Evaluating an attempt to restore summer fire in the Northern Great Plains

Supplementary Information

Devan Allen McGranahan and Jay P. Angerer

This document provides script for fetching remotely-sensed imagery and processing results in the R statistical environment.

Remote sensing

Sentinel-2

This `EvalScript` can be used in the Copernicus browser as a Custom script to create a Custom visualization that can be exported as a 16-bit `.tiff` for raster analysis. The user must select the area of interest and imagery dates and is responsible for assessing cloudiness.

```
//VERSION=3
function setup() {
  return {
    input: ["B04", "B08", "B8A", "B11", "B12"],
    output: { bands: 2, sampleType: "UINT16" }
  };
}

function evaluatePixel(sample) {
  let nbr = index(sample.B08, sample.B12);
  let ndvi = index(sample.B08, sample.B04);
  // apply offset for UINT16
  return [10000 * nbr + 10000,
          10000 * ndvi + 10000];
}
```

Landsat

This script for Google Earth Engine exports a composited NDVI `.tiff` file to the user's Google Drive for the area of interest defined as `geometry` for an eight week period beginning with each date given in `listDate`. The script combines imagery from Landsat missions 5, 7, & 8 and handles cloud masking.

```
// Geometry
var geometry: Polygon, 4 vertices
  type: Polygon
  coordinates: List (1 element)
    0: List (5 elements)
      0: [-99.51276195869765,46.71741699210676]
      1: [-99.42521465645156,46.71741699210676]
      2: [-99.42521465645156,46.778346255312734]
      3: [-99.51276195869765,46.778346255312734]
      4: [-99.51276195869765,46.71741699210676]
  geodesic: false

// Main script

var batch = require('users/fitoprincipe/geetools:batch');

///****variables that need to be changed by user
//****ADD google drive directory name that you want to download files to
var gdrivedir = 'landsat'

var listDate= ["1992-04-20", "1992-07-20", "1993-04-20", "1993-07-20",
               "1994-04-20", "1994-07-20", "1995-04-20", "1995-07-20",
               "1996-04-20", "1996-07-20", "1997-04-20", "1997-07-20",
               "1998-04-20", "1998-07-20", "1999-04-20", "1999-07-20",
               "2000-04-20", "2000-07-20", "2001-04-20", "2001-07-20",
               "2002-04-20", "2002-07-20", "2003-04-20", "2003-07-20",
               "2004-04-20", "2004-07-20", "2005-04-20", "2005-07-20",
               "2006-04-20", "2006-07-20", "2007-04-20", "2007-07-20",
               "2008-04-20", "2008-07-20", "2009-04-20", "2009-07-20",
               "2010-04-20", "2010-07-20", "2011-04-20", "2011-07-20",
               "2012-04-20", "2012-07-20", "2013-04-20", "2013-07-20",
               "2014-04-20", "2014-07-20", "2015-04-20", "2015-07-20",
               "2016-04-20", "2016-07-20", "2017-04-20", "2017-07-20",
               "2018-04-20", "2018-07-20", "2019-04-20", "2019-07-20",
               "2020-04-20", "2020-07-20", "2021-04-20", "2021-07-20",
               "2022-04-20", "2022-07-20"]

listDate.forEach(function (listDate) {
  //yearRanges.forEach(function (yearRange) {
    exportTimeseries(listDate)
  })
//})
function exportTimeseries(listDate) {

  // Defines a base date/time for the following examples.
  var startDate = ee.Date(listDate);
  print(startDate, 'The start date/time');
  print(startDate.format('YYYYMMdd'))
  var sdate = startDate.format('YYYYMMdd')
```

```
var endDate = startDate.advance(8, 'week')
print(endDate, 'The end date/time');
var edate = endDate.format('YYYYMMdd')

print(endDate.format('YYYYMMdd'))

//from https://developers.google.com/earth-engine/tutorials/
//                community/extract-raster-values-for-points
//function to mask cloud and shadow pixels
function fmask(img) {
  var cloudShadowBitMask = 1 << 4;
  var cloudsBitMask = 1 << 3;
  var qa = img.select('QA_PIXEL');
  var mask = qa.bitwiseAnd(cloudShadowBitMask).eq(0)
    .and(qa.bitwiseAnd(cloudsBitMask).eq(0));
  return img.updateMask(mask);
}

// Selects and renames bands of interest for Landsat OLI.
function renameOli(img) {
  return img.select(
    ['SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B6', 'SR_B7'],
    ['Blue', 'Green', 'Red', 'NIR', 'SWIR1', 'SWIR2']);
}

// Selects and renames bands of interest for TM/ETM+.
function renameEtm(img) {
  return img.select(
    ['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'SR_B5', 'SR_B7'],
    ['Blue', 'Green', 'Red', 'NIR', 'SWIR1', 'SWIR2']);
}

// Prepares (cloud masks and renames) OLI images.
function prepOli(img) {
  img = fmask(img);
  //img = lcfmask(img)
  img = renameOli(img);
  return img;
}

// Prepares (cloud masks and renames) TM/ETM+ images.
function prepEtm(img) {
  img = fmask(img);
  //img = lcfmask(img)
  img = renameEtm(img);
  return img;
}

// Apply scaling factors
function applyScaleFactors(image) {
  var opticalBands = image.select('SR_B.').multiply(0.0000275).add(-0.2);
  var thermalBands = image.select('ST_B.*').multiply(0.00341802).add(149.0);
  return image.addBands(opticalBands, null, true)
              .addBands(thermalBands, null, true);
```

```
  }

  // Get surface reflectance collections for Landsat
  // maps scaling factors and cloud masking functions

  var oliCol = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
              .filter(ee.Filter.bounds(geometry))
              .map(applyScaleFactors)
              .map(prepOli);

  var etmCol = ee.ImageCollection('LANDSAT/LE07/C02/T1_L2')
                .filter(ee.Filter.bounds(geometry))
                .map(applyScaleFactors)
                .map(prepEtm);

  var tmCol = ee.ImageCollection('LANDSAT/LT05/C02/T1_L2')
              .filter(ee.Filter.bounds(geometry))
              .map(applyScaleFactors)
              .map(prepEtm) ;

  var landsatCol = oliCol.merge(etmCol).merge(tmCol)
                   .filterDate(startDate, endDate) ;

  // Calculate NDVI
  var addIndices = function(image) {
     var ndvi = image.normalizedDifference(['NIR', 'Red'])
                .rename('ndvi').float();
  return image.addBands([ndvi]);
  };

  var indices = landsatCol.map(addIndices)
                .select('ndvi');
  // Create composite of images from date range
    var composite = indices.mean() ;

    var fileName = ee.Date(startDate)
                  .format('yyyy-MM-dd')
                  .getInfo() ;

  Export.image.toDrive({
      image: composite,
      description: fileName,
      scale: 30,
      folder: 'landsat',
      region: geometry
      });
}
```

Analysis

Because the analysis script assumes hundreds of imagery files and a large Excel file with 42 years of hourly weather observations have been saved locally, it cannot be run directly from this document but is provided here for transparency and reference.

Remotely sensed imagery

The following R script processes the imagery produced by the scripts above after they have been saved to a drive that can be mapped to from the session as `imagery_dir`.

```
# Load necessary packages
# Note that package terra is required but is not loaded
# (called directly to not create conflicts with dplyr verbs)
  pacman::p_load(tidyverse, sf, stars, foreach, doSNOW)

# Load location boundaries (data not provided)
  cgrec_gpkg = './CGREC_PBG_26914.gpkg'

  pastures <- st_read(cgrec_gpkg, 'Pastures')
  patches <- st_read(cgrec_gpkg, 'PasturePatches')
#
# Get veg data for unburned pastures
#
  NoFirePts <- st_read(cgrec_gpkg, 'SamplePoints') %>%
               filter(location == 'Refuge')
  # Map to directory with Sentinel-2 imagery
    imagery_dir = 'C:/Path/To/Sentinel'
    images <- list.files(imagery_dir)
  # Use parallel processing to chug imagery
  {
    begin = Sys.time()
    pacman::p_load()
    cores = parallel::detectCores()
    cl <- makeCluster(cores, methods = F, useXDR = F)
    registerDoSNOW(cl)
    NoFireIndices <-
      foreach(i=1:length(images),
              .combine = 'bind_rows',
              .errorhandling = 'remove',
              .packages=c('tidyverse', 'sf')) %dopar% {
        image = images[i]
        image_path = paste0(imagery_dir, '/', image)
        ras <- terra::rast(image_path)
        names(ras) <- c('nbr', 'ndvi')
        ras <- ras[['ndvi']]
        float = (ras-10000)/10000
        terra::extract(float,
          NoFirePts %>%
            select(pasture, sample) %>%
            terra::vect() ,
          FUN = mean,
          bind = TRUE)  %>%
        st_as_sf() %>%
```

```
          as_tibble() %>%
        mutate(ImageDate = substr(image, 1, 10)) %>%
        select(ImageDate, pasture, sample, ndvi)
            }
    stopCluster(cl)
  Sys.time() - begin
  }
#
# Get fuel greenness and dNBR for completed burns
#
  fires <- st_read(cgrec_gpkg, 'FirePerimeters') %>%
             mutate(Year = as.factor(Year)) %>%
             filter(status == 'Completed') %>%
             unite('fire', c(unit, Pasture, Patch), sep = "-") %>%
             select(fire, Year, Season, PreBurn, PostBurn )
  SamplePts <-
    st_read(cgrec_gpkg, 'SamplePointsRegular')
{
  begin = Sys.time()
  pacman::p_load(foreach, doSNOW)
  cores = parallel::detectCores()
  cl <- makeCluster(cores , methods = F, useXDR = F)
  registerDoSNOW(cl)
  BurnIndices <-
    foreach(i=1:length(fires$fire),
            .combine = 'bind_rows',
            .errorhandling = 'remove',
            .packages=c('tidyverse', 'sf')) %dopar% {
              # Get fire
                fire = slice(fires, i)
              # Get sample points
                pts <-
                  fire %>%
                    st_intersection(SamplePts)
              # Get dates
                pre_date = fire$PreBurn
                post_date = fire$PostBurn
              # Fetch & process multi-band rasters
                # pre image
                  pre_image = images[substr(images, 1, 10) == pre_date]
                  pre_path = paste0(imagery_dir, '/', pre_image)
                  pre_ras <- terra::rast(pre_path) %>%
                               terra::crop(terra::vect(fire))
                  names(pre_ras) <- c('nbr', 'ndvi')
                  pre_ras <-  (pre_ras-10000)/10000
                # post-fire
                  post_image = images[substr(images, 1, 10) == post_date]
                  post_path = paste0(imagery_dir, '/', post_image)
                  post_ras <- terra::rast(post_path)[[1]] %>%
                               terra::crop(terra::vect(fire))
                  post_ras = (post_ras-10000)/10000
                # Calculate dNBR & replace in pre-fire raster
                  d_ras = pre_ras['nbr'] - post_ras
                  names(d_ras) <- 'dNBR'
```

```
                pre_ras[[1]] <- d_ras
              # Sample rasters
                terra::extract( pre_ras,
                          pts %>%
                            select(-PostBurn) %>%
                            terra::vect() ,
                          FUN = mean,
                          bind = TRUE)  %>%
                 st_as_sf() %>%
                   as_tibble() %>%
                   select(-geometry)
            }
  stopCluster(cl)
  Sys.time() - begin
}
#
# Get historical Landsat data for unburned pastures
#
  imagery_dir = 'C:/Path/To/Landsat'
  images <- list.files(imagery_dir)
  {
    begin = Sys.time()
    pacman::p_load(foreach, doSNOW)
    cores = parallel::detectCores()
    cl <- makeCluster(cores, methods = F, useXDR = F)
    registerDoSNOW(cl)
    NoFireTrends <-
      foreach(i=1:length(images),
              .combine = 'bind_rows',
              .errorhandling = 'remove',
              .packages=c('tidyverse', 'sf')) %dopar% {
                image = images[i]
                image_path = paste0(imagery_dir, '/', image)
                ras <- terra::rast(image_path)
                ras <- ras[['ndvi']]
                 terra::extract(ras,
                                NoFirePts %>%
                                  select(pasture, sample) %>%
                                  st_transform(4326) %>%
                                  terra::vect() ,
                                FUN = mean,
                                bind = TRUE)  %>%
                  st_as_sf() %>%
                  as_tibble() %>%
                  mutate(ImageDate = tools::file_path_sans_ext(image))  %>%
                  select(ImageDate, pasture, sample, ndvi)
              }
    stopCluster(cl)
    Sys.time() - begin
  }
```

Weather data

This script wrangles the weather data downloaded from the North Dakota Ag Weather Network's Streeter station.

```
pacman::p_load(tidyverse, readxl)

seasons <- tibble(season = c('spring', 'summer'),
                  start = c('04-20', '07-20'),
                  end = c('06-10', '09-10') ) %>%
           mutate(start = as.Date(start, '%m-%d'),
                  end = as.Date(end, '%m-%d'))

BurnDays <-
  read_xlsx('./data/BurnData.xlsx', 'BurnDays') %>%
  filter(is.na(certainty)) %>%
    mutate(date = paste(year, date),
           date = as.Date(date, '%Y %B %d')) %>%
    select(date) %>%
  distinct()

WxData <- lst()
# Get daily data
  DailyWx <-
    read_xlsx('./data/CGREC_weather.xlsx', 'daily')
# Identify rainy days
  WxData$Rainfall <-
    DailyWx %>%
    select(Year, Month, Day, Rainfall) %>%
    unite( c(Month, Day), col = 'day', sep = '-', remove = F) %>%
    mutate(day = as.Date(day, '%m-%d')  ,
           season = case_when(
             between(day, seasons$start[1], seasons$end[1]) ~'Spring',
             between(day, seasons$start[2], seasons$end[2]) ~'Summer',
             TRUE ~ NA
           ) ) %>%
     unite( c(Year, Month, Day), col = 'date', sep = '-')

# Hourly data
  WxData$HourlyWx <-
    read_xlsx('./data/CGREC_weather.xlsx', 'hourly') %>%
    filter(between(Hour, 1000, 1700)) %>%
    unite( c(Year, Month, Day), col = 'date', sep = '-', remove = F) %>%
    unite( c(Month, Day), col = 'day', sep = '-', remove = F) %>%
    mutate(day = as.Date(day, '%m-%d'),
           season = case_when(
             between(day, seasons$start[1], seasons$end[1]) ~'Spring',
             between(day, seasons$start[2], seasons$end[2]) ~'Summer',
             TRUE ~ NA
           ) ) %>%
    mutate(e = 6.11 * (10 ^ ( (7.5 * DewPoint)/ (237.3 + DewPoint) ) ),
           es = 6.11 * (10 ^ ( (7.5 * AirTemp)/ (237.3 + AirTemp) ) ),
           VPD = es - e)
# Get burn day weather
  WxData$BurnDayWx <-
```

```
HourlyWx  %>%
filter(date %in% BurnDays$date) %>%
group_by(Year, date, season) %>%
summarise_at(.vars = vars(c(DewPoint, RelHum, WindSpeed,VPD)),
             .funs = c("mean")) %>%
ungroup() %>%
pivot_longer(names_to = 'variable',
             values_to = 'value',
             cols = DewPoint:VPD)
```