Supplementary File S1: Goggle Earth Engine Script for extracting flood from Sentinel-1 images

```
//Filter Date_before flood
var start = ee.Date('');
var finish = ee.Date(''); //Change the date if needed

//Filter Date_after flood
var start_1 = ee.Date('');
var finish_1 = ee.Date(''); //Change the date if needed

// Filter by metadata properties.
var sentinel1 = ee.ImageCollection('COPERNICUS/S1_GRD')
.filterDate(start, finish)
.filterBounds(cws)
// Filter to get images with VV and VH dual polarization.
.filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VH'))
// .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))
// Filter to get images collected in interferometric wide swath mode.
.filter(ee.Filter.eq('instrumentMode', 'IW'))
// filter 10m resolution
.filter(ee.Filter.eq('resolution_meters', 10));

// one collection per polarisation
var vh = sentinel1.select("VH");
var vv = sentinel1.select("VV");

// Filter to get images from different look angles.
var vhAscending = vh.filter(ee.Filter.eq('orbitProperties_pass', 'ASCENDING'));
var vhDescending = vh.filter(ee.Filter.eq('orbitProperties_pass', 'DESCENDING'));

// Convert the collection to a list and get the number of images.
var sizeAscendingvh = vhAscending.toList(10000).length();
print('Number of VH Ascending images_before: ', sizeAscendingvh);
var sizeDescendingvh = vhDescending.toList(10000).length();
print('Number of VH Descending images_before: ', sizeDescendingvh);

// Create a median from VH polarization and different look angles.
var median_image_VH = vhDescending.median();
var before = median_image_VH.clip(cws);


Map.addLayer(median_image_VH.clip(cws), {min: [-30], max: [0]}, 'CWS_vhDesc_before', 1);
//Map.addLayer(median_image_vhAsc.clip(Kutch), {min: [-30], max: [0]}, 'Kutch_vhAsc_2011417', 1);
//Map.addLayer(median_image_VH_clip, {bands:['VH']}, {min: [-30], max: [0]}, 'test', 1);

// Export the image, specifying scale and region.
Export.image.toDrive({
```

```
image: before,
description: 'CWS_vhDesc_before',
scale: 10,
region: cws,
maxPixels: 15000000000
});

// Filter by metadata properties_after flood
var sentinel1 = ee.ImageCollection('COPERNICUS/S1_GRD')
.filterDate(start_1, finish_1)
.filterBounds(cws)
// Filter to get images with VV and VH dual polarization.
.filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VH'))
// .filter(ee.Filter.listContains('transmitterReceiverPolarisation', 'VV'))
// Filter to get images collected in interferometric wide swath mode.
.filter(ee.Filter.eq('instrumentMode', 'IW'))
// filter 10m resolution
.filter(ee.Filter.eq('resolution_meters', 10));

// one collection per polarisation
var vh = sentinel1.select("VH");
var vv = sentinel1.select("VV");

// Filter to get images from different look angles.
var vhAscending = vh.filter(ee.Filter.eq('orbitProperties_pass', 'ASCENDING'));
var vhDescending = vh.filter(ee.Filter.eq('orbitProperties_pass', 'DESCENDING'));

// Convert the collection to a list and get the number of images.
var sizeAscendingvh = vhAscending.toList(10000).length();
print('Number of VH Ascending images_after: ', sizeAscendingvh);
var sizeDescendingvh = vhDescending.toList(10000).length();
print('Number of VH Descending images_after: ', sizeDescendingvh);

// Create a median from VH polarization and different look angles.
var median_image_VH = vhDescending.median();
var after = median_image_VH.clip(cws);


Map.addLayer(median_image_VH.clip(cws), {min: [-30], max: [0]}, 'CWS_vhDesc_after', 1);
//Map.addLayer(median_image_vhAsc.clip(Kutch), {min: [-30], max: [0]},
'Kutch_vhAsc_2011417', 1);
//Map.addLayer(median_image_VH_clip, {bands:['VH']}, {min: [-30], max: [0]}, 'test', 1);

// Export the image, specifying scale and region.
Export.image.toDrive({
image: after,
description: 'CWS_vhDesc_after',
scale: 10,
region: cws,
maxPixels: 15000000000
```

```
});


var srtm = ee.Image('USGS/SRTMGL1_003');

before = before.updateMask(srtm.gt(0));
after = after.updateMask(srtm.gt(0));
//later = later.updateMask(srtm.gt(0));

Map.setCenter(79.73, 10.93, 7);

Map.addLayer(before, {min:-30,max:0}, 'before flooding');
Map.addLayer(after, {min:-30,max:0}, 'after flooding');
Map.addLayer(after.subtract(before), {min:-10,max:10}, 'After - before', 0);

// Threshold smoothed radar intensities to identify "flooded" areas
var SMOOTHING_RADIUS = 100;
var DIFF_UPPER_THRESHOLD = -3;
var NORMAL_WATER_THRESHOLD = 0.05; // Anything that occurs >5% of the time

var diff_smoothed = after.focal_median(SMOOTHING_RADIUS,'circle','meters')
.subtract(before.focal_median(SMOOTHING_RADIUS,'circle','meters'));
Map.addLayer(
diff_smoothed,
{min:-10,max:10}, 'diff smoothed', 0);
var diff_thresholded = diff_smoothed.lt(DIFF_UPPER_THRESHOLD);
Map.addLayer(
diff_thresholded.updateMask(diff_thresholded),
{min:0,max:1, palette:"blue"}, 'potentially flooded areas (w/0 expected water removed) - blue');
```