

var districtNames = ['Sukkur', 'Shikarpur', 'Larkana']; // Add more district names as needed

// Define the dropdown menu widget for district selection

var districtSelect = ui.Select({

items: districtNames,

placeholder: 'Select a district',

});

// Define the submit button

var submitButton = ui.Button({

label: 'Submit',

onClick: function() {

// Get the selected district from the dropdown menu

var selectedDistrict = districtSelect.getValue();

// Clear the map

Map.clear();

// Perform analysis based on the selected district

if (selectedDistrict === 'Sukkur') {

// Call function to display analysis for Sukkur district

displaySukkurAnalysis();

} else if (selectedDistrict === 'Shikarpur') {

// Call function to display analysis for Shikarpur district

displayShikarpurAnalysis();

} else if (selectedDistrict === 'Larkana') {

// Call function to display analysis for Larkana district

displayLarkanaAnalysis();

} else {

// Handle other district selections if needed

print('Analysis for selected district is not available.');

}

}

});

// Create a panel to hold the dropdown menu and submit button

var panel = ui.Panel({

widgets: [

districtSelect,

submitButton

],

layout: ui.Panel.Layout.Flow('horizontal'),

style: {position: 'top-center', padding: '10px'}

});

// Add the panel to the map

Map.add(panel);

// Function to display analysis for Sukkur district

function displaySukkurAnalysis() {

// Add your analysis code for Sukkur district here

print('Analysis for Sukkur district will be displayed.');

// Define date ranges for flood analysis

var before\_start= '2022-07-20';

var before\_end='2022-07-30';

var after\_start='2022-09-10';

var after\_end='2022-09-20';

// Load district data

var province\_collection = ee.FeatureCollection(district1);

// Filter for a specific district

var geometry = province\_collection.filterMetadata("DISTRICT", "equals", "SUKKUR");

Map.addLayer(geometry, { color: 'green' }, 'SUKKUR DISTRICT');

// Define parameters

var polarization = "VH";

var pass\_direction = "DESCENDING";

var difference\_threshold = 1.25;

// Define area of interest

var aoi = ee.FeatureCollection(geometry);

// Load Sentinel-1 image collection

var collection = ee.ImageCollection('COPERNICUS/S1\_GRD')

.filter(ee.Filter.eq('instrumentMode', 'IW'))

.filter(ee.Filter.listContains('transmitterReceiverPolarisation', polarization))

.filter(ee.Filter.eq('orbitProperties\_pass', pass\_direction))

.filter(ee.Filter.eq('resolution\_meters', 10))

.filterBounds(aoi)

.select(polarization);

// Filter collections by date

var before\_collection = collection.filterDate(before\_start, before\_end);

var after\_collection = collection.filterDate(after\_start, after\_end);

// Exit the function or handle accordingly if no images are available

if (before\_collection.size().getInfo() === 0 || after\_collection.size().getInfo() === 0) {

print('No images available for the specified date ranges.');

return;

}

// Function to print date range

function dates(imgcol) {

var range = imgcol.reduceColumns(ee.Reducer.minMax(), ["system:time\_start"]);

var printed = ee.String('from ')

.cat(ee.Date(range.get('min')).format('YYYY-MM-dd'))

.cat(' to ')

.cat(ee.Date(range.get('max')).format('YYYY-MM-dd'));

return printed;

}

// Print selected tiles and their date range

var before\_count = before\_collection.size();

print(ee.String('Tiles selected: Before Flood ').cat('(').cat(before\_count).cat(')'),

dates(before\_collection), before\_collection);

var after\_count = before\_collection.size();

print(ee.String('Tiles selected: After Flood ').cat('(').cat(after\_count).cat(')'),

dates(after\_collection), after\_collection);

// Process images

var before = before\_collection.mosaic().clip(aoi);

var after = after\_collection.mosaic().clip(aoi);

var smoothing\_radius = 50;

var before\_filtered = before.focal\_mean(smoothing\_radius, 'circle', 'meters');

var after\_filtered = after.focal\_mean(smoothing\_radius, 'circle', 'meters');

var difference = after\_filtered.divide(before\_filtered);

var threshold = difference\_threshold;

var difference\_binary = difference.gt(threshold);

// Load surface water data

var swater = ee.Image('JRC/GSW1\_0/GlobalSurfaceWater').select('seasonality');

var swater\_mask = swater.gte(10).updateMask(swater.gte(10));

// Mask flooded areas based on surface water data

var flooded\_mask = difference\_binary.where(swater\_mask, 0);

var flooded = flooded\_mask.updateMask(flooded\_mask);

var connections = flooded.connectedPixelCount();

var flooded = flooded.updateMask(connections.gte(8));

var DEM = ee.Image('WWF/HydroSHEDS/03VFDEM');

var terrain = ee.Algorithms.Terrain(DEM);

var slope = terrain.select('slope');

var flooded = flooded.updateMask(slope.lt(5));

// Calculate flood pixel area

var flood\_pixelarea = flooded.select(polarization)

.multiply(ee.Image.pixelArea());

// Reduce flood pixel area to get statistics

var flood\_stats = flood\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 10,

bestEffort: true

});

var flood\_area\_ha = flood\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Load population data

var population\_count = ee.Image('JRC/GHSL/P2016/POP\_GPW\_GLOBE\_V1/2015').clip(aoi);

// Reproject flooded area to match population data projection

var GHSLprojection = population\_count.projection();

var flooded\_res1 = flooded

.reproject({

crs: GHSLprojection

});

// Mask population data with flooded area

var population\_exposed = population\_count

.updateMask(flooded\_res1)

.updateMask(population\_count);

// Calculate statistics for exposed population

var stats = population\_exposed.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 250,

maxPixels:1e9

});

var number\_pp\_exposed = stats.getNumber('population\_count').round();

// Load MODIS land cover data

var LC = ee.ImageCollection('MODIS/006/MCD12Q1')

.filterDate('2014-01-01', after\_end)

.sort('system:index', false)

.select("LC\_Type1")

.first()

.clip(aoi);

// Define cropland mask

var cropmask = LC.eq(12).or(LC.eq(14))

var cropland = LC.updateMask(cropmask)

// Reproject flooded area to match MODIS land cover projection

var MODISprojection = LC.projection();

var flooded\_res = flooded

.reproject({

crs: MODISprojection

});

// Mask cropland with flooded area

var cropland\_affected = flooded\_res

.updateMask(cropland)

// Calculate cropland pixel area

var crop\_pixelarea = cropland\_affected

.multiply(ee.Image.pixelArea());

var crop\_stats = crop\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

maxPixels: 1e9

});

var crop\_area\_ha = crop\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Define urban mask

var urbanmask = LC.eq(13)

var urban = LC.updateMask(urbanmask)

// Mask urban area with flooded area

var urban\_affected = urban

.mask(flooded\_res)

.updateMask(urban);

// Calculate urban pixel area

var urban\_pixelarea = urban\_affected

.multiply(ee.Image.pixelArea());

var urban\_stats = urban\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

bestEffort: true,

});

var urban\_area\_ha = urban\_stats

.getNumber('LC\_Type1')

.divide(10000)

.round();

// Add layers to map

Map.centerObject(aoi, 8);

Map.addLayer(before\_filtered, {min:-25,max:0}, 'Before Flood', 0);

Map.addLayer(after\_filtered, {min:-25,max:0}, 'After Flood', 1);

Map.addLayer(difference, {min:0,max:2}, "Difference Layer", 0);

Map.addLayer(flooded, {palette:"0000FF"}, 'Flooded areas');

var populationCountVis = {

min: 0,

max: 200.0,

palette: ['060606','337663','337663','ffffff'],

};

Map.addLayer(population\_count, populationCountVis, 'Population Density', 0);

var populationExposedVis = {

min: 0,

max: 200.0,

palette: ['yellow', 'orange', 'red'],

};

Map.addLayer(population\_exposed, populationExposedVis, 'Exposed Population');

var LCVis = {

min: 1.0,

max: 17.0,

palette: [

'05450a', '086a10', '54a708', '78d203', '009900', 'c6b044', 'dcd159',

'dade48', 'fbff13', 'b6ff05', '27ff87', 'c24f44', 'a5a5a5', 'ff6d4c',

'69fff8', 'f9ffa4', '1c0dff'

],

};

Map.addLayer(LC, LCVis, 'Land Cover', 0);

var croplandVis = {

min: 0,

max: 14.0,

palette: ['30b21c'],

};

Map.addLayer(cropland, croplandVis, 'Cropland', 0)

Map.addLayer(cropland\_affected, croplandVis, 'Affected Cropland');

var urbanVis = {

min: 0,

max: 13.0,

palette: ['grey'],

};

Map.addLayer(urban, urbanVis, 'Urban', 0)

Map.addLayer(urban\_affected, urbanVis, 'Affected Urban');

// Export images and vectors

Export.image.toDrive({

image: flooded,

description: 'Flood\_extent\_raster',

fileNamePrefix: 'flooded',

region: aoi,

maxPixels: 1e10

});

var flooded\_vec = flooded.reduceToVectors({

scale: 10,

geometryType:'polygon',

geometry: aoi,

eightConnected: false,

bestEffort:true,

tileScale:2,

});

Export.table.toDrive({

collection: flooded\_vec,

description: 'Flood\_extent\_vector',

fileFormat: 'SHP',

fileNamePrefix: 'flooded\_vec'

});

Export.image.toDrive({

image: population\_exposed,

description: 'Exposed\_Populuation',

scale: 250,

fileNamePrefix: 'population\_exposed',

region: aoi,

maxPixels: 1e10

});

// Create results panel

var results = ui.Panel({

style: {

position: 'bottom-left',

padding: '8px 15px',

width: '350px'

}

});

var textVis = {

'margin':'0px 8px 2px 0px',

'fontWeight':'bold'

};

var numberVIS = {

'margin':'0px 0px 15px 0px',

'color':'bf0f19',

'fontWeight':'bold'

};

var subTextVis = {

'margin':'0px 0px 2px 0px',

'fontSize':'12px',

'color':'grey'

};

var titleTextVis = {

'margin':'0px 0px 15px 0px',

'fontSize': '18px',

'font-weight':'',

'color': '3333ff'

};

var title = ui.Label('Results', titleTextVis);

var text1 = ui.Label('Flood status between:', textVis);

var number1 = ui.Label(after\_start.concat(" and ", after\_end), numberVIS);

var text2 = ui.Label('Estimated flood extent:', textVis);

var text2\_2 = ui.Label('Please wait...', subTextVis);

dates(after\_collection).evaluate(function(val){text2\_2.setValue('based on Sentinel-1 imagery '+val)});

var number2 = ui.Label('Please wait...', numberVIS);

flood\_area\_ha.evaluate(function(val){number2.setValue(val+' hectares')}), numberVIS;

var text3 = ui.Label('Estimated number of exposed people: ', textVis);

var text3\_2 = ui.Label('based on GHSL 2015 (250m)', subTextVis);

var number3 = ui.Label('Please wait...', numberVIS);

number\_pp\_exposed.evaluate(function(val){number3.setValue(val)}), numberVIS;

var MODIS\_date = ee.String(LC.get('system:index')).slice(0,4);

var text4 = ui.Label('Estimated affected cropland:', textVis);

var text4\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text4\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number4 = ui.Label('Please wait...', numberVIS);

crop\_area\_ha.evaluate(function(val){number4.setValue(val+' hectares')}), numberVIS;

var text5 = ui.Label('Estimated affected urban areas:', textVis);

var text5\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text5\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number5 = ui.Label('Please wait...', numberVIS);

urban\_area\_ha.evaluate(function(val){number5.setValue(val+' hectares')}), numberVIS;

results.add(ui.Panel([

title,

text1,

number1,

text2,

text2\_2,

number2,

text3,

text3\_2,

number3,

text4,

text4\_2,

number4,

text5,

text5\_2,

number5,

]));

Map.add(results);

// Create legend

var legend = ui.Panel({

style: {

position: 'bottom-right',

padding: '8px 15px',

}

});

var legendTitle = ui.Label('Legend', titleTextVis);

legend.add(legendTitle);

var makeRow = function(color, name) {

var colorBox = ui.Label({

style: {

backgroundColor: color,

padding: '8px',

margin: '0 0 4px 0'

}

});

var description = ui.Label({

value: name,

style: {margin: '0 0 4px 6px'}

});

return ui.Panel({

widgets: [colorBox, description],

layout: ui.Panel.Layout.Flow('horizontal')

});

};

var palette =['#0000FF', '#30b21c', 'grey'];

var names = ['potentially flooded areas','affected cropland','affected urban'];

for (var i = 0; i < 3; i++) {

legend.add(makeRow(palette[i], names[i]));

}

var legendTitle2 = ui.Label({

value: 'Exposed population density',

style: {

fontWeight: 'bold',

fontSize: '15px',

margin: '10px 0 0 0',

padding: '0'

}

});

legend.add(legendTitle2);

var lon = ee.Image.pixelLonLat().select('latitude');

var gradient = lon.multiply((populationExposedVis.max-populationExposedVis.min)/100.0).add(populationExposedVis.min);

var legendImage = gradient.visualize(populationExposedVis);

var panel = ui.Panel({

widgets: [

ui.Label('> '.concat(populationExposedVis['max']))

],

});

legend.add(panel);

var thumbnail = ui.Thumbnail({

image: legendImage,

params: {bbox:'0,0,10,100', dimensions:'10x50'},

style: {padding: '1px', position: 'bottom-center'}

});

legend.add(thumbnail);

var panel = ui.Panel({

widgets: [

ui.Label(populationExposedVis['min'])

],

});

legend.add(panel);

Map.add(legend);

}

// Function to display analysis for SHIKARPUR district

function displayShikarpurAnalysis() {

// Add your analysis code for SHIKARPUR district here

print('Analysis for SHIKARPUR district will be displayed.');

// Define date ranges

var before\_start= '2022-07-20';

var before\_end='2022-07-30';

var after\_start='2022-09-10';

var after\_end='2022-09-20';

// Load district data

var province\_collection = ee.FeatureCollection(district1);

// Filter for a specific district

var geometry = province\_collection.filterMetadata("DISTRICT", "equals", "SHIKARPUR");

Map.addLayer(geometry, { color: 'green' }, 'SHIKARPUR DISTRICT');

// Define parameters

var polarization = "VH";

var pass\_direction = "DESCENDING";

var difference\_threshold = 1.25;

// Define area of interest

var aoi = ee.FeatureCollection(geometry);

// Load Sentinel-1 image collection

var collection = ee.ImageCollection('COPERNICUS/S1\_GRD')

.filter(ee.Filter.eq('instrumentMode', 'IW'))

.filter(ee.Filter.listContains('transmitterReceiverPolarisation', polarization))

.filter(ee.Filter.eq('orbitProperties\_pass', pass\_direction))

.filter(ee.Filter.eq('resolution\_meters', 10))

.filterBounds(aoi)

.select(polarization);

// Filter collections by date

var before\_collection = collection.filterDate(before\_start, before\_end);

var after\_collection = collection.filterDate(after\_start, after\_end);

// Exit the function or handle accordingly if no images are available

if (before\_collection.size().getInfo() === 0 || after\_collection.size().getInfo() === 0) {

print('No images available for the specified date ranges.');

return;

}

// Function to print date range

function dates(imgcol) {

var range = imgcol.reduceColumns(ee.Reducer.minMax(), ["system:time\_start"]);

var printed = ee.String('from ')

.cat(ee.Date(range.get('min')).format('YYYY-MM-dd'))

.cat(' to ')

.cat(ee.Date(range.get('max')).format('YYYY-MM-dd'));

return printed;

}

// Print selected tiles and their date range

var before\_count = before\_collection.size();

print(ee.String('Tiles selected: Before Flood ').cat('(').cat(before\_count).cat(')'),

dates(before\_collection), before\_collection);

var after\_count = before\_collection.size();

print(ee.String('Tiles selected: After Flood ').cat('(').cat(after\_count).cat(')'),

dates(after\_collection), after\_collection);

// Process images

var before = before\_collection.mosaic().clip(aoi);

var after = after\_collection.mosaic().clip(aoi);

var smoothing\_radius = 50;

var before\_filtered = before.focal\_mean(smoothing\_radius, 'circle', 'meters');

var after\_filtered = after.focal\_mean(smoothing\_radius, 'circle', 'meters');

var difference = after\_filtered.divide(before\_filtered);

var threshold = difference\_threshold;

var difference\_binary = difference.gt(threshold);

// Load surface water data

var swater = ee.Image('JRC/GSW1\_0/GlobalSurfaceWater').select('seasonality');

var swater\_mask = swater.gte(10).updateMask(swater.gte(10));

// Mask flooded areas based on surface water data

var flooded\_mask = difference\_binary.where(swater\_mask, 0);

var flooded = flooded\_mask.updateMask(flooded\_mask);

var connections = flooded.connectedPixelCount();

var flooded = flooded.updateMask(connections.gte(8));

var DEM = ee.Image('WWF/HydroSHEDS/03VFDEM');

var terrain = ee.Algorithms.Terrain(DEM);

var slope = terrain.select('slope');

var flooded = flooded.updateMask(slope.lt(5));

// Calculate flood pixel area

var flood\_pixelarea = flooded.select(polarization)

.multiply(ee.Image.pixelArea());

// Reduce flood pixel area to get statistics

var flood\_stats = flood\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 10,

bestEffort: true

});

var flood\_area\_ha = flood\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Load population data

var population\_count = ee.Image('JRC/GHSL/P2016/POP\_GPW\_GLOBE\_V1/2015').clip(aoi);

// Reproject flooded area to match population data projection

var GHSLprojection = population\_count.projection();

var flooded\_res1 = flooded

.reproject({

crs: GHSLprojection

});

// Mask population data with flooded area

var population\_exposed = population\_count

.updateMask(flooded\_res1)

.updateMask(population\_count);

// Calculate statistics for exposed population

var stats = population\_exposed.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 250,

maxPixels:1e9

});

var number\_pp\_exposed = stats.getNumber('population\_count').round();

// Load MODIS land cover data

var LC = ee.ImageCollection('MODIS/006/MCD12Q1')

.filterDate('2014-01-01', after\_end)

.sort('system:index', false)

.select("LC\_Type1")

.first()

.clip(aoi);

// Define cropland mask

var cropmask = LC.eq(12).or(LC.eq(14))

var cropland = LC.updateMask(cropmask)

// Reproject flooded area to match MODIS land cover projection

var MODISprojection = LC.projection();

var flooded\_res = flooded

.reproject({

crs: MODISprojection

});

// Mask cropland with flooded area

var cropland\_affected = flooded\_res

.updateMask(cropland)

// Calculate cropland pixel area

var crop\_pixelarea = cropland\_affected

.multiply(ee.Image.pixelArea());

var crop\_stats = crop\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

maxPixels: 1e9

});

var crop\_area\_ha = crop\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Define urban mask

var urbanmask = LC.eq(13)

var urban = LC.updateMask(urbanmask)

// Mask urban area with flooded area

var urban\_affected = urban

.mask(flooded\_res)

.updateMask(urban);

// Calculate urban pixel area

var urban\_pixelarea = urban\_affected

.multiply(ee.Image.pixelArea());

var urban\_stats = urban\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

bestEffort: true,

});

var urban\_area\_ha = urban\_stats

.getNumber('LC\_Type1')

.divide(10000)

.round();

// Add layers to map

Map.centerObject(aoi, 8);

Map.addLayer(before\_filtered, {min:-25,max:0}, 'Before Flood', 0);

Map.addLayer(after\_filtered, {min:-25,max:0}, 'After Flood', 1);

Map.addLayer(difference, {min:0,max:2}, "Difference Layer", 0);

Map.addLayer(flooded, {palette:"0000FF"}, 'Flooded areas');

var populationCountVis = {

min: 0,

max: 200.0,

palette: ['060606','337663','337663','ffffff'],

};

Map.addLayer(population\_count, populationCountVis, 'Population Density', 0);

var populationExposedVis = {

min: 0,

max: 200.0,

palette: ['yellow', 'orange', 'red'],

};

Map.addLayer(population\_exposed, populationExposedVis, 'Exposed Population');

var LCVis = {

min: 1.0,

max: 17.0,

palette: [

'05450a', '086a10', '54a708', '78d203', '009900', 'c6b044', 'dcd159',

'dade48', 'fbff13', 'b6ff05', '27ff87', 'c24f44', 'a5a5a5', 'ff6d4c',

'69fff8', 'f9ffa4', '1c0dff'

],

};

Map.addLayer(LC, LCVis, 'Land Cover', 0);

var croplandVis = {

min: 0,

max: 14.0,

palette: ['30b21c'],

};

Map.addLayer(cropland, croplandVis, 'Cropland', 0)

Map.addLayer(cropland\_affected, croplandVis, 'Affected Cropland');

var urbanVis = {

min: 0,

max: 13.0,

palette: ['grey'],

};

Map.addLayer(urban, urbanVis, 'Urban', 0)

Map.addLayer(urban\_affected, urbanVis, 'Affected Urban');

// Export images and vectors

Export.image.toDrive({

image: flooded,

description: 'Flood\_extent\_raster',

fileNamePrefix: 'flooded',

region: aoi,

maxPixels: 1e10

});

var flooded\_vec = flooded.reduceToVectors({

scale: 10,

geometryType:'polygon',

geometry: aoi,

eightConnected: false,

bestEffort:true,

tileScale:2,

});

Export.table.toDrive({

collection: flooded\_vec,

description: 'Flood\_extent\_vector',

fileFormat: 'SHP',

fileNamePrefix: 'flooded\_vec'

});

Export.image.toDrive({

image: population\_exposed,

description: 'Exposed\_Populuation',

scale: 250,

fileNamePrefix: 'population\_exposed',

region: aoi,

maxPixels: 1e10

});

// Create results panel

var results = ui.Panel({

style: {

position: 'bottom-left',

padding: '8px 15px',

width: '350px'

}

});

var textVis = {

'margin':'0px 8px 2px 0px',

'fontWeight':'bold'

};

var numberVIS = {

'margin':'0px 0px 15px 0px',

'color':'bf0f19',

'fontWeight':'bold'

};

var subTextVis = {

'margin':'0px 0px 2px 0px',

'fontSize':'12px',

'color':'grey'

};

var titleTextVis = {

'margin':'0px 0px 15px 0px',

'fontSize': '18px',

'font-weight':'',

'color': '3333ff'

};

var title = ui.Label('Results', titleTextVis);

var text1 = ui.Label('Flood status between:', textVis);

var number1 = ui.Label(after\_start.concat(" and ", after\_end), numberVIS);

var text2 = ui.Label('Estimated flood extent:', textVis);

var text2\_2 = ui.Label('Please wait...', subTextVis);

dates(after\_collection).evaluate(function(val){text2\_2.setValue('based on Sentinel-1 imagery '+val)});

var number2 = ui.Label('Please wait...', numberVIS);

flood\_area\_ha.evaluate(function(val){number2.setValue(val+' hectares')}), numberVIS;

var text3 = ui.Label('Estimated number of exposed people: ', textVis);

var text3\_2 = ui.Label('based on GHSL 2015 (250m)', subTextVis);

var number3 = ui.Label('Please wait...', numberVIS);

number\_pp\_exposed.evaluate(function(val){number3.setValue(val)}), numberVIS;

var MODIS\_date = ee.String(LC.get('system:index')).slice(0,4);

var text4 = ui.Label('Estimated affected cropland:', textVis);

var text4\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text4\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number4 = ui.Label('Please wait...', numberVIS);

crop\_area\_ha.evaluate(function(val){number4.setValue(val+' hectares')}), numberVIS;

var text5 = ui.Label('Estimated affected urban areas:', textVis);

var text5\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text5\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number5 = ui.Label('Please wait...', numberVIS);

urban\_area\_ha.evaluate(function(val){number5.setValue(val+' hectares')}), numberVIS;

results.add(ui.Panel([

title,

text1,

number1,

text2,

text2\_2,

number2,

text3,

text3\_2,

number3,

text4,

text4\_2,

number4,

text5,

text5\_2,

number5,

]));

Map.add(results);

// Create legend

var legend = ui.Panel({

style: {

position: 'bottom-right',

padding: '8px 15px',

}

});

var legendTitle = ui.Label('Legend', titleTextVis);

legend.add(legendTitle);

var makeRow = function(color, name) {

var colorBox = ui.Label({

style: {

backgroundColor: color,

padding: '8px',

margin: '0 0 4px 0'

}

});

var description = ui.Label({

value: name,

style: {margin: '0 0 4px 6px'}

});

return ui.Panel({

widgets: [colorBox, description],

layout: ui.Panel.Layout.Flow('horizontal')

});

};

var palette =['#0000FF', '#30b21c', 'grey'];

var names = ['potentially flooded areas','affected cropland','affected urban'];

for (var i = 0; i < 3; i++) {

legend.add(makeRow(palette[i], names[i]));

}

var legendTitle2 = ui.Label({

value: 'Exposed population density',

style: {

fontWeight: 'bold',

fontSize: '15px',

margin: '10px 0 0 0',

padding: '0'

}

});

legend.add(legendTitle2);

var lon = ee.Image.pixelLonLat().select('latitude');

var gradient = lon.multiply((populationExposedVis.max-populationExposedVis.min)/100.0).add(populationExposedVis.min);

var legendImage = gradient.visualize(populationExposedVis);

var panel = ui.Panel({

widgets: [

ui.Label('> '.concat(populationExposedVis['max']))

],

});

legend.add(panel);

var thumbnail = ui.Thumbnail({

image: legendImage,

params: {bbox:'0,0,10,100', dimensions:'10x50'},

style: {padding: '1px', position: 'bottom-center'}

});

legend.add(thumbnail);

var panel = ui.Panel({

widgets: [

ui.Label(populationExposedVis['min'])

],

});

legend.add(panel);

Map.add(legend);

}

// Function to display analysis for Larkana district

function displayLarkanaAnalysis() {

// Add your analysis code for Larkana district here

print('Analysis for Larkana district will be displayed.');

// Define date ranges

var before\_start= '2022-07-20';

var before\_end='2022-07-30';

var after\_start='2022-09-10';

var after\_end='2022-09-20';

// Load district data

var province\_collection = ee.FeatureCollection(district1);

// Filter for a specific district

var geometry = province\_collection.filterMetadata("DISTRICT", "equals", "LARKANA");

Map.addLayer(geometry, { color: 'green' }, 'LARKANA DISTRICT');

// Define parameters

var polarization = "VH";

var pass\_direction = "DESCENDING";

var difference\_threshold = 1.25;

// Define area of interest

var aoi = ee.FeatureCollection(geometry);

// Load Sentinel-1 image collection

var collection = ee.ImageCollection('COPERNICUS/S1\_GRD')

.filter(ee.Filter.eq('instrumentMode', 'IW'))

.filter(ee.Filter.listContains('transmitterReceiverPolarisation', polarization))

.filter(ee.Filter.eq('orbitProperties\_pass', pass\_direction))

.filter(ee.Filter.eq('resolution\_meters', 10))

.filterBounds(aoi)

.select(polarization);

// Filter collections by date

var before\_collection = collection.filterDate(before\_start, before\_end);

var after\_collection = collection.filterDate(after\_start, after\_end);

// Exit the function or handle accordingly if no images are available

if (before\_collection.size().getInfo() === 0 || after\_collection.size().getInfo() === 0) {

print('No images available for the specified date ranges.');

return;

}

// Function to print date range

function dates(imgcol) {

var range = imgcol.reduceColumns(ee.Reducer.minMax(), ["system:time\_start"]);

var printed = ee.String('from ')

.cat(ee.Date(range.get('min')).format('YYYY-MM-dd'))

.cat(' to ')

.cat(ee.Date(range.get('max')).format('YYYY-MM-dd'));

return printed;

}

// Print selected tiles and their date range

var before\_count = before\_collection.size();

print(ee.String('Tiles selected: Before Flood ').cat('(').cat(before\_count).cat(')'),

dates(before\_collection), before\_collection);

var after\_count = before\_collection.size();

print(ee.String('Tiles selected: After Flood ').cat('(').cat(after\_count).cat(')'),

dates(after\_collection), after\_collection);

// Process images

var before = before\_collection.mosaic().clip(aoi);

var after = after\_collection.mosaic().clip(aoi);

var smoothing\_radius = 50;

var before\_filtered = before.focal\_mean(smoothing\_radius, 'circle', 'meters');

var after\_filtered = after.focal\_mean(smoothing\_radius, 'circle', 'meters');

var difference = after\_filtered.divide(before\_filtered);

var threshold = difference\_threshold;

var difference\_binary = difference.gt(threshold);

// Load surface water data

var swater = ee.Image('JRC/GSW1\_0/GlobalSurfaceWater').select('seasonality');

var swater\_mask = swater.gte(10).updateMask(swater.gte(10));

// Mask flooded areas based on surface water data

var flooded\_mask = difference\_binary.where(swater\_mask, 0);

var flooded = flooded\_mask.updateMask(flooded\_mask);

var connections = flooded.connectedPixelCount();

var flooded = flooded.updateMask(connections.gte(8));

var DEM = ee.Image('WWF/HydroSHEDS/03VFDEM');

var terrain = ee.Algorithms.Terrain(DEM);

var slope = terrain.select('slope');

var flooded = flooded.updateMask(slope.lt(5));

// Calculate flood pixel area

var flood\_pixelarea = flooded.select(polarization)

.multiply(ee.Image.pixelArea());

// Reduce flood pixel area to get statistics

var flood\_stats = flood\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 10,

bestEffort: true

});

var flood\_area\_ha = flood\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Load population data

var population\_count = ee.Image('JRC/GHSL/P2016/POP\_GPW\_GLOBE\_V1/2015').clip(aoi);

// Reproject flooded area to match population data projection

var GHSLprojection = population\_count.projection();

var flooded\_res1 = flooded

.reproject({

crs: GHSLprojection

});

// Mask population data with flooded area

var population\_exposed = population\_count

.updateMask(flooded\_res1)

.updateMask(population\_count);

// Calculate statistics for exposed population

var stats = population\_exposed.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 250,

maxPixels:1e9

});

var number\_pp\_exposed = stats.getNumber('population\_count').round();

// Load MODIS land cover data

var LC = ee.ImageCollection('MODIS/006/MCD12Q1')

.filterDate('2014-01-01', after\_end)

.sort('system:index', false)

.select("LC\_Type1")

.first()

.clip(aoi);

// Define cropland mask

var cropmask = LC.eq(12).or(LC.eq(14))

var cropland = LC.updateMask(cropmask)

// Reproject flooded area to match MODIS land cover projection

var MODISprojection = LC.projection();

var flooded\_res = flooded

.reproject({

crs: MODISprojection

});

// Mask cropland with flooded area

var cropland\_affected = flooded\_res

.updateMask(cropland)

// Calculate cropland pixel area

var crop\_pixelarea = cropland\_affected

.multiply(ee.Image.pixelArea());

var crop\_stats = crop\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

maxPixels: 1e9

});

var crop\_area\_ha = crop\_stats

.getNumber(polarization)

.divide(10000)

.round();

// Define urban mask

var urbanmask = LC.eq(13)

var urban = LC.updateMask(urbanmask)

// Mask urban area with flooded area

var urban\_affected = urban

.mask(flooded\_res)

.updateMask(urban);

// Calculate urban pixel area

var urban\_pixelarea = urban\_affected

.multiply(ee.Image.pixelArea());

var urban\_stats = urban\_pixelarea.reduceRegion({

reducer: ee.Reducer.sum(),

geometry: aoi,

scale: 500,

bestEffort: true,

});

var urban\_area\_ha = urban\_stats

.getNumber('LC\_Type1')

.divide(10000)

.round();

// Add layers to map

Map.centerObject(aoi, 8);

Map.addLayer(before\_filtered, {min:-25,max:0}, 'Before Flood', 0);

Map.addLayer(after\_filtered, {min:-25,max:0}, 'After Flood', 1);

Map.addLayer(difference, {min:0,max:2}, "Difference Layer", 0);

Map.addLayer(flooded, {palette:"0000FF"}, 'Flooded areas');

var populationCountVis = {

min: 0,

max: 200.0,

palette: ['060606','337663','337663','ffffff'],

};

Map.addLayer(population\_count, populationCountVis, 'Population Density', 0);

var populationExposedVis = {

min: 0,

max: 200.0,

palette: ['yellow', 'orange', 'red'],

};

Map.addLayer(population\_exposed, populationExposedVis, 'Exposed Population');

var LCVis = {

min: 1.0,

max: 17.0,

palette: [

'05450a', '086a10', '54a708', '78d203', '009900', 'c6b044', 'dcd159',

'dade48', 'fbff13', 'b6ff05', '27ff87', 'c24f44', 'a5a5a5', 'ff6d4c',

'69fff8', 'f9ffa4', '1c0dff'

],

};

Map.addLayer(LC, LCVis, 'Land Cover', 0);

var croplandVis = {

min: 0,

max: 14.0,

palette: ['30b21c'],

};

Map.addLayer(cropland, croplandVis, 'Cropland', 0)

Map.addLayer(cropland\_affected, croplandVis, 'Affected Cropland');

var urbanVis = {

min: 0,

max: 13.0,

palette: ['grey'],

};

Map.addLayer(urban, urbanVis, 'Urban', 0)

Map.addLayer(urban\_affected, urbanVis, 'Affected Urban');

// Export images and vectors

Export.image.toDrive({

image: flooded,

description: 'Flood\_extent\_raster',

fileNamePrefix: 'flooded',

region: aoi,

maxPixels: 1e10

});

var flooded\_vec = flooded.reduceToVectors({

scale: 10,

geometryType:'polygon',

geometry: aoi,

eightConnected: false,

bestEffort:true,

tileScale:2,

});

Export.table.toDrive({

collection: flooded\_vec,

description: 'Flood\_extent\_vector',

fileFormat: 'SHP',

fileNamePrefix: 'flooded\_vec'

});

Export.image.toDrive({

image: population\_exposed,

description: 'Exposed\_Populuation',

scale: 250,

fileNamePrefix: 'population\_exposed',

region: aoi,

maxPixels: 1e10

});

// Create results panel

var results = ui.Panel({

style: {

position: 'bottom-left',

padding: '8px 15px',

width: '350px'

}

});

var textVis = {

'margin':'0px 8px 2px 0px',

'fontWeight':'bold'

};

var numberVIS = {

'margin':'0px 0px 15px 0px',

'color':'bf0f19',

'fontWeight':'bold'

};

var subTextVis = {

'margin':'0px 0px 2px 0px',

'fontSize':'12px',

'color':'grey'

};

var titleTextVis = {

'margin':'0px 0px 15px 0px',

'fontSize': '18px',

'font-weight':'',

'color': '3333ff'

};

var title = ui.Label('Results', titleTextVis);

var text1 = ui.Label('Flood status between:', textVis);

var number1 = ui.Label(after\_start.concat(" and ", after\_end), numberVIS);

var text2 = ui.Label('Estimated flood extent:', textVis);

var text2\_2 = ui.Label('Please wait...', subTextVis);

dates(after\_collection).evaluate(function(val){text2\_2.setValue('based on Sentinel-1 imagery '+val)});

var number2 = ui.Label('Please wait...', numberVIS);

flood\_area\_ha.evaluate(function(val){number2.setValue(val+' hectares')}), numberVIS;

var text3 = ui.Label('Estimated number of exposed people: ', textVis);

var text3\_2 = ui.Label('based on GHSL 2015 (250m)', subTextVis);

var number3 = ui.Label('Please wait...', numberVIS);

number\_pp\_exposed.evaluate(function(val){number3.setValue(val)}), numberVIS;

var MODIS\_date = ee.String(LC.get('system:index')).slice(0,4);

var text4 = ui.Label('Estimated affected cropland:', textVis);

var text4\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text4\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number4 = ui.Label('Please wait...', numberVIS);

crop\_area\_ha.evaluate(function(val){number4.setValue(val+' hectares')}), numberVIS;

var text5 = ui.Label('Estimated affected urban areas:', textVis);

var text5\_2 = ui.Label('Please wait', subTextVis)

MODIS\_date.evaluate(function(val){text5\_2.setValue('based on MODIS Land Cover '+val +' (500m)')}), subTextVis;

var number5 = ui.Label('Please wait...', numberVIS);

urban\_area\_ha.evaluate(function(val){number5.setValue(val+' hectares')}), numberVIS;

results.add(ui.Panel([

title,

text1,

number1,

text2,

text2\_2,

number2,

text3,

text3\_2,

number3,

text4,

text4\_2,

number4,

text5,

text5\_2,

number5,

]));

Map.add(results);

// Create legend

var legend = ui.Panel({

style: {

position: 'bottom-right',

padding: '8px 15px',

}

});

var legendTitle = ui.Label('Legend', titleTextVis);

legend.add(legendTitle);

var makeRow = function(color, name) {

var colorBox = ui.Label({

style: {

backgroundColor: color,

padding: '8px',

margin: '0 0 4px 0'

}

});

var description = ui.Label({

value: name,

style: {margin: '0 0 4px 6px'}

});

return ui.Panel({

widgets: [colorBox, description],

layout: ui.Panel.Layout.Flow('horizontal')

});

};

var palette =['#0000FF', '#30b21c', 'grey'];

var names = ['potentially flooded areas','affected cropland','affected urban'];

for (var i = 0; i < 3; i++) {

legend.add(makeRow(palette[i], names[i]));

}

var legendTitle2 = ui.Label({

value: 'Exposed population density',

style: {

fontWeight: 'bold',

fontSize: '15px',

margin: '10px 0 0 0',

padding: '0'

}

});

legend.add(legendTitle2);

var lon = ee.Image.pixelLonLat().select('latitude');

var gradient = lon.multiply((populationExposedVis.max-populationExposedVis.min)/100.0).add(populationExposedVis.min);

var legendImage = gradient.visualize(populationExposedVis);

var panel = ui.Panel({

widgets: [

ui.Label('> '.concat(populationExposedVis['max']))

],

});

legend.add(panel);

var thumbnail = ui.Thumbnail({

image: legendImage,

params: {bbox:'0,0,10,100', dimensions:'10x50'},

style: {padding: '1px', position: 'bottom-center'}

});

legend.add(thumbnail);

var panel = ui.Panel({

widgets: [

ui.Label(populationExposedVis['min'])

],

});

legend.add(panel);

Map.add(legend);

}