postEocColl.map(function(image){
  var currYr = image.select('Year');
  var at80regen = ee.Image(0)
    .where(image.select('NBR_fitted').gte(img80perc), currYr)
    .rename('y2r80');
  return at80regen.set('year', image.get('year')).updateMask(at80regen.gt(0));
});

//=== Extract the minimum value per pixel, from the collection
var yrAt80img = y2r80Coll.min();
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//=== Calulate years it took to reach 80% regenerated
var y2r80img = yrAt80img.subtract(eocYrImage).rename('y2r80');
y2r80img.updateMask(y2r80img.gt(0));

//############################################################################
//######## MASKING (OF IRRELEVANT OR INAPPROPRIATE PIXELSFOR ANALYSIS) ########
//############################################################################

//-----------------------------------------
// ------- Minimum Disturbance Masking -------

//=== Create mask for masking out those pixels where the total disturbance is below a set
//    threshold (indicating that there was not reliably identifiable harvest happening in that pixel)
var harvThreshMask = totDistbVal.gte(175); //** in NBR units, based on first-hand observations

//=== Add to maskFlagsImg an image with flags where the harvest threshold mask was used (no harvest)
var reverseHarvMask = harvThreshMask.not().rename("noHarvestFlag");

//-----------------------------------------
// ------- Out-f-Date-Range Masking -------

//=== Create mask to remove those pixels where the greatest change (yod) is withihn 5 years of the start
//    of the time series, and end of change (eoc) is <5 years before the end of the time series
var minCutoff = ee.Image.constant(startYear + 5);
var maxCutoff = ee.Image.constant(endYear - 5);
var relvDatesMask = yodYrImage.gte(minCutoff).and(eocYrImage.lte(maxCutoff));

//=== Create image with flags for pixels masked because of yod date range, and add to maskFlagsImg
var reverseYOODMask = relvDatesMask.not().rename("outOfRangeFlag");

//-----------------------------------------
// ------- No Regeneration Masking -------

//=== Create mask to ID those pixels where EOC equals endYear (2018); EOC was
//     set to endYear in those cases where no post-YOD increases were found
var regenMask = eocYrImage.neq(endYear);

//=== Create image band where pixels with no regeneration are flagged
var noRegenMask = regenMask.not().rename("noRegnFlag");

//-----------------------------------------------
// ------- Mulitiple Disturbance Masking -------

//=== Extract 'mag' and 'yod' for largest and second-largest drops in NBR in the time series,
//    to identify where there may be multiple disturbance events in a time series
var magImage = distImg.select(["mag"]);
var mag2ndImg = distImg.select(["mag2nd"]);
var yod2ndImg = distImg.select(["yod2nd"]);

//=== Create masks for: where 2nd largest disturbance is > 175 NBR, and for dates
//     that are > 3 years before YOD and > 3 years after EOC
var mag2gt175 = mag2ndImg.gt(175);
var yodMinus3yrs = yodYrImage.subtract(ee.Image.constant(3));
var eocPlus3yrs = eocYrImage.add(ee.Image.constant(3));
var yod2ndInRange = yod2ndImg.lt(yodMinus3yrs).or(yod2ndImg.gt(eocPlus3yrs));

//=== Combine above masks to flag pixels in whic the 2nd largest disturbance is > 175 NBR,
//    and is not within 3 years of 1st disturbance (YOD to EOC) - so is separate disturbance
//    Create reverse to mask out pixels with multiple disturbances
var multiDistbPix = mag2gt175.eq(1).and(yod2ndInRange.eq(1)).rename("multiDistbFlag");
var noMultiDistb = multiDistbPix.not().rename("noMultiDistb");

//############################################################################

//############################################################################
//############ EXPORT LAYERS TO ASSET FOR STATS CALCS :: ############
//############################################################################

//---------------------------------------------------------------------------
// ------- Extract original projection CRS and transform for Landsat images -------

//=== Get a landsat collection for given year, day range, and sensor,
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//    and print first image; the CRS and CRS Transform can be found
//    by expanding the information for one of the image's bands
//    The CRS Transform is important for later re-importing assets
//    in a manner that they will be mapped/analyzed using the same projection
//    and transform as they were produced here
var tempColl = ee.ImageCollection('LANDSAT/LT05/C01/T1_SR')
                     .filterBounds(aoi)
                     .filterDate(startYear+'-'+startDay, startYear+'-'+endDay);
// print(tempColl.first());
var origProjCrs = 'EPSG:32611';
var origProjCrsTrsfm = [30,0,520785,0,-30,5996115];

// ------------------------------------------------
// ------- Mask metric layers for export -------

//=== Create final mask from masks of pixels to remove from analysis
var fullMaskImage = ee.Image(1).updateMask(regenMask).updateMask(relvDatesMask)
  .updateMask(harvThreshMask).updateMask(noMultiDistb).selfMask();

//=== Apply mask to layers being exported, and rename
preDistMean = preDistMean.updateMask(fullMaskImage).rename('preDistMean');
yodYrImage = yodYrImage.updateMask(fullMaskImage).rename('yodYrImage');
eocYrImage = eocYrImage.updateMask(fullMaskImage).rename('eocYrImage');
lenDistb = lenDistb.updateMask(fullMaskImage).rename('lenDistb');
totDistbVal = totDistbVal.updateMask(fullMaskImage).rename('totDistbVal');
regen5yrPerc = regen5yrPerc.updateMask(fullMaskImage).rename('regen5yrPerc');
y2r80img = y2r80img.updateMask(fullMaskImage).rename('y2r80img');
totPercRegen = totPercRegen.updateMask(fullMaskImage).rename('totPercRegen');

// ---------------------------------
// ------- Export mask layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: fullMaskImage.clip(aoi),
  assetId: 'maskImage_1984to2018',
  description: 'maskImage_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------
// ------- Export pre-disturbance NBR layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: preDistMean.clip(aoi),
  assetId: 'preDistbNBR_1984to2018',
  description: 'preDistbNBR_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ----------------------------------------
// ------- Export harvest year layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: yodYrImage.clip(aoi),
  assetId: 'harvYrImg_1984to2018',
  description: 'harvYrImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------
// ------- Export start of regeneration year layer -------
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: eocYrImage.clip(aoi),
  assetId: 'regnStartImg_1984to2018',
  description: 'regnStartImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------
// ------- Export time to start of regeneration layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: lenDistb.clip(aoi),
  assetId: 'regnLagImg_1984to2018',
  description: 'regnLagImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------
// ------- Export total NBR spectral disturbance change layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: totDistbVal.clip(aoi),
  assetId: 'totNBRdistbImg_1984to2018',
  description: 'totNBRdistbImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------
// ------- Export 5-year % spectral regeneration -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: regen5yrPerc.clip(aoi),
  assetId: 'regn5yrImg_1984to2018',
  description: 'regn5yrImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------
// ------- Export years to 80 % spectral regeneration layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: y2r80img.clip(aoi),
  assetId: 'yr2Regn80Img_1984to2018',
  description: 'yr2Regn80Img_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------------
// ------- Export end of time series % spectral regeneration layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
```

```
  image: totPercRegen.clip(aoi),
  assetId: 'regn2018Img_1984to2018',
  description: 'regn2018Img_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------------
// ------- Export 'no harvest' flags layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: reverseHarvMask.selfMask().clip(aoi),
  assetId: 'noHarvFlagImg_1984to2018',
  description: 'noHarvFlagImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------------
// ------- Export 'out of date range' flags layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: reverseYOODMask.selfMask().clip(aoi),
  assetId: 'outOfRangeFlagImg_1984to2018',
  description: 'outOfRangeFlagImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------------
// ------- Export 'no regeneration' flags layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: noRegenMask.selfMask().clip(aoi),
  assetId: 'noRegnFlagImg_1984to2018',
  description: 'noRegnFlagImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});

// ------------------------------------------------------------------------
// ------- Export 'multiple disturbance' flags layer -------

//=== Export to Asset List (used crs and crsTransform variables; no scale or dimensions)
Export.image.toAsset({
  image: multiDistbPix.selfMask().clip(aoi),
  assetId: 'multDistbImg_1984to2018',
  description: 'multDistbImg_1984to2018',
  region: aoi,
  crs: origProjCrs,
  crsTransform: origProjCrsTrsfm,
  maxPixels: 1e13
});
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

(a)-i Summarizing per-feature spectral recovery metrics

The following source code is written in the JavaScript programming language and designed for use within the Google Earth Engine (GEE) online API environment. The code imports metric, flag and mask layers that were exported by code (a)-i to the user's list of Assets, and summarizes these for a set of specified polygon features (i.e., harvest area polygons). The code exports the results to a user's Google Drive, as csv files. These csv files are then reformatted using a simple R script, provided in (b) below. If importing your own fusion table (i.e., shapefile), the following code assumes that this table has a unique identifier field labeled "uniqueID".

```
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////
//
// CODE:    Importing (from Assets) spectral regeneration layers and harvest area polygons of interest,
//          extracing per-feature statistics, and exporting to Drive as csv files
//
// AUTHORS: Jennifer N. Hird, Remote Sensing Scientist, Alberta Biodiversity Monitoring
//          Institute | Geospatial Research Coordinator, Geography, University of Calgary
//          (Calgary, AB, Canada)
//          email: jennifer.hird@ucalgary.ca
//
// INPUTS:  Spectral regeneration layers, from Asset list
//          Geometry of an area of interest
//
// NOTES/COMMENTS: This  code is used to create per-harvest area summary statistics, based
//                 on spectral regeneration metrics exported (to Asset) using a separate code
//                 *Note, it is important for features to have a unique identifier field labeled,
//                 as this is used for linking various results for output 'uniqueID'
//
// LAST UPDATED: 2021-10-01
//
////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////

//=== Add geometry object encompassing area of interest (copied from layer export script)
var aoi = ee.Geometry.Polygon(
  [[[-118.4335260235648, 54.19740097006359],
  [-118.4335260235648, 53.644236588674055],
  [-116.10167787903355, 53.644236588674055],
  [-116.10167787903355, 54.19740097006359]]], null, false);
Map.addLayer(aoi);

//#############################################################################
//##### INPUTS #####
//#############################################################################

// -------------------------------------------
// ----- Define harvest areas of interest -----

// //** Option 1:
// //=== Import features from a fusion table asset
// var harvAreas = ee.Image('users/myAccount/myFolder/myFusionTable'); //** Set this to a path to your data

//=== OR:

//** Option 2:
// === Create a feature or set of features

var harvAreas = ee.FeatureCollection([
  ee.Feature(ee.Geometry.Polygon([
    [-117.36696493251391, 53.8910987237269],
    [-117.36610662562914, 53.88927773557097],
    [-117.355635281635, 53.88695080191392],
    [-117.34997045619555, 53.88846838203498],
    [-117.34619390590258, 53.89079523121049],
    [-117.34619390590258, 53.892009188053336],
    [-117.35649358851977, 53.89079523121049],
    [-117.36387502772875, 53.89190802632987]]),
    {uniqueID:1}),
  ee.Feature(ee.Geometry.Polygon([
```

```
    [-117.33949911220141, 53.88087993044206],
    [-117.34052908046313, 53.8848260971799],
    [-117.34653722865649, 53.88705197563614],
    [-117.34945547206469, 53.88421901884885]]),
    {uniqueID:2}),
  ee.Feature(ee.Geometry.Polygon([
    [-117.35975515468188, 53.87571899651551],
    [-117.35460531337328, 53.872176810527606],
    [-117.3530603609807, 53.872784063718925],
    [-117.3527170382268, 53.87470697399036],
    [-117.34550726039477, 53.87703458918418],
    [-117.34636556727953, 53.87915968990993],
    [-117.35512029750414, 53.879362074822474]]),
    {uniqueID:3}),
  ee.Feature(ee.Geometry.Polygon([
    [-117.29915868861742, 53.89929218865425],
    [-117.28782903773852, 53.88958123910657],
    [-117.28594076259203, 53.89039057109378],
    [-117.28782903773852, 53.89251499299772],
    [-117.28594076259203, 53.89504392589631],
    [-117.2926355562932, 53.89969676258129],
    [-117.29641210658617, 53.900404757527085]]),
    {uniqueID:4}),
  ee.Feature(ee.Geometry.Polygon([
    [-117.27152120692797, 53.90252867038129],
    [-117.26860296351977, 53.90121387991749],
    [-117.26654302699633, 53.903034348003516],
    [-117.2648264132268, 53.906978423391124],
    [-117.27186452968188, 53.90475360613517]]),
    {uniqueID:5})
  ]);
Map.addLayer(harvAreas,{},'harvAreas polygons');

// ----------------------------------------
// ----- Import metric layer asset(s) -----

//=== Import layer from exported assets
var assetImg = ee.Image('users/myAccount/myFolder/myLayer'); //* Set to your own asset

//=== Name for exporting - metric layers
var exportDescp = 'Exported_thisMetricLayer_Stats'; //* Set to your own asset name
// //=== Or, for flag layers
// var exportDescp = 'Exported_thisFlagcLayer_FlagStats'; //* Set to your own asset name
// //=== Or, for pixel mask stats
// var exportDescp = 'Exported_PixelStats'; //* Set to your own asset name

//#############################################################################################
//##### EXTRACT PER-FEATURE METRIC & FLAG STATISTICS #####
//#############################################################################################

// ------------------------------------------------------------------
// ----- Extract zonal mean and std dev statistics (via reducers) -----

//=== Crs Transform copied from the imported asset's band info
print(assetImg);
var origProjCrsTrsfm = [30,0,405225,0,-30,6006435]; //* Customize to your own area of interest and layer(s)
var thisCRS = assetImg.projection().crs();

//=== Extract name of img, create custom output names
var imgName = ee.String(assetImg.bandNames().get(0));

//=== Combine reducers (mean & std dev)
var twoReducers = ee.Reducer.mean().combine({
  reducer2: ee.Reducer.stdDev(),
  sharedInputs: true
});

//=== Apply reducers to img
var haStats = assetImg.reduceRegions({
  collection: harvAreas,
  reducer: twoReducers,
  crs: assetImg.projection().crs(),
  crsTransform: origProjCrsTrsfm,
```

```
  tileScale: 4
});

//=== Fill null values with -999, and set property names
var exportStats = haStats.map(function(poly){
  var mean = ee.Algorithms.If(poly.get('mean'), poly.get('mean'), -999);
  var std = ee.Algorithms.If(poly.get('stdDev'), poly.get('stdDev'), -999);
  var outPoly = poly.set({
    METRIC_MEAN: mean,
    METRIC_STDEV: std,
    METRIC: imgName
  });

  return outPoly;
});

// -------------------------------------------
// ----- Export to Google Drive (as csv) -----

//=== Export to google drive
Export.table.toDrive({
  collection: exportStats,
  description: exportDescp,
  fileFormat: 'CSV',
  folder: 'myFolder', //* Set to a folder of your choice
  selectors: 'uniqueID,METRIC,METRIC_MEAN,METRIC_STDEV'
});

// //*** Alternative (e.g., for flags)::
// //*********************************

// // -----------------------------------------------------
// // ----- Extract zonal sum statistics (via reducers) -----

// //=== Crs Transform copied from the imported asset's band info
// print(assetImg);
// var origProjCrsTrsfm = [30,0,190785,0,-30,5843415]; //* Customize to your own area of interest and layer(s)
// var thisCRS = assetImg.projection().crs();

// //=== Extract name of img, create custom output names
// var imgName = ee.String(assetImg.bandNames().get(0));

// //=== Apply reducers to img
// var haStats = assetImg.reduceRegions({
//   collection: harvAreas,
//   reducer: ee.Reducer.sum(),
//   crs: thisCRS,
//   crsTransform: origProjCrsTrsfm,
//   tileScale: 4
// });

// //=== Fill null values with -999, and set property names
// var exportStats = haStats.map(function(poly){
//   var sum = ee.Algorithms.If(poly.get('sum'), poly.get('sum'), -999);
//   var outPoly = poly.set({
//     METRIC_SUM: sum,
//     FLAG: imgName
//   });

//   return outPoly;
// });

// // -------------------------------------------
// // ----- Export to Google Drive (as csv) -----

// //=== Export to google drive
// Export.table.toDrive({
//   collection: exportStats,
//   description: exportDescp,
//   fileFormat: 'CSV',
//   folder: 'myFolder', //* Set to a folder of your choice
//   selectors: 'uniqueID,FLAG,METRIC_SUM'
// });
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
//#############################################################################################
//##### EXTRACT ADDITIONAL PER-FEATURE STATISTICS ON RELEVANT PIXELS #####
//#############################################################################################

//=== Import mask layer
var fullMask = ee.Image('users/myAccount/myFolder/myLayer'); //* Set to your own asset
// var fullMask = ee.Image("users/jnhird/maskImage_1984to2018");

//=== Create blank image filled with 1s, and then create masked version
var constImg = ee.Image.constant(1).rename('allPix');
var constImgMskd = constImg.updateMask(fullMask).rename('unMskPix');

// ------------------------------------------------------------------------------
// ------- Extract number/percent of unmasked/relevant pixels per feature -------

//=== Use reducer to extract sum of all pixels per HA polygon
var sumAllPix = constImg.reduceRegions({
  collection: harvAreas,
  reducer: ee.Reducer.sum(),
  crs: thisCRS,
  crsTransform: origProjCrsTrsfm,
  tileScale: 4
});

//=== Use reducer to extract sum of unmasked (kept) pixels per HA polygon
var sumUnMskPix = constImgMskd.reduceRegions({
  collection: harvAreas,
  reducer: ee.Reducer.sum(),
  crs: thisCRS,
  crsTransform: origProjCrsTrsfm,
  tileScale: 4
});

// ------------------------------------------------------------------------------
// ------- Extract Number of contiguous (relevant) pixels per feature -------

//=== Clip layer identifying unmasked pixels (not flagged at all) to polygon feature collection
var clipConstImgMskd = constImgMskd.clip(harvAreas);

//=== Create binary clipped image of pixels used in calculations (where weights are all 1)
var usedPixels = ee.Image(1).updateMask(clipConstImgMskd.gt(0));

//=== Calculate per-pixel 'connected pixel count' tharvAreast includes only those pixels attached on sides (not diagonally)
var contigPixImg = usedPixels.connectedPixelCount(11,false);

//=== Use max() reducer to extract maximum "number of connected pixels" per each polygon
var maxContigPix = contigPixImg.reduceRegions({
  collection: harvAreas,
  reducer: ee.Reducer.max(),
  crs: thisCRS,
  crsTransform: origProjCrsTrsfm,
  tileScale: 4
});

// ----------------------------------------
// ------- Combine results for export -------

//=== Add stats to features
var pixStats = harvAreas.map(function(poly){
  //=== Parse uniqueID of polygon, filter stats by uniqueID
  var ID = ee.Number.parse(poly.get('uniqueID'));
  var allPixPoly = sumAllPix.filterMetadata('uniqueID','equals',ID).first();
  var unmskPixPoly = sumUnMskPix.filterMetadata('uniqueID','equals',ID).first();
  var maxCnnctdPixPoly = maxContigPix.filterMetadata('uniqueID','equals',ID).first();
  //=== Extract stats from filtered polygon
  var sumAllVal = ee.Number(allPixPoly.get('sum'));
  var sumMskdVal = ee.Number(unmskPixPoly.get('sum'));
  var maxCnnctdPix = ee.Number(maxCnnctdPixPoly.get('max'));
  //=== Add results as new properties to the polygon
  return poly.set({
    TOT_PIXELS: sumAllVal,
    UNMSKD_PIXELS: sumMskdVal,
```

Supplementary Material (Document S2) to: *Hird, Kariyeva, & McDermid. Satellite time series and Google Earth Engine democratize the process of forest-recovery monitoring over large areas. 2021.*

```
    PERC_UNMSKD: ee.Algorithms.If(sumAllVal.gt(0), sumMskdVal.divide(sumAllVal).multiply(100),0),
    MAX_CONNCTDPIX: maxCnnctdPix
  });
});

// -----------------------------------------
// ----- Export to Google Drive (as csv) -----

//=== Export to google drive
Export.table.toDrive({
  collection: pixStats,
  description: 'Summarized_PixelStats',
  fileFormat: 'CSV',
  folder: 'myFolder', //* Set to a folder of your choice
  selectors: 'uniqueID,TOT_PIXELS,UNMSKD_PIXELS,PERC_UNMSKD,MAX_CONNCTDPIX'
});
```

(b) Reformatting Google Earth Engine output generated by (a)

The following source code is written in R. It reformats some of the csv files exported by the above Google Earth Engine code given in (a)-ii. In particular, it reformats the "*_Stats.csv" files that contain per-polygon mean and standard deviation summaries for various spectral recovery metrics, and the "*_FlagStats.csv" files that contain per-polygon sums of various pixel flags (e.g., the "No Harvest" flag). The resulting, formatted files can be combined into a single dataset using the "uniqueID" field.

```
## ****************************************************************************************
## Script for (batch) reformatting spectral recovery output exported from GEE
## ****************************************************************************************

# LAST UPDATED: October 1, 2021

#---------------------------------------------------------------------------------------------
#--- Load required libraries
#---------------------------------------------------------------------------------------------

library(data.table) ## For reading in and working with very large tables
library(dplyr) ## For manipulating data frames

#---------------------------------------------------------------------------------------------
#--- Set working directory, get list of input files
#---------------------------------------------------------------------------------------------

#--- Set working directory
# inFolder <- "D:/MyFolder" # **Replace with your own folder path
setwd(inFolder)

#---------------------------------------------------------------------------------------------
#--- Recovery Metric Outputs: Import original files, loop through to reformat and write to new files
#---------------------------------------------------------------------------------------------

#--- Get list of names for original metric GEE output files
origMetFiles <- list.files(path=inFolder, pattern="*_Stats.csv", full.names=TRUE, recursive=FALSE)
origMetFiles

#--- For each file, do...
for (i in 1:length(origMetFiles)) {

  #--- Import data from current file
  thisDat <- fread(origMetFiles[i], data.table = FALSE)

  #--- Extract name of metric
  metName <- thisDat[1,"METRIC"]

  #--- Replace METRIC_MEAN and METRIC_STDEV with custom column names
  meanName <- paste0(metName,"_mean")
  colnames(thisDat)[grep("MEAN",colnames(thisDat))] <- meanName
  sdName <- paste0(metName,"_sd")
  colnames(thisDat)[grep("STDEV",colnames(thisDat))] <- sdName

  #--- Remove "METRIC" column from table
  outTable <- thisDat[,-grep("METRIC",colnames(thisDat))]

  #--- For each metric columns, replace -999 with NA
  metCols <- c(meanName, sdName)
  for (x in 1:length(metCols)) {
    outTable[which(outTable[,metCols[x]] == -999),metCols[x]] <- NA
  }

  #--- Create output file name
  fullName <- substr(origMetFiles[i], 1, nchar(origMetFiles[i])-4)
  fileName <- substr(fullName, nchar(inFolder)+2, nchar(fullName))
  extension <- substr(origMetFiles[i], nchar(origMetFiles[i])-3, nchar(origMetFiles[i]))
  outputName <- paste0(fileName,"_RFMTTD",extension)

  #--- Write to file
  fwrite(outTable, file = outputName)
```

```
  #--- Output message to console
  print(paste0("Done formatting file ",i," of ",length(origMetFiles),"."))

}

#-------------------------------------------------------------------------------------------
#--- Flag Outputs: Import original files, loop through to reformat and write to new files
#-------------------------------------------------------------------------------------------

#--- Get list of names for original metric GEE output files
origFlgFiles <- list.files(path=inFolder, pattern="*_FlagStats.csv", full.names=TRUE, recursive=FALSE)
origFlgFiles

#--- For each file, do...
for (i in 1:length(origFlgFiles)) {

  #--- Import data from current file
  thisDat <- fread(origFlgFiles[i], data.table = FALSE)

  #--- Extract name of metric
  flagName <- thisDat[1,"FLAG"]

  #--- Replace FLAG_SUM awith custom column names
  flagName <- paste0(flagName,"_sum")
  colnames(thisDat)[grep("SUM",colnames(thisDat))] <- flagName

  #--- Remove "FLAG" column from table
  outTable <- thisDat[,-grep("FLAG",colnames(thisDat))]

  #--- Replace -999 with NA
  outTable[which(outTable[,flagName] == -999),flagName] <- NA

  #--- Create output file name
  fullName <- substr(origFlgFiles[i], 1, nchar(origFlgFiles[i])-4)
  fileName <- substr(fullName, nchar(inFolder)+2, nchar(fullName))
  extension <- substr(origFlgFiles[i], nchar(origFlgFiles[i])-3, nchar(origFlgFiles[i]))
  outputName <- paste0(fileName,"_RFMTTD",extension)

  #--- Write to file
  fwrite(outTable, file = outputName)

  #--- Output message to console
  print(paste0("Done formatting file ",i," of ",length(origFlgFiles),"."))

}
```