To the best of my knowledge, this code is complete and ready to go.

**Snow Season Begin Difference between 2000 and 2022**

var startDoy = 1;

var startYear = 2000;

var endYear = 2022;

var startDate;

var startYear;

//This is the difference in snow season start date for the years 2000 and 2022

//The Red pixels indicate the date is earlier in 2022 than in 2000

//The blue pixels indicate the date is later in 2022 than in 2000

var studyArea = ee.FeatureCollection('projects/kaander08/assets/study\_area').geometry();

function addDateBands(img) {

// Get image date.

var date = img.date();

// Get calendar day-of-year.

var calDoy = date.getRelative('day', 'year');

// Get relative day-of-year; enumerate from user-defined startDoy.

var relDoy = date.difference(startDate, 'day');

// Get the date as milliseconds from Unix epoch.

var millis = date.millis();

// Add all of the above date info as bands to the snow fraction image.

var dateBands = ee.Image.constant([calDoy, relDoy, millis, startYear])

.rename(['calDoy', 'relDoy', 'millis', 'year']);

// Cast bands to correct data type before returning the image.

return img.addBands(dateBands)

.cast({'calDoy': 'int', 'relDoy': 'int', 'millis': 'long', 'year': 'int'})

.set('millis', millis);

}

var waterMask = ee.Image('MODIS/MOD44W/MOD44W\_005\_2000\_02\_24')

.select('water\_mask')

.not();

var completeCol = ee.ImageCollection('MODIS/006/MOD10A1')

.select('NDSI\_Snow\_Cover');

// Pixels must have been 10% snow covered for at least 2 weeks in 2018.

var snowCoverEphem = completeCol.filterDate('2018-01-01', '2019-01-01')

.map(function(img) {

return img.gte(10);

})

.sum()

.gte(14);

// Pixels must not be 10% snow covered more than 124 days in 2018.

var snowCoverConst = completeCol.filterDate('2018-01-01', '2019-01-01')

.map(function(img) {

return img.gte(10);

})

.sum()

.lte(124);

var analysisMask = waterMask.multiply(snowCoverEphem).multiply(snowCoverConst);

var years = ee.List.sequence(startYear, endYear);

var annualList = years.map(function(year) {

// Set the global startYear variable as the year being worked on so that

// it will be accessible to the addDateBands mapped to the collection below.

startYear = year;

// Get the first day-of-year for this year as an ee.Date object.

var firstDoy = ee.Date.fromYMD(year, 1, 1);

// Advance from the firstDoy to the user-defined startDay; subtract 1 since

// firstDoy is already 1. Set the result as the global startDate variable so

// that it is accessible to the addDateBands mapped to the collection below.

startDate = firstDoy.advance(startDoy-1, 'day');

// Get endDate for this year by advancing 1 year from startDate.

// Need to advance an extra day because end date of filterDate() function

// is exclusive.

var endDate = startDate.advance(1, 'year').advance(1, 'day');

// Filter the complete collection by the start and end dates just defined.

var yearCol = completeCol.filterDate(startDate, endDate);

// Construct an image where pixels represent the first day within the date

// range that the lowest snow fraction is observed.

var noSnowImg = yearCol

// Add date bands to all images in this particular collection.

.map(addDateBands)

// Sort the images by ascending time to identify the first day without

// snow. Alternatively, you can use .sort('millis', false) to

// reverse sort (find first day of snow in the fall).

.sort('millis')

// Make a mosaic composed of pixels from images that represent the

// observation with the minimum percent snow cover (defined by the

// NDSI\_Snow\_Cover band); include all associated bands for the selected

// image.

.reduce(ee.Reducer.min(5))

// Rename the bands - band names were altered by previous operation.

.rename(['snowCover', 'calDoy', 'relDoy', 'millis', 'year'])

// Apply the mask.

.updateMask(analysisMask)

// Set the year as a property for filtering by later.

.set('year', year);

// Mask by minimum snow fraction - only include pixels that reach 0

// percent cover. Return the resulting image.

return noSnowImg.updateMask(noSnowImg.select('snowCover').eq(0));

});

var annualCol = ee.ImageCollection.fromImages(annualList);

var annualColClipped = annualCol.map(function(img) {

return img.clip(studyArea);

})

// Define the years to difference.

var firstYear = 2000;

var secondYear = 2022;

// Calculate difference image.

var firstImg = annualColClipped.filter(ee.Filter.eq('year', firstYear))

.first().select('calDoy');

var secondImg = annualColClipped.filter(ee.Filter.eq('year', secondYear))

.first().select('calDoy');

var dif = secondImg.subtract(firstImg);

// Define the visualization arguments and color pallette.

var visArgs = {

min: -15,

max: 15,

palette: ['b2182b', 'ef8a62', 'fddbc7', 'f7f7f7', 'd1e5f0', '67a9cf', '2166ac']};

// Center the map to Noerthern Alaska

Map.setCenter(-149.677, 69.559, 5);

Map.addLayer(dif, visArgs, '2000-2022 first day no snow dif');

**Snow Season End Difference between 2000 and 2022**

var startDoy = 1;

var startYear = 2000;

var endYear = 2022;

var startDate;

var startYear;

var studyArea = ee.FeatureCollection('projects/kaander08/assets/study\_area').geometry();

function addDateBands(img) {

// Get image date.

var date = img.date();

// Get calendar day-of-year.

var calDoy = date.getRelative('day', 'year');

// Get relative day-of-year; enumerate from user-defined startDoy.

var relDoy = date.difference(startDate, 'day');

// Get the date as milliseconds from Unix epoch.

var millis = date.millis();

// Add all of the above date info as bands to the snow fraction image.

var dateBands = ee.Image.constant([calDoy, relDoy, millis, startYear])

.rename(['calDoy', 'relDoy', 'millis', 'year']);

// Cast bands to correct data type before returning the image.

return img.addBands(dateBands)

.cast({'calDoy': 'int', 'relDoy': 'int', 'millis': 'long', 'year': 'int'})

.set('millis', millis);

}

var waterMask = ee.Image('MODIS/MOD44W/MOD44W\_005\_2000\_02\_24')

.select('water\_mask')

.not();

var completeCol = ee.ImageCollection('MODIS/006/MOD10A1')

.select('NDSI\_Snow\_Cover');

// Pixels must have been 10% snow covered for at least 2 weeks in 2018.

var snowCoverEphem = completeCol.filterDate('2018-01-01', '2019-01-01')

.map(function(img) {

return img.gte(10);

})

.sum()

.gte(14);

// Pixels must not be 10% snow covered more than 124 days in 2018.

var snowCoverConst = completeCol.filterDate('2018-01-01', '2019-01-01')

.map(function(img) {

return img.gte(10);

})

.sum()

.lte(124);

var analysisMask = waterMask.multiply(snowCoverEphem).multiply(snowCoverConst);

var years = ee.List.sequence(startYear, endYear);

var annualList = years.map(function(year) {

// Set the global startYear variable as the year being worked on so that

// it will be accessible to the addDateBands mapped to the collection below.

startYear = year;

// Get the first day-of-year for this year as an ee.Date object.

var firstDoy = ee.Date.fromYMD(year, 1, 1);

// Advance from the firstDoy to the user-defined startDay; subtract 1 since

// firstDoy is already 1. Set the result as the global startDate variable so

// that it is accessible to the addDateBands mapped to the collection below.

startDate = firstDoy.advance(startDoy-1, 'day');

// Get endDate for this year by advancing 1 year from startDate.

// Need to advance an extra day because end date of filterDate() function

// is exclusive.

var endDate = startDate.advance(1, 'year').advance(1, 'day');

// Filter the complete collection by the start and end dates just defined.

var yearCol = completeCol.filterDate(startDate, endDate);

// Construct an image where pixels represent the first day within the date

// range that the lowest snow fraction is observed.

var noSnowImg = yearCol

// Add date bands to all images in this particular collection.

.map(addDateBands)

// Sort the images by ascending time to identify the first day without

// snow. Alternatively, you can use .sort('millis', false) to

// reverse sort (find first day of snow in the fall).

.sort('millis', false)

// Make a mosaic composed of pixels from images that represent the

// observation with the minimum percent snow cover (defined by the

// NDSI\_Snow\_Cover band); include all associated bands for the selected

// image.

.reduce(ee.Reducer.min(5))

// Rename the bands - band names were altered by previous operation.

.rename(['snowCover', 'calDoy', 'relDoy', 'millis', 'year'])

// Apply the mask.

.updateMask(analysisMask)

// Set the year as a property for filtering by later.

.set('year', year);

// Mask by minimum snow fraction - only include pixels that reach 0

// percent cover. Return the resulting image.

return noSnowImg.updateMask(noSnowImg.select('snowCover').eq(0));

});

var annualCol = ee.ImageCollection.fromImages(annualList);

var annualColClipped = annualCol.map(function(img) {

return img.clip(studyArea);

})

// Define the years to difference.

var firstYear = 2000;

var secondYear = 2022;

// Calculate difference image.

var firstImg = annualColClipped.filter(ee.Filter.eq('year', firstYear))

.first().select('calDoy');

var secondImg = annualColClipped.filter(ee.Filter.eq('year', secondYear))

.first().select('calDoy');

var dif = secondImg.subtract(firstImg);

// Define visualization arguments.

var visArgs = {

min: -15,

max: 15,

palette: ['b2182b', 'ef8a62', 'fddbc7', 'f7f7f7', 'd1e5f0', '67a9cf', '2166ac']};

//Center map to Northern Alaska

Map.setCenter(-149.677, 69.559, 5);

Map.addLayer(dif, visArgs, '2000-2022 first day no snow dif');