CODES

Eugenio Straffelini, Massimiliano N. Lippa, Paolo Tarolli
Department of Land, Environment, Agriculture and Forestry, University of Padova, Agripolis, Viale dell'Università 16, 35020 Legnaro (PD), Italy
Correspondence paolo.tarolli@unipd.it

Hot and Dry Days (HDD)

```
// Define our inputs, model and thresholds.
var modelName = 'MODEL HERE';         // Model name for both scenarios.
var temperatureThreshold = 35;       // °C
var precipitationThreshold = 1;      // mm/day

//We used a general ROI over middle and southern Europe because we then extract
// in R over the smaller olive growing areas.

// First we set over the entire year.
function processScenario(scenario, startYear, endYear) {
  // Define the date range for this scenario.
  var startDate = ee.Date.fromYMD(startYear, 1, 1);
  var endDate   = ee.Date.fromYMD(endYear, 12, 31);

  // We need to take our precipitation data from the band "pr" over March-November.
  var prCollection = ee.ImageCollection('NASA/GDDP-CMIP6')
    .select('pr')
    .filter(ee.Filter.eq('scenario', scenario))
    .filter(ee.Filter.eq('model', modelName))
    .filterDate(startDate, endDate)
    .filter(ee.Filter.calendarRange(3, 11, 'month'));

  // Here we take our temperature data from max temperature band "tasmax".
  var tasmaxCollection = ee.ImageCollection('NASA/GDDP-CMIP6')
    .select('tasmax')
    .filter(ee.Filter.eq('scenario', scenario))
    .filter(ee.Filter.eq('model', modelName))
    .filterDate(startDate, endDate)
    .filter(ee.Filter.calendarRange(3, 11, 'month'));

  // Here we start to join our two variables of interest.
  var join = ee.Join.inner();
  var filterTimeEq = ee.Filter.equals({
    leftField: 'system:time_start',
    rightField: 'system:time_start'
  });
  var joinedCollection = join.apply(prCollection, tasmaxCollection, filterTimeEq);

  // Now that we joined, we convert and make a binary mask based on our thresholds of interest.
  var hotDryCollection = joinedCollection.map(function(feature) {
```

```
    var prImg  = ee.Image(feature.get('primary'));
    var tasImg = ee.Image(feature.get('secondary'));

    // Convert precipitation to mm/day, threshold defined at the top.
    var prMm = prImg.multiply(86400);
    var dry  = prMm.lt(precipitationThreshold);

    // Convert temperature to °C, threshold is defined at the top.
    var tasC = tasImg.subtract(273.15);
    var hot  = tasC.gte(temperatureThreshold);

    // Combine the two conditions.
    var hotAndDry = dry.and(hot).rename('hotDry').multiply(1);

    // Propagate the date property.
    return hotAndDry.set('system:time_start', prImg.get('system:time_start'));
  });

  var hdCollection = ee.ImageCollection(hotDryCollection);

  // Combined the daily images into counts by year.
  var years = ee.List.sequence(startYear, endYear);
  var yearlyHotDry = ee.ImageCollection.fromImages(
    years.map(function(y) {
      y = ee.Number(y);
      // Below is where we are summing our days.
      var yearImages = hdCollection.filter(ee.Filter.calendarRange(y, y, 'year'));
      var hotDryCount = yearImages.sum().toInt16().rename('hotDryCount');
      return hotDryCount.set('year', y)
                        .set('system:time_start', ee.Date.fromYMD(y, 3, 1))
                        .set('scenario', scenario);
    })
  );

  return yearlyHotDry;
}

// Here we start to process our counts across both scenarios, historial and SSP585.

var historicalCollection = processScenario('historical', 1984, 2014);
var sspCollection        = processScenario('ssp585',    2015, 2100);

// Merge the two ImageCollections to create 1984-2100 collection.
var combinedYearlyCollection = historicalCollection.merge(sspCollection);

// Here we rename the bands to hot and dry days with year and scenario.
var renamedCollection = combinedYearlyCollection.map(function(img) {
  var year = ee.Number(img.get('year'));
  var scenario = ee.String(img.get('scenario'));
  var yearString = year.format('%d');
  var newBandName = ee.String('hotdryDay_')
```

```
                    .cat(modelName).cat('_')
                    .cat(yearString);
  return img.rename(newBandName);
});

// Having renamed our bands, we create a multi-band stack we can export.
var hotDryStack = renamedCollection.toBands().clip(roi);

// Here we export the multi-band stack.
Export.image.toDrive({
  image: hotDryStack,
  description: 'HotAndDryDays_Stack_1984_2100_' + modelName,
  folder: 'FOLDER',
  scale: 27830,
  region: roi,
  crs: 'EPSG:3035',
  maxPixels: 1e13,
  fileFormat: 'GeoTIFF'
});
```

Frost Days (FTD)

```
var geometry = ee.Geometry.Rectangle(
  [-11.60467728735416, 33.44877039995147, 46.227353962645836, 48.003818173442056],
  null,
  false  //All this ensures we have a nice rectangle for our study area.
);

// Center the map and add the rectangle
Map.centerObject(geometry, 4);
Map.addLayer(geometry, {color: 'red'}, 'Fixed Rectangle Geometry');


// Define the threshold for frost days (0°C).
var threshold = 0;

// Select the model. (The scenario will be chosen later based on the frost year.)
var modelName = 'MODEL HERE';

// Define the resolution of the dataset.
var scale = 27830;


// Single Frost Year Layer from December 1 to March 31.
var frostYear = 2070;
var seasonStart = ee.Date.fromYMD(frostYear - 1, 12, 1);
var seasonEnd   = ee.Date.fromYMD(frostYear, 4, 1);

// For the single frost season we use the SSP585 scenario.
```

```
var frostSeasonData = ee.ImageCollection('NASA/GDDP-CMIP6')
  .filter(ee.Filter.eq('model', modelName))
  .filter(ee.Filter.eq('scenario', 'ssp585'))
  .select('tasmin')
  .filterDate(seasonStart, seasonEnd);

// We convert from Kelvin to Celsius, appy a 1/0 mask of our threshold and sum.
var singleFrostSeason = frostSeasonData.map(function(img) {
  return img.subtract(273.15).lt(threshold);
}).sum()
.rename('frost_days');

// Add layers to the map.
Map.centerObject(geometry);
Map.addLayer(geometry, {color: 'red'}, 'Geometry');
Map.addLayer(singleFrostSeason.clip(geometry),
  {min: 0, max: 150, palette: ['white', 'blue']}, //150 because it captures all days in the period of
interest.
  'Frost Season (DJFM) for ' + frostYear,
  true
);

// Export the single frost season image.
Export.image.toDrive({
  image: singleFrostSeason.clip(geometry).float(),
  description: 'FrostSeason_DJFM_' + frostYear + '_' + modelName,
  scale: scale,
  region: geometry,
  crs: 'EPSG:3035'
});


// Multi-Year Frost Days Raster Creation.
// Define our range, in this case 1984-2100.
// Same as single year, we are interested in December 1 of previous year to March 31 of next.
var frostYearStart = 1984;
var frostYearEnd = 2100;

// Create a list of frost years.
var frostYearList = ee.List.sequence(frostYearStart, frostYearEnd);

// We create an ImageCollection where each image represents frost each period of interest,
December to March, for each year.
// For years up to 2014 we use the historical scenario and for 2015 onwards, we use ssp585.
var frostSeasonCollection = ee.ImageCollection(
  frostYearList.map(function(year) {
    year = ee.Number(year);
    // Any year 2014 or before is historical, onwards in SSP585.
    var scenario = ee.Algorithms.If(year.lte(2014), 'historical', 'ssp585');
    scenario = ee.String(scenario);
```

```
    var seasonStart = ee.Date.fromYMD(year.subtract(1), 12, 1);
    var seasonEnd = ee.Date.fromYMD(year, 4, 1);   //April 1 is not included but March 31 is
given how enddate functions.

    var frostSeason = ee.ImageCollection('NASA/GDDP-CMIP6')
      .filter(ee.Filter.eq('model', modelName))
      .filter(ee.Filter.eq('scenario', scenario))
      .select('tasmin')
      .filterDate(seasonStart, seasonEnd)
      .map(function(img) {
        // Here is our conversion from Kelvin to Celsius.
        return img.subtract(273.15).lt(threshold);
      })
      .sum()
      .rename('frost_days'); //Here is where we rename and sum.

    return frostSeason
      .set('system:time_start', seasonStart.millis())
      .set('system:index', year.format('%d'))
      .set('scenario', scenario);
  })
);

// To understand the trend over our area of interest, we print a mean timeseries.
print(
  ui.Chart.image.series({
    imageCollection: frostSeasonCollection,
    region: geometry,
    reducer: ee.Reducer.mean(),
    scale: scale,
    xProperty: 'system:time_start'
  }).setOptions({
    title: 'Yearly Frost Seasons (DJFM) - ' + modelName + ' (historical / ssp585)',
    vAxis: {title: 'Mean frost days'},
    hAxis: {title: 'Frost Season Start Date'}
  })
);

// Here we have the first layer added for inspection and checks.
var firstSeason = ee.Image(frostSeasonCollection.first());
Map.addLayer(firstSeason.clip(geometry),
  {min: 0, max: 150, palette: ['white', 'blue']},
  'Frost Season (DJFM) for ' + frostYearStart,
  false
);

// Here, we are renaming the bands with a prefix.
var prefix = 'FrostDays_Stack_' + frostYearStart + '_' + frostYearEnd + '_' + modelName +
'_lessthan0';

// For each image, rename its 'frost_days' band to include the prefix and the frost year.
```

```
var renamedFrostSeasonCollection = frostSeasonCollection.map(function(img) {
  var year = ee.String(img.get('system:index'));
  var newBandName = ee.String(prefix).cat('_').cat(year);
  return img.select(['frost_days'], [newBandName]);
});

// We convert to a multi-band image for improve workflow in GIS software.
var frostSeasonMultiBand = renamedFrostSeasonCollection.toBands().clip(geometry);

// Export our multi-band stack.
Export.image.toDrive({
  image: frostSeasonMultiBand.float(),
  description: prefix,
  scale: scale,
  region: geometry,
  crs: 'EPSG:3035'
});
```

Chill Days (CHD)

```
// Here, we define our thresholds of interest.
var lowerBound = 0;
var upperBound = 7;

// Here, we define some of our input parameters, the model, scenario, and scale.
var modelName = 'MODEL HERE';
var futureScenario = 'ssp585';
var scale = 27830;

// Here we look at just a single season to help check for issues.
var analysisYear = 2015;
var seasonStart = ee.Date.fromYMD(analysisYear - 1, 12, 1);
var seasonEnd   = ee.Date.fromYMD(analysisYear, 4, 1);

// We defined model and scenario above, and select the band "tasmin".
var seasonData = ee.ImageCollection('NASA/GDDP-CMIP6')
  .filter(ee.Filter.eq('model', modelName))
  .filter(ee.Filter.eq('scenario', futureScenario))
  .select('tasmin')
  .filterDate(seasonStart, seasonEnd);

// Here we make our conversion from Kelvin to Celsius and sum.
var seasonCount = seasonData.map(function(img) {
  var tempC = img.subtract(273.15);
  return tempC.gte(lowerBound).and(tempC.lte(upperBound));
}).sum();

// Add the single year layer onto the map.
Map.centerObject(geometry);
Map.addLayer(geometry, {color: 'red'}, 'Geometry');
```

```
Map.addLayer(seasonCount.clip(geometry),
  {min: 0, max: 150, palette: ['white', 'blue']},
  'Days 0-7°C (DJFM) for ' + analysisYear,
  true
);

// Export the single season image.
Export.image.toDrive({
  image: seasonCount.clip(geometry).float(),
  description: 'Days_0to7_DJFM_' + analysisYear + '_' + modelName,
  scale: scale,
  region: geometry,
  crs: 'EPSG:3035',
  folder: 'FOLDER'
});


// Multi Year 1984-2100. In this section, we are going to creating our multi-band rasters.
var startYear = 1984;
var endYear = 2100;
var transitionYear = 2015;

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

// Here we define our time period, December of the previous year to March of the following.
var seasonCollection = ee.ImageCollection(
  yearList.map(function(year) {
    year = ee.Number(year);
    var startDate = ee.Date.fromYMD(year.subtract(1), 12, 1);
    var endDate = ee.Date.fromYMD(year, 4, 1);
    var yearString = year.format('%d');

    // Here we make the band name.
    var bandName = ee.String('Dormancy Days ')
                    .cat(yearString)
                    .cat(' ')
                    .cat(modelName);

    // With our transition year of 2015, anything less is historical and above is future.
    var scenarioForYear = ee.String(ee.Algorithms.If(
      year.lt(transitionYear), 'historical', futureScenario));

    var seasonImage = ee.ImageCollection('NASA/GDDP-CMIP6')
      .filter(ee.Filter.eq('model', modelName))
      .filter(ee.Filter.eq('scenario', scenarioForYear))
      .select('tasmin')
      .filterDate(startDate, endDate)
      .map(function(img) {
        var tempC = img.subtract(273.15);
        return tempC.gte(lowerBound).and(tempC.lte(upperBound));
      })
```

```
      .sum()
      .rename(bandName);

    return seasonImage
      .set('system:time_start', startDate.millis())
      .set('system:index', yearString);
  })
);

// Here we create a timeseries. The chart points of means of our geometry, but the rasters are
sums.
print(
  ui.Chart.image.series({
    imageCollection: seasonCollection,
    region: geometry,
    reducer: ee.Reducer.mean(),
    scale: scale,
    xProperty: 'system:time_start'
  }).setOptions({
    title: 'Yearly Count of Days 0-7°C (DJFM) - ' + modelName + ' (historical/ssp585)',
    vAxis: {title: 'Mean count of days'},
    hAxis: {title: 'Season Start Date'}
  })
);

// To check for issues, we add the layer of the first year, 1984.
var firstSeason = ee.Image(seasonCollection.first());
Map.addLayer(firstSeason.clip(geometry),
  {min: 0, max: 150, palette: ['white', 'blue']},
  'Days 0-7°C (DJFM) for ' + startYear,
  false
);

// Export the multi-year season stack as a multiband image.
var seasonMultiBand = seasonCollection.toBands().clip(geometry);
Export.image.toDrive({
  image: seasonMultiBand.float(),
  description: 'Dormancy_Days_MultiYear_0to7_' + modelName + '_' + startYear + '_' + endYear,
  scale: scale,
  region: geometry,
  crs: 'EPSG:3035',
  folder: 'FOLDER'
});
```

Warm and Humid Days (WHD)

```
// Here we set up and efine our ROI. It is over europe and then data is further extracted in R.
var roi = ee.Geometry.Rectangle(
  [-11.60467728735416, 33.44877039995147, 46.227353962645836, 48.003818173442056],
```

```
  null,
  false
);

// Here we an define our model of choice.
var modelName = 'MODEL HERE';

// We start with the general year and then filter for March to November.
function processScenario(scenario, startYear, endYear) {
  var startDate = ee.Date.fromYMD(startYear, 1, 1);
  var endDate   = ee.Date.fromYMD(endYear, 12, 31);
  var monthFilter = ee.Filter.calendarRange(3, 11, 'month');

  // For the Warm and Humid Days, we are concerned with humidty, minimum and max
temperature.
  var hursCollection = ee.ImageCollection('NASA/GDDP-CMIP6')
    .select('hurs')
    .filter(ee.Filter.eq('scenario', scenario))
    .filter(ee.Filter.eq('model', modelName))
    .filterDate(startDate, endDate)
    .filter(monthFilter);

  var tasminCollection = ee.ImageCollection('NASA/GDDP-CMIP6')
    .select('tasmin')
    .filter(ee.Filter.eq('scenario', scenario))
    .filter(ee.Filter.eq('model', modelName))
    .filterDate(startDate, endDate)
    .filter(monthFilter);

  var tasmaxCollection = ee.ImageCollection('NASA/GDDP-CMIP6')
    .select('tasmax')
    .filter(ee.Filter.eq('scenario', scenario))
    .filter(ee.Filter.eq('model', modelName))
    .filterDate(startDate, endDate)
    .filter(monthFilter);

  // We have to make our joins. We first join the humidity to min temp and then combine
  // that join with another join to max temp.
  var joinFilter = ee.Filter.equals({leftField: 'system:time_start', rightField: 'system:time_start'});
  var innerJoin = ee.Join.inner();

  var hursTasminJoined = ee.ImageCollection(innerJoin.apply(hursCollection, tasminCollection,
joinFilter))
    .map(function(joinedImage) {
      var hurs = ee.Image(joinedImage.get('primary'));
      var tasmin = ee.Image(joinedImage.get('secondary')).rename('tasmin');
      return hurs.addBands(tasmin);
    });

  var hursTasminTasmaxJoined = ee.ImageCollection(innerJoin.apply(hursTasminJoined,
tasmaxCollection, joinFilter))
```

```
    .map(function(joinedImage) {
      var hursTasmin = ee.Image(joinedImage.get('primary'));
      var tasmax = ee.Image(joinedImage.get('secondary')).rename('tasmax');
      return hursTasmin.addBands(tasmax);
    });

  // Here we the correct conversions and threshold.
  var suitableCollection = hursTasminTasmaxJoined.map(function(img) {
    var hursImg = img.select('hurs');
    var tasminC = img.select('tasmin').subtract(273.15);
    var tasmaxC = img.select('tasmax').subtract(273.15);

    var hursCond  = hursImg.gte(55).and(hursImg.lte(75));
    var tasminCond = tasminC.gte(7.5);
    var tasmaxCond = tasmaxC.lte(32);

    var suitableDay = hursCond.and(tasminCond).and(tasmaxCond)
                      .rename('suitableDay')
                      .multiply(1);

    return suitableDay.set('system:time_start', img.get('system:time_start'));
  });

  // Here we create our function to count suitable days given these joins.
  var years = ee.List.sequence(startYear, endYear);
  var yearlySuitable = ee.ImageCollection.fromImages(
    years.map(function(y) {
      y = ee.Number(y);
      var yearImages = suitableCollection.filter(ee.Filter.calendarRange(y, y, 'year'));
      var suitableCount = yearImages.sum().toInt16().rename('suitableCount');
      return suitableCount.set('year', y)
                          .set('system:time_start', ee.Date.fromYMD(y, 3, 1))
                          .set('scenario', scenario);
    })
  );

  return yearlySuitable;
}

// We define our scenarios.
var historicalCollection = processScenario('historical', 1984, 2014);
var sspCollection = processScenario('ssp585', 2015, 2100);

// Merge the two ImageCollections.
var combinedYearlyCollection = historicalCollection.merge(sspCollection);

// Here we rename the bands for our multi year band.
var renamedCollection = combinedYearlyCollection.map(function(img) {
  var year = ee.Number(img.get('year'));
  var scenario = ee.String(img.get('scenario'));
  var yearString = year.format('%d');
```

```
  var newBandName = ee.String('suitableDay_')
                    .cat(modelName).cat('_')
                    .cat(yearString);
  return img.rename(newBandName);
});

// We now need to convert to a multi year band.
var suitableDayStack = renamedCollection.toBands().clip(roi);

// We create a time series chart.
var timeSeries = combinedYearlyCollection.map(function(img) {
  var mean = img.reduceRegion({
    reducer: ee.Reducer.mean(),
    geometry: roi,
    scale: 27830,
    maxPixels: 1e13
  }).get('suitableCount');
  return ee.Feature(null, {
    year: img.get('year'),
    meanSuitableDays: mean,
    scenario: img.get('scenario')
  });
});

var timeSeriesChart = ui.Chart.feature.groups({
  features: timeSeries,
  xProperty: 'year',
  yProperty: 'meanSuitableDays',
  seriesProperty: 'scenario'
})
.setChartType('LineChart')
.setOptions({
  title: 'Mean Suitable Days per Year (March–November)',
  hAxis: {title: 'Year'},
  vAxis: {title: 'Suitable Days (mean over ROI)'},
  lineWidth: 2,
  pointSize: 4,
  series: {
    0: {color: 'blue'},
    1: {color: 'red'}
  }
});


print(timeSeriesChart);

// We export the multiband image.
Export.image.toDrive({
  image: suitableDayStack,
  description: 'SuitableDays_Stack_1984_2100_' + modelName,
  folder: 'FOLDER',
```

```
  scale: 27830,
  region: roi,
  crs: 'EPSG:3035',
  maxPixels: 1e13,
  fileFormat: 'GeoTIFF'
});

Map.addLayer(roi);
```