Last Updated: March 2022

EXERCISE 4
Run LT-GEE to map disturbance

Introduction

Now that you have explored the LT-GEE algorithm and learned how to fit trajectories to map change, you are ready to learn how to run the algorithm in the Earth Engine code editor to make a map! In the last exercise, you used UI tools to obtain a set of parameters for segmenting change on the Dixie NF in the area affected by the 2017 Brian Head fire. In this exercise, we will use those parameters and our knowledge of image processing to build and execute a complete LandTrendr script and download results.

Objectives

- Use the LT-GEE module to build, process Landsat Time Series data and run LandTrendr
- Create and download a map from LT-GEE output

Required Data

- Everything is in Earth Engine!

Prerequisites

- An approved Earth Engine account
- Read access to the EMAPR LT-GEE public repository (click this link to add if necessary)
- Completion of exercises 2 and 3 for Advanced Change Detection. Optionally, review the provided script for completion of exercise 2.

Table of Contents

Part 1: Explore information on LT-GEE outputs

The core process of LandTrendr is segmentation of Landsat image pixel values through time. The Pixel Time Series Plotter tool visualizes the linear segmentation and model fitting that the LandTrendr algorithm applies to each pixel in an area of interest. The Change Mapper tool visualizes these products over a landscape. But how do we extract map layers to use for our own analyses or visualization? This process can seem a bit complicated at first. To get a feel for what map products the LT-GEE algorithm produces, we will start by reviewing the documentation.

A. Review documentation on LT-GEE outputs

1. Navigate to this link in the LT-GEE documentation and review the diagram illustrating data returned by LT-GEE.
2. Note that LT-GEE returns the following:
   i. The year of observations and whether each year represents a vertex (i.e., a turning point or change year).
   ii. The original source values before segmentation (i.e., composite NBR value).
   iii. The fitted values (i.e., the model fitted NBR value).
   iv. The root mean square error or difference between fitted and source (observed) values.

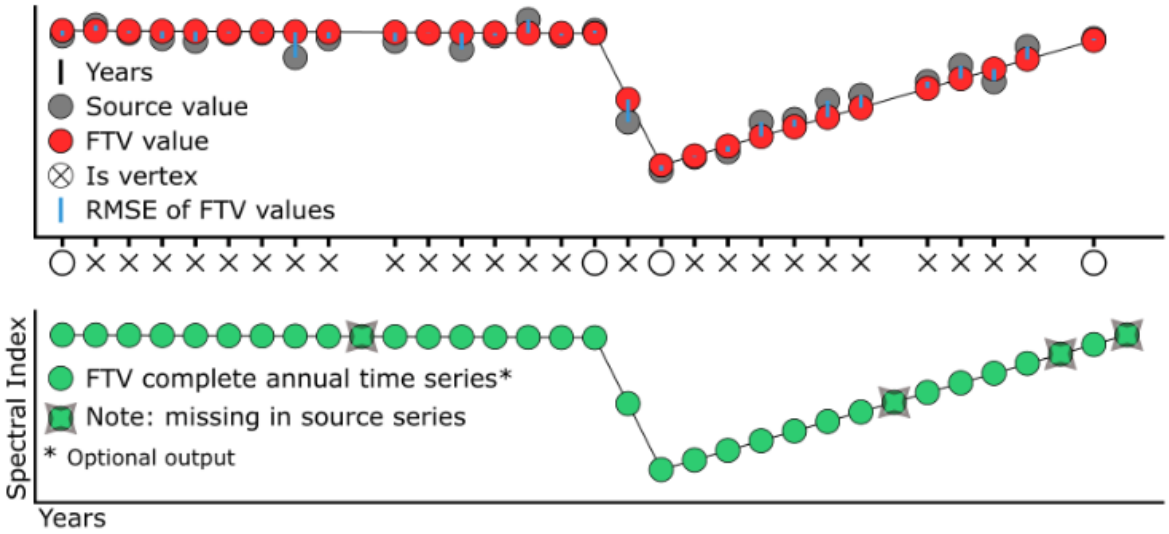
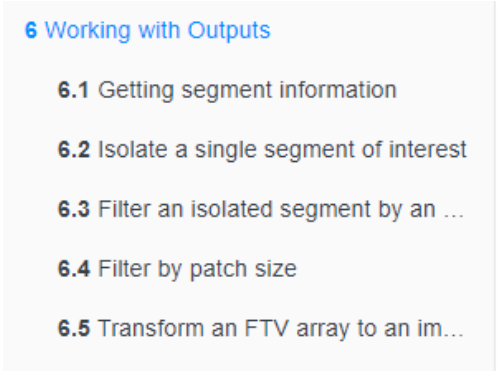


Fig 5.1. A visual diagram of what data are returned from LT-GEE. Every legend item is returned as an output.

3. Note that the LT-GEE outputs need to be 'unpacked' in order to create a disturbance map.
   i. There are example code snippets for 'extracting' various outputs from the image bands returned in a LandTrendr result image object.

B. Review LT-GEE documentation on working with outputs

1. Using the menu on the left, navigate to section 6 to learn how to work with LandTrendr outputs in Earth Engine



2. Read *just the introduction* section titled "Working with outputs".

There is a lot of information in the documentation and you certainly don't need to understand all of it. If all of this stuff about projecting and flattening arrays sounds complicated to you, you are not alone! Working with complex arrays can be quite confusing and take some time to get used to. Luckily, reading the documentation, we find that the developers behind the LT-GEE module have created shortcut functions to make extracting map products much easier! We will look at these next.

C. Review API functions for running LandTrendr

1. Use the navigation on the left to open the documentation on LT-GEE functions.
2. Scroll down to section 9.1 for the documentation of the central functions implemented by LT-GEE.
3. Read the description of the **buildSRcollection** function.
   i. This function is the initial processing step in preparing imagery for use in the LandTrendr analysis. It creates annual composites for the years of interest, applying specified masks, and returns an image collection where each year is represented by one image.
4. Read the description of the **buildLTcollection** function.
   i. After the initial image collection is created by the **buildSRcollection** function, this function adds the spectral transformations with the first band being the index you want to use for segmentation. In our case, this will be NBR.
5. Read the description of the **runLT** function.
   i. This is the command that takes parameters for LandTrendr and initializes the algorithm.
   ii. **runLT** takes the image processing parameters we explored, as well as the segmentation parameters that we found using the Pixel Time Series Plotter.
   iii. We can see that this function is a wrapper around the now-familiar **buildSRcollection** and **buildLTcollection**.

D. Read the docs on shortcut functions to extracting map layers

1. Scroll down or use the menu to navigate to the **getChangeMap** function and read the description.

i. This function takes the output of the LandTrendr segmentation from the **runLT** function and a set of change parameters.

ii. This sounds good and so much easier than slicing complex arrays with JavaScript!

iii. The change parameters are the same parameters that we explored with the LandTrendr Change Mapper tool and are illustrated nicely in the graphic below from the documentation.

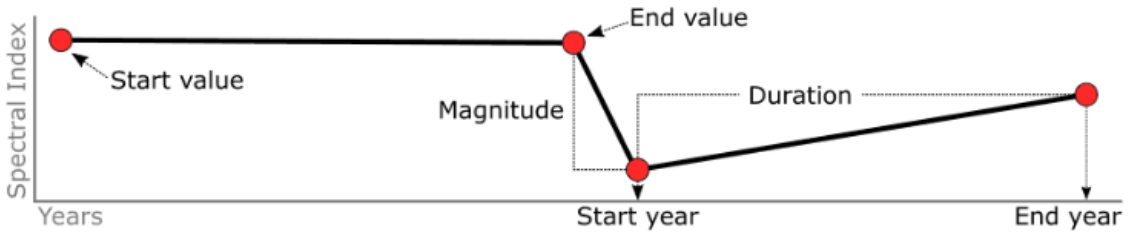


Fig 2.6. Diagram of segment attributes. From these attributes we can summarize and query change per pixel over the landscape.

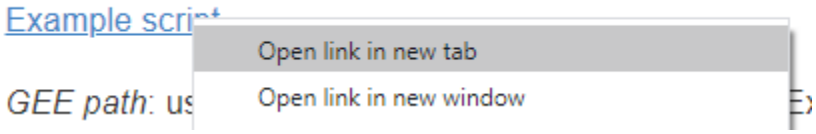
Perfect! We now have all the pieces that we need to run LandTrendr in the EE Code Editor where we can save or download our resulting change map data!

Part 2: Review the example script for mapping loss

The LT-GEE module documentation comes with example scripts that we can easily adapt to our study area. We will first review to ensure we understand how all the pieces fit together. Then, we will pass in the variables for our example study area affected by the Brian Head fire.

A. Open an example script for mapping vegetation loss.

1. In the documentation, use the menu to navigate to the **Example Scripts** section (section 7).
2. Read the section titled **Map vegetation loss**. This is the script that we will use as a template for creating a disturbance map for the Dixie NF Brian Head area.
3. Right click on the link the **Example script** and select **Open link in new tab** (shown below).
   i. This will open the script in the code editor.



B. Review the user inputs

1. Lines 1-20 provide contact information and resources for learning more about LT-GEE.
2. Lines 27-33 define the parameters for the Landsat Time Series processing including years, seasonal dates, masking parameters and the spectral index to use for segmentation.

3. Lines 36-45 creates a JavaScript object (also called a dictionary) for the LandTrendr segmentation parameters.
   i. Here, we will use the parameters we found with the UI Pixel Time Series Plotter.
4. Lines 48-56 define the parameters for 'unpacking' the LandTrendr output to obtain a disturbance map.
   i. These are the parameters we explored with the UI Change Mapper.

C. Review the main script body

1. Read through lines 63-105 and check your knowledge.
   i. Where and how are the special LT-GEE functions being imported?
   ii. Can you differentiate the LT-GEE functions from Earth Engine functions?
      (a) Hint – look for the module name, ltgee, followed by dot notation.
   iii. Where is the LandTrendr segmentation process being called?
   iv. Where is the disturbance map getting created?
   v. What kind of JavaScript objects are the following: **changeImg**, **pallete**, **magVizParms** and **region**? If you don't know the answer, how could you use the code editor to find out?
2. Answers appear in the appendix!

***Note**: You do not necessarily need to edit any of the JavaScript after the user inputs section in order to run this script and generate a disturbance map. However, it is a good idea to see how things work and to test your knowledge. You might find that much of the code in this script is familiar if you take your time going through it, unraveling it bit by bit. You could create something like this on your own with some patience and practice!*

Part 3: Adapt and run the script to get a disturbance map

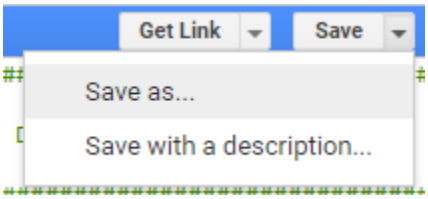
You have now seen all the pieces that you need to run LandTrendr in the Code Editor to obtain a disturbance map. We will refer to the parameters we found using the two UI applications, edit the script, run it and download our results.

A. Add notes and save your copy of the script

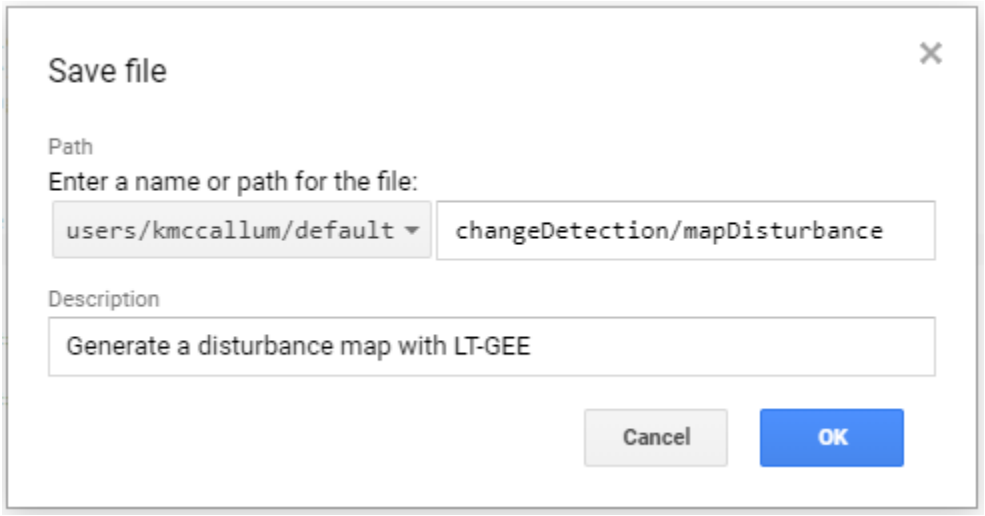
1. On line 21 add a commented note like the one shown below.

```
21  // modified by Lila Leatherman
22  // date 1/25/2022
23  // Objective: map disturbance on the Dixie NF and fire near Brian Head, UT
24
```

2. Click the **Save** button and select **Save as** to save this script to your own repository. If the save button is disabled, make sure you have made changes by adding notes to save this in your own repository.



3. Type the path to your course folder and give the script a name like the one shown below.



4. Click **OK**

B. Edit the user input parameters

1. Change the **endYear** parameter to the integer value **2019** so that we get the two available post fire years.
2. Change the string value for the **startDay** parameter to **'07-10'** as we did in the UI app
3. Change the coordinates for the study area to refer to the Dixie NF point we have been exploring. Use the code below:

```
var aoi = ee.Geometry.Point(-112.73836, 37.75617);
```

4. Keep the index value set at 'NBR' as this was our chosen index.
5. Don't change the array values for the masking parameters.

C. Review the LandTrendr segmentation parameters

1. The variable **runParams** defines the object of parameter values to use for segmentation. We explored these values in the previous exercise using the Pixel Time Series Plotter and found these defaults to be suitable, so no need to change anything here. As was noted in Exercise 3, the Pixel Time Series Plotter is an acceptable but not quantitatively-robust way to select parameters for the model.

D. Review the change mapping parameters for the disturbance map.

1. The **changeParams** variable is an object with the parameters for extracting a change map from the LT-GEE outputs.
2. This object contains nested objects which is a very common pattern in JavaScript.

3. The initial values in here are the same as those we explored with the UI Change Mapper App.

E. Update the year and magnitude parameter objects

1. In the **changeParams** object, locate the **year** key and change the **start** and **end** values in the nested object to match the years we explored previously **1985** to **2019**.
2. In the **changeParams** object, locate the **mag** key (magnitude) and change the **value** to **100**.
   i. This will ensure that we don't omit the slow, gradual change from the spruce beetle mortality.
   ii. Check your understanding: how does lowering the value for **mag** help us detect slower, more gradual change?
3. You can keep the remaining values at their defaults here, matching the values that we evaluated with the UI Change Mapper App.
4. The object should look like the code below:

```
// define change parameters
var changeParams = {
  delta: "loss",
  sort: "greatest",
  year: { checked: true, start: 1985, end: 2019 },
  mag: { checked: true, value: 100, operator: ">" },
  dur: { checked: true, value: 4, operator: "<" },
  preval: { checked: true, value: 300, operator: ">" },
  mmu: { checked: true, value: 11 },
};
```

Note: *In this set of change parameters, we are specifying that we want to export the disturbance representing the 'greatest' vegetation loss disturbance. We saw earlier with the UI app that in this area, this will generate a map showing primarily the 2017 Brian Head fire. If we instead wanted to create a map showing the extent of the spruce beetle mortality, we could change the value for* ***sort*** *to be either '****oldest****' or '****slowest****' as we explored in the last exercise. Refer to the documentation for the* ***getChangeMap*** *function for more info on the valid parameters you can provide here.*

F. Reduce the size of the exported disturbance map

1. Scroll down into the main script until you find where the **region** variable is declared.
   i. This should be on or close to line 99 depending on what additional comments you might have added.
2. The code here chains two **ee.Feature** methods to get a bounding box for exporting the final map.
3. The method **ee.Feature.buffer** function simply buffers the feature by the distance provided. In this case, no projection argument is provided so it is buffering our point by 100,000 meters or 100 km.

G. Reduce the size of the exported map to increase speed

1. Change this value to 10,000 so that it only buffers by 10 km.

2. The resulting code should look like the line below:

```
var region = aoi.buffer(10000).bounds();
```

H. Update export parameters

1. Especially as you adapt scripts like this for your own use, you will likely need to change file names, adjust the CRS to match your other inputs, or adjust other parameters.
2. In this case, we need to replace the export command with a function to export to a Google Cloud Project, rather than Google Drive. If you are unfamiliar with Google Cloud Projects and do not yet have a Bucket set up, review and complete Intro to Geospatial Scripting in Javascript Exercise 1, Part 5.
3. Copy the code below to replace the section that begins with "export change data to google drive"

```
// export change data to cloud project
var region = aoi.buffer(10000).bounds();
var exportImg = changeImg.clip(region).unmask(0).short();

Export.image.toCloudStorage({
  image: exportImg,
  description: 'lt-gee_disturbance_map',
  bucket: 'lleatherman-gtactrainingstudents',
  fileNamePrefix: 'lt-gee_disturbance_map',
  region: region,
  scale: 30,
  crs: 'EPSG:5070',
  maxPixels: 1e13
});
```

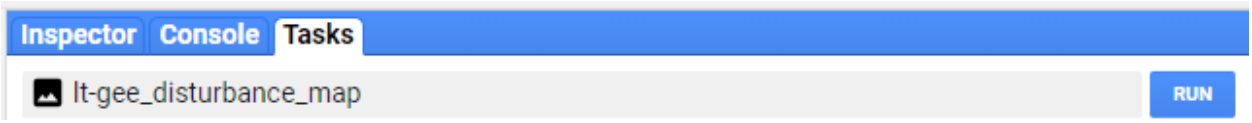
4. Review the code and update the **bucket** parameter to your appropriate bucket. Make sure you are working in the correct Cloud Project. The GTAC-Training-Students cloud project is a great place to export this image.

I. Save and run your script!

1. Click **Save** to save your script.
2. Click **Run** to and wait while processing takes place.
3. You can compare your script to the one found within the course repository.

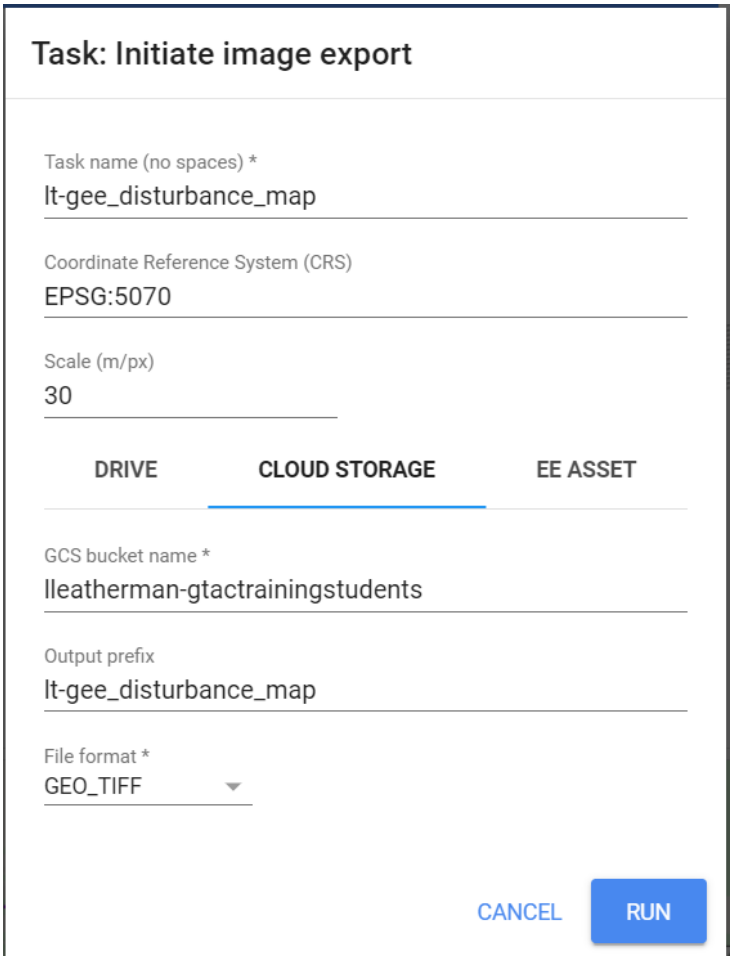
J. Export your disturbance map.

1. As soon as you see the **Tasks** tab turn gold in the upper right-hand panel, you are ready to export.
2. Click on the **Tasks** tab and you should see the export task (below)



3. Click the **RUN** button

4. The **Task: Initiate image export** dialogue will appear.
5. Feel free to accept the defaults-- or change the task name or filename. You can also adjust these parameters in the script itself!



6. When you are ready, click the **Run** button to initiate the export.
7. When the export job has completed, it will look like the graphic below.

8. Navigate to you're the Google Cloud Project Browser and locate your bucket.
   i. Click on your bucket to open it, and check for your exported disturbance map.
   ii. Download the map. Now, you can inspect or perform additional analyses on it in your preferred desktop GIS software.
   iii. To review how to export images from Google Cloud Project, review Intro to Geospatial Scripting in Javascript Exercise 3, Part 3.

Congratulations! You have successfully completed this exercise and gone through the complete process of running the LandTrendr algorithm in Earth Engine. You learned how to follow documentation for using a complex JavaScript library, use example code and adapt scripts for your own analyses.

Part 4: Appendix

A. Answers to questions in Part 2C

1. Where and how are the special LT-GEE functions being imported?
    i. Line 63

```
var ltgee = require('users/emaprlab/public:Modules/LandTrendr.js');
```

2. Can you differentiate the LT-GEE functions from Earth Engine functions?
    i. The LT-GEE functions begin with “lt.gee”. E.g., line 69

```
var lt = ltgee.runLT(startYear, endYear, startDay, endDay, aoi, index, [], runPar
ams, maskThese);
```

3. Where is the LandTrendr segmentation process being called?
    i. Line 69

```
var lt = ltgee.runLT(startYear, endYear, startDay, endDay, aoi, index, [], runPar
ams, maskThese);
```

4. Where is the disturbance map getting created?
    i. Line 72

```
var changeImg = ltgee.getChangeMap(lt, changeParams);
```

5. What kind of JavaScript objects are the following: **changeImg**, **pallete**, **magVizParms** and **region**? If you don’t know the answer, how could you use the code editor to find out?
    i. changeImg: Image
    ii. pallete : list
    iii. magVizParms : dictionary
    iv. region : polygon
    v. You can use the code editor to write a “print” command for each object, and then inspect the results in the Console. E.g.,

```
print(region)
```