//REGIONAL SCALE

var geometry =

 /\* color: #d63000 \*/

 /\* shown: false \*/

 ee.Geometry.Polygon(

 [[[70, 25],

 [105, 25],

 [105, 45],

 [70, 45],

 [70, 25]]]),

 ASTERemis = ee.Image("NASA/ASTER\_GED/AG100\_003"),

 ERA5 = ee.ImageCollection("ECMWF/ERA5/MONTHLY"),

 SouthAsiaEast\_Scherler = ee.FeatureCollection("users/kb621/15\_rgi60\_SouthAsiaEast\_LS\_DC\_2013\_2017\_RATIO"),

 SouthAsiaWest\_Scherler = ee.FeatureCollection("users/kb621/14\_rgi60\_SouthAsiaWest\_LS\_DC\_2013\_2017\_RATIO"),

 CentralAsia\_Scherler = ee.FeatureCollection("users/kb621/13\_rgi60\_CentralAsia\_LS\_DC\_2013\_2017\_RATIO"),

 WaterVapour = ee.ImageCollection("NCEP\_RE/surface\_wv");

// 1. Filter Landsat image collection and composite into a single image

//Get Landsat 8 image collection

var l8 = ee.ImageCollection('LANDSAT/LC08/C01/T1');

var l8visParams = {bands: ['B4', 'B3', 'B2'], max: 30000};

//Filter Landsat 8 image collection by space and time

var l8spatialFiltered = l8.filterBounds(geometry);

var l8temporalFiltered = l8spatialFiltered

.filter(ee.Filter.calendarRange(2013,2016,'year'))

.filter(ee.Filter.calendarRange(5,10,'month'));

//Filter to find the images with less than 20% cloud cover

var lesscloudy = l8temporalFiltered

.filterMetadata('CLOUD\_COVER', 'less\_than', 20);

print(lesscloudy, 'less cloudy');

//Set visual parameters

var l8visParams\_mean = {bands: ['B4\_mean', 'B3\_mean', 'B2\_mean'], max: 30000};

//Temporally reduce the image collection by taking the mean of the

//middle 60% of pixels of the images with less than 20% cloud cover

var mean\_lesscloudy = lesscloudy

.reduce(ee.Reducer.intervalMean(20, 80));

Map.addLayer(mean\_lesscloudy, l8visParams\_mean, 'Mean of middle 60%');

// 2. Calculate the land surface temperature from the composite thermal band imagery;

// Convert the raw data (composite image) to radiance

var radiance\_mean = ((mean\_lesscloudy.multiply(0.0003342)).add(0.1));

// Convert the raw data to top-of-atmosphere reflectance

var one = radiance\_mean.divide(radiance\_mean);

var k1constant = one.multiply(774.8853);

var k2constant = one.multiply(1321.0789);

var toa\_mean = k2constant.divide(((k1constant.divide(radiance\_mean)).add(1)).log());

// EITHER

//Calculation of emissivity corrected land surface temperature

//(using ASTER emissivity product)

var ASTERemis13 = ASTERemis.select(['emissivity\_band13']);

var ASTERemis14 = ASTERemis.select(['emissivity\_band14']);

var ASTERemis\_mean = (ASTERemis13.add(ASTERemis14)).divide(2);

var logASTERemis = (ASTERemis\_mean.multiply(0.001)).log();

var logASTERfilled = logASTERemis.unmask((ee.Number(0.94)).log());

var LST\_mean = toa\_mean.divide(((toa\_mean.divide(14388)).multiply(10.8).multiply(logASTERfilled)).add(1));

//Map.addLayer(LST\_mean, {bands: ['B10\_mean'], min: 250, max: 320, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'LST of mean image');

var LST\_mean\_C = LST\_mean.subtract(273.15);

Map.addLayer(LST\_mean\_C, {bands: ['B10\_mean'], min: -10, max: 30, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'LST of mean image degrees C');

// //OR

// //Calculation of land surface temperature, correcting for the

// //atmosphere using water vapour

// var ASTERemis13 = ASTERemis.select(['emissivity\_band13']);

// var ASTERemis14 = ASTERemis.select(['emissivity\_band14']);

// var ASTERemis\_mean = (ASTERemis13.add(ASTERemis14)).divide(2);

// var ASTERemis\_mean\_final = ASTERemis\_mean.multiply(0.001);

// var ASTERfilled = ASTERemis\_mean\_final.unmask(ee.Number(0.94));

// var logASTERemis = (ASTERemis\_mean.multiply(0.001)).log();

// var logASTERfilled = logASTERemis.unmask((ee.Number(0.94)).log());

// var WaterVapour\_spatialFiltered =WaterVapour.filterBounds(geometry);

// var WaterVapour\_temporalFiltered = WaterVapour\_spatialFiltered

// .filter(ee.Filter.calendarRange(2015,2015,'year'))

// .filter(ee.Filter.calendarRange(5,10,'month'));

// var mean\_WaterVapour = WaterVapour\_temporalFiltered

// .reduce(ee.Reducer.mean());

// Map.addLayer(mean\_WaterVapour, {bands: ['pr\_wtr\_mean'], min: -10, max: 30, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'water vapour');

// var y = (toa\_mean.multiply(toa\_mean)).divide( radiance\_mean.multiply(1324));var delta =

// toa\_mean.subtract((toa\_mean.multiply(toa\_mean)).divide(1324));

// var w1 = (((mean\_WaterVapour.multiply(mean\_WaterVapour)).multiply(0.04019)).add(mean\_WaterVapour.multiply(0.02916))).add(1.101523);

// var w2 = (((mean\_WaterVapour.multiply(mean\_WaterVapour)).multiply(-0.38333)).add(mean\_WaterVapour.multiply(-1.50294))).add(0.20324);

// var w3 = (((mean\_WaterVapour.multiply(mean\_WaterVapour)).multiply(0.00918)).add(mean\_WaterVapour.multiply(1.36072))).add(-0.27514);

// var LST\_mean3 = (((((w1.multiply(radiance\_mean)).add(w2)).multiply(one.divide(ASTERfilled))).add(w3)).multiply(y)).add(delta);

// Map.addLayer(LST\_mean3, {bands: ['B10\_mean'], min: 250, max: 320, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'LST of mean image 3');

// var LST\_mean3\_C = LST\_mean3.subtract(273.15);

// Map.addLayer(LST\_mean3\_C, {bands: ['B10\_mean'], min: -10, max: 30, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'LST of mean image 3 degrees C');

// 3. Normalise the land surface temperature

//Filter ERA5 Reanalysis data

var ERA5temporalFiltered =

ERA5.filter(ee.Filter.calendarRange(2013,2016,'year'))

.filter(ee.Filter.calendarRange(5,10,'month'));

//Temporally reduce the image collection by taking the mean of the

//middle 60% of pixels

var mean\_ERA5filtered = ERA5temporalFiltered

.reduce(ee.Reducer.intervalMean(20, 80));

var ERA5visParams\_mean = {bands: ['mean\_2m\_air\_temperature\_mean'],

min:260,max: 320};

//Set visual parameters

var visParams = {bands: ['mean\_2m\_air\_temperature'], min:260,max: 310};

//Map.addLayer(mean\_ERA5filtered, ERA5visParams\_mean, 'Mean of ERA5');

//Find mean air temperature of HMA region

var meanDictionary = mean\_ERA5filtered.reduceRegion({

 reducer: ee.Reducer.mean(),

 geometry: geometry,

 scale: 30000,

 maxPixels: 1e9

});

print('Mean:', meanDictionary.get('mean\_2m\_air\_temperature\_mean'));

//Produce a constant array of the mean air temperature

var constant = ee.Number(meanDictionary.get( 'mean\_2m\_air\_temperature\_mean'));

var constant\_array = (mean\_ERA5filtered.divide(mean\_ERA5filtered)).multiply(constant);

//Divide constant array by ERA5 composite image

var multiplier = constant\_array.divide(mean\_ERA5filtered);

//Multiply surface temperature image by multiplier image to normalise

var LST\_mean\_norm\_C = LST\_mean\_C.multiply(multiplier.select("mean\_2m\_air\_temperature\_mean"));

Map.addLayer(LST\_mean\_norm\_C, {bands: ['B10\_mean'], min: -10, max: 30, palette: ['5EDAFF', 'FBFF5E', 'FF8989']}, 'LST of mean image normalised degrees C');

// 4. Apply regional rational curve debris thickness/surface temperature relationship to regional surface temp calculation

//Apply empirically derived relationship to the regionally calculated land surface temperature

var dt\_c = LST\_mean\_norm\_C.divide((LST\_mean\_norm\_C.multiply(-0.0198)).add(0.558));

var dt\_c\_clippedW = (dt\_c.select('B10\_mean')).clipToCollection(SouthAsiaWest\_Scherler);

var dt\_c\_clippedE = (dt\_c.select('B10\_mean')).clipToCollection(SouthAsiaEast\_Scherler);

var dt\_c\_clippedC = (dt\_c.select('B10\_mean')).clipToCollection(CentralAsia\_Scherler);

var dt\_c\_clipped = ee.ImageCollection([dt\_c\_clippedW, dt\_c\_clippedE, dt\_c\_clippedC]).mosaic();

Map.addLayer(dt\_c\_clipped, {bands: ['B10\_mean'], min: 0, max: 200, palette: ['FF0000', '00FF00', '0000FF']}, 'dt of LST\_mean normalised degrees C clipped');