[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

PROCESSING **IMAGE COLLECTION** [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

An image collection is is an EE variable object that represents a set of images that usually all depict spatial variation the same geographical characteristic but do so:

- for the same place at different times,

- for different places at the same time; or

- for different places at different times.

Image collections can be processed by using operations of the types listed below, which vary according to the nature of that processing. Each operation name is linked to a separate page describing that operation.

**ACCESSING** IMAGE COLLECTIONS[ee.ImageCollection](#ImageCollection) [ee.ImageCollection.load](#load)

**EDITING** IMAGE COLLECTIONS

BY **LIMITING** IMAGES [imageCollection.limit](#limit)

BY **FILTERING** IMAGES [imageCollection.filterMetadata](#filterMetadata) [imageCollection.filterDate](#filterDate)

[imageCollection.filterBounds](#filterBounds) [imageCollection.filter](#filter)

BY **SELECTING** BANDS [imageCollection.select](#select) [imageCollection.distinct](#distinct)

BY **COMBINING** BANDS [imageCollection.combine](#combine)

BY **JOINING** [ee.Join.apply](#apply)

BY **RECASTING** DATA TYPES [imageCollection.uint8](#uint8) [imageCollection.Uint8](#uint8)

[imageCollection.uint16](#uint8) [imageCollection.Uint16](#uint8)

[imageCollection.uint32](#uint8) [imageCollection.Uint32](#uint8)

[imageCollection.int8](#uint8) [imageCollection.toInt8](#uint8)

[imageCollection.byte](#uint8) [imageCollection.toByte](#uint8)

[imageCollection.int16](#uint8) [imageCollection.toInt16](#uint8)

[imageCollection.short](#uint8) [imageCollection.toShort](#uint8)

[imageCollection.int32](#uint8) [imageCollection.toInt32](#uint8)

[imageCollection.int](#uint8)  [imageCollection.toInt](#uint8)

[imageCollection.int64](#uint8) [imageCollection.toInt64](#uint8)

[imageCollection.long](#uint8) [imageCollection.toLong](#uint8)

[imageCollection.float](#uint8) [imageCollection.toFloat](#uint8)

[imageCollection.double](#uint8) [imageCollection.toDouble](#uint8)

[imageCollection.cast](#cast)

BY **RESETTING** VALUES [imageCollection.set](#set) [imageCollection.setMulti](#setMulti)

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

PROCESSING **IMAGE COLLECTION** [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

**TRANSFORMING** IMAGE COLLECTIONS ( INTO IMAGES )

BY **MOSAICKING** [imageCollection.mosaic](#mosaic)

WITH **LOGICAL** OPERATIONS [imageCollection.and](#and_or) [imageCollection.or](#and_or)

WITH **MATHEMATICAL** OPERATIONS [imageCollection.sum](#sum_) [imageCollection.product](#sum_)

[imageCollection.max](#sum_) [imageCollection.min](#sum_)

[imageCollection.mean](#sum_) [imageCollection.mode](#sum_)

[imageCollection.median](#sum_) [imageCollection.count](#sum_)

[imageCollection.formaTrend](#formaTrend)

WITH **REDUCERS** [imageCollection.reduce](#reduce)

**REPRODUCING** IMAGE COLLECTIONS

AS LISTS OF **PIXEL DATA** [imageCollection.getRegion](#getRegion)

AS LISTS OF **IMAGES** [imageCollection.first](#first) [imageCollection.toList](#toList)

AS GOOGLEMAP **OVERLAYS** [imageCollection.getMap](#getMap)

AS **IMAGES OF ARRAYS** [imageCollection.toArray](#toArray) [imageCollection.toArrayPerBand](#toArrayPerBand)

**COMPARING** IMAGE COLLECTIONS [ee.Algorithms.IsEqual(imageCollection)](#IsEqual)

**PARALLEL PROCESSING** IMAGE COLLECTIONS [imageCollection.map](#map)

**DOCUMENTING** IMAGE COLLECTIONS [featureCollection.getInfo](#Describe_getInfo) [ee.Algorithms.Describe(imageCollection)](#Describe_getInfo)

[imageCollection.toString](#toString_serialize) [imageCollection.serialize](#toString_serialize)

[ee.data.getList](#getList)

**PRESENTING** IMAGE COLLECTIONS

IN **PRINT** [print(imageCollection)](#print_console) [console.log(imageCollection)](#print_console)

[alert(imageCollection)](#alert_confirm) [confirm(imageCollection)](#alert_confirm)

IN **MAPS** [Map.addLayer(imageCollection)](#addLayer)

IN **CHARTS** [Chart.image.series](#series) [Chart.image.seriesByRegion](#seriesByRegion)

[Chart.image.doySeries](#doySeries) [Chart.image.doySeriesByReg…](#doySeriesByRegion) [Chart.image.doySeriesByYear](#doySeriesByYear)

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**ACCESSING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

ee.ImageCollection creates a new image collection from an image collection asset identified by a specified asset ID.

newImageCollection = ee.ImageCollection ( assetID)

The specified asset ID , given as a string

The new image collection

var NewIMAGES = ee.ImageCollection( 'NOAA/DMSP-OLS/NIGHTTIME\_LIGHTS' );

Map.setCenter( 126, 39, 6 ); // Korea

Map.addLayer( NewIMAGES, {min:0, max:100, opacity:0.7} );

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**ACCESSING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

ee.ImageCollection.load creates a new image collection from an image collection asset identified by a specified asset ID and (optional) version.

newImageCollection = ee.ImageCollection.load ( assetID, *version* )

The version number. Default: -1 (calling for the current version

The specified asset ID, given as a string

The new image collection

var NewIMAGES = ee.ImageCollection.load( 'NOAA/DMSP-OLS/NIGHTTIME\_LIGHTS', -1 );

Map.setCenter( 126, 39, 6 ); // Korea

Map.addLayer( NewIMAGES, {min:0, max:100, opacity:0.7} );

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **LIMITING IMAGES**

imageCollection.limit creates new image collection that contains only the first of a specified number of elements from a specified image collection

after (optionally) sorting those elements according to the ascending or descending order of their values for a specified property.

newImageCollection = oldImageCollection.limit ( howMany*, sortProperty, ascendingOrder?* )

The specified

image collection

The specified number of

elements, given as an integer

The specified property,

given as a string

The specified order, given as a Boolean set to

True for ascending or False for descending

The new image

collection

var OldIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04' );

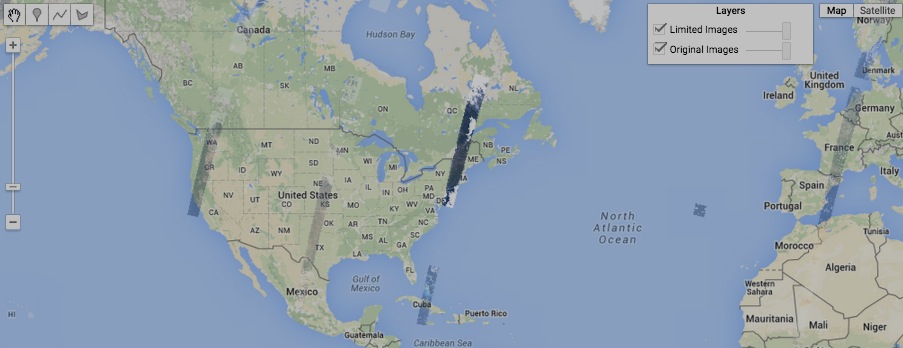
var NewIMAGES = OldIMAGES.limit( 15 );

print( OldIMAGES, NewIMAGES );

Map.setCenter( -73.015, 41.228, 3 );

Map.addLayer( OldIMAGES, {bands:'B4,B3,B2', min:0, max:0.2, opacity:0.4}, 'Original Images' );

Map.addLayer( NewIMAGES, {bands:'B4,B3,B2', min:0, max:0.4, opacity:1.0}, 'Limited Images' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **FILTERING IMAGES**

imageCollection.filterMetadata creates new image collection that contains only those images from a specified image collection FIX

whose values for a specified property bear a specified relationship to a specified value.

newImageCollection = oldImageCollection.filterMetadata ( property, relationship, value )

The specified value,

given as a string or a number

The name of the specified

property, given as a string

The specified image collection

The new image collection

The specified relationship, given as **"equals" "less\_than" "greater\_than" "starts\_with" "ends\_with" "contains"**

one of the following strings. **"not\_equals" "not\_less\_than" "not\_greater\_than" "not\_starts\_with" "not\_ends\_with" "not\_contains"**

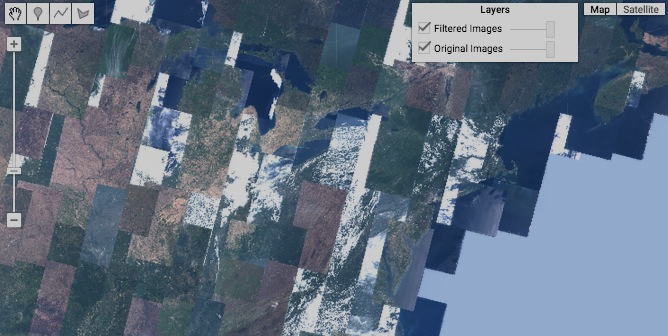
var OldIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ); ADD PRINT

var NewIMAGES = OldIMAGES.filterMetadata('CLOUD\_COVER','less\_than',0.1);

Map.setCenter( -73.015, 41.228, 5);

Map.addLayer( OldIMAGES, {bands:'B4,B3,B2', min:0, max:0.2, opacity:1.0}, 'Original Images' );

Map.addLayer( NewIMAGES, {bands:'B4,B3,B2', min:0, max:0.2, opacity:1.0}, 'Filtered Images' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **FILTERING IMAGES**

imageCollection.filterDate creates new image collection that contains only those images from a specified image collection

that are associated with the time period between specified starting and stopping dates.

newImageCollection = oldImageCollection.filterDate ( startingDate, stoppingDate )

The specified starting and stopping dates, given either as a string in 'month#/day#/year#'

(or 'month#-day#-year#') format or the number of milliseconds since January 1, 1970

The specified image collection

The new image collection

var FilteredIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-5' );

Map.setCenter( -96.15, 44.15, 4);

Map.addLayer( FilteredIMAGES, {bands:'B4,B3,B2', min:0, max:0.2}, 'Filtered Images' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **FILTERING IMAGES**

imageCollection.filterBounds creates new image collection that contains only those images from a specified image collection

that intersect a specified geometry, feature, or feature collection.

newImageCollection = oldImageCollection.filterBounds( oldGeometry )

The specified image collection

The specified geometry

The new image collection

var OldIMAGES = ee.ImageCollection('LANDSAT/LE7').filterDate('2000-04-01','2000-04-15');

var AllStateFEATURES = ee.FeatureCollection( 'ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8' );

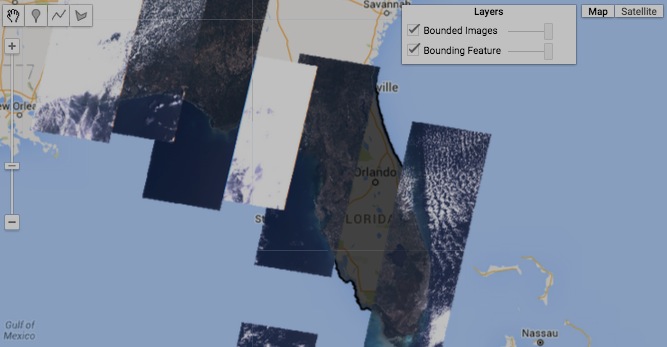
var OneStateFEATURE = AllStateFEATURES.filter( ee.Filter.eq('Name','Florida') );

var FilteredIMAGES = OldIMAGES.filterBounds( OneStateFEATURE );

Map.centerObject( OneStateFEATURE, 6);

Map.addLayer( OneStateFEATURE, {color: '000000'}, 'Bounding Feature' );

Map.addLayer( FilteredIMAGES, {bands: ['B3','B2','B1'], min:0, max:255 }, 'Bounded Images ' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **FILTERING IMAGES**

imageCollection.filter creates new image collection that contains only those images from a specified image collection

that satisfy the conditions of a specified filter.

newImageCollection = oldImageCollection.filter ( oldFilter )

The specified filter

The specified image collection

The new image collection

var OldIMAGES = ee.ImageCollection('LANDSAT/LE7').filterDate('2000-04-01','2000-04-15');

var BoundingFEATURES = ee.FeatureCollection ('ft:1dXsoWNDPeUEXzlQV52jaI4zeckuyIlHiUwox3K4');

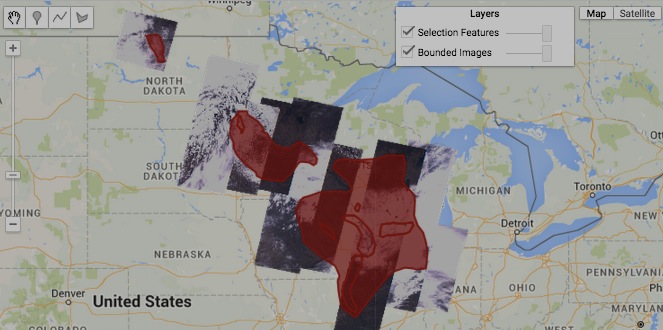
var TheFILTER = ee.Filter.bounds( BoundingFEATURES );

var FilteredIMAGES = OldIMAGES.filter( TheFILTER );

Map.centerObject (FilteredIMAGES, 5 );

Map.addLayer( FilteredIMAGES, {bands: ['B3','B2','B1'], min:0, max:300 }, 'Bounded Images' );

Map.addLayer( BoundingFEATURES, {color:'990000'}, 'Selection Features' );

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **SELECTING BANDS**

imageCollection.select creates a new image collection containing only a specified set of bands from the images in a specified image collection

newImageCollection = oldImageCollection.select( oldBands, *newBandNames* )

The specified

image collection

The specified names, indices, or regexes, given as an array of strings

An array of new names to be ascribed to (all of) the

bands of the new image, given as an array of strings.

The new image

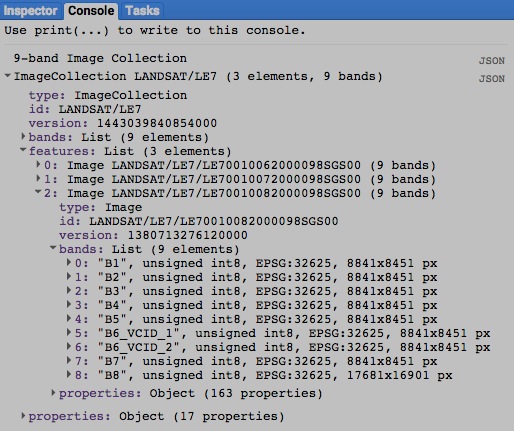
collection

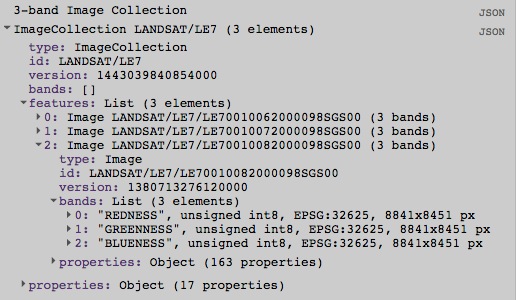
var OldIMAGES = ee.ImageCollection('LANDSAT/LE7').filterDate('2000-04-01','2000-04-15').limit(3);

var NewIMAGES = OldIMAGES.select(['B3','B2','B1'], ['REDNESS','GREENNESS','BLUENESS']);

print( '9-band Image Collection', OldIMAGES );

print( '3-band Image Collection', NewIMAGES );





[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **SELECTING BANDS**

imageCollection.distinct creates a new image collection by replicating a specified image collection after removing any image

whose value(s) for a specified set of one or more properties match those of an image already considered.

newImageCollection = oldImageCollection.distinct ( properties )

The specified properties, given as a selector set

The new image collection

The specified image collection

var OldIMAGES = xxxxx

var NewIMAGES = NewIMAGES.distinct( 'which property' );

print( OldIMAGES.getInfo( ), NewIMAGES.getInfo( ) );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **COMBINING BANDS**

imageCollection.combine creates new image collection by replicating a specified image collection and adding whichever of the bands

from a second specified image collection (with the same footprint and metadata) have a matching band names.

newImageCollection = 1stOldImageCollection.combine ( 2ndOldImageCollection, *overwrite?* )

The second specified image collection

The first specified image collection

A Boolean set to True (only) if bands of the same name are to be overwritten. Default: False

The new image collection

var TheseIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04' ).limit(15).select(3,2,1);

var ThoseIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04' ).limit(15).select(6,5,4);

var CombinedIMAGES = TheseIMAGES.combine( ThoseIMAGES );

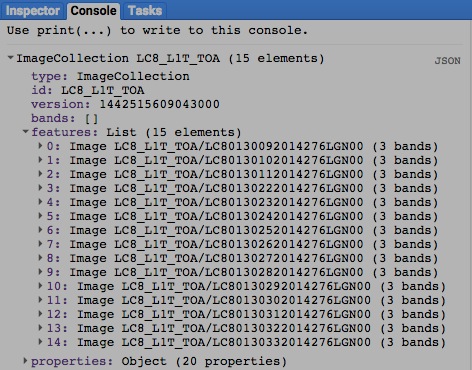
print( TheseIMAGES.getInfo(), ThoseIMAGES.getInfo(), CombinedIMAGES.getInfo() );

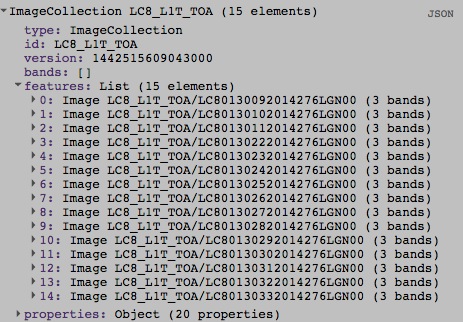
Map.setCenter( -73.015, 41.228, 5);

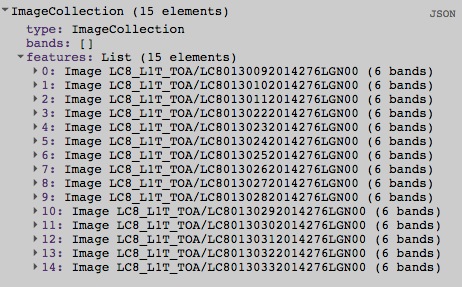
Map.addLayer( TheseIMAGES, {bands:'B4,B3,B2', min:0, max:0.2, opacity:0.7, gamma:0.5}, 'Earlier Images' );

Map.addLayer( ThoseIMAGES, {bands:'B7,B6,B5', min:0, max:0.2, opacity:0.7, gamma:0.5}, 'Later Images' );

Map.addLayer( CombinedIMAGES, {bands:'B7,B5,B2', min:0, max:0.2, opacity:0.7, gamma:0.5}, 'Combined Images' );

****

****

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **JOINING**

ee.Join.apply creates a new image collection by using a specified join to augment a specified “primary” (image or feature) collection

with whatever elements of a specified “secondary” (image or feature) collection satisfy a specified condition.

newImageCollection = oldJoin.apply( primaryCollection, secondaryCollection, condition )

The specified condition,

given as a filter

The specified

secondary collection

The specified

primary collection

The specified join

The new image collection

var AprilJuneIMAGES = ee.ImageCollection('LANDSAT/LC8\_L1T\_TOA').filterDate('2014-04-01', '2014-06-01')

.filterBounds(ee.Geometry.Point(-122.092406, 37.424149));

var MayJulyIMAGES = ee.ImageCollection('LANDSAT/LC8\_L1T\_TOA').filterDate('2014-05-01', '2014-07-01')

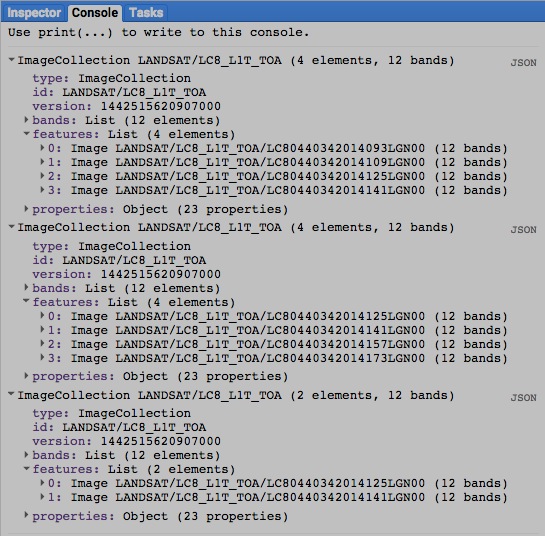
.filterBounds(ee.Geometry.Point(-122.092406, 37.424149));

var TheFILTER = ee.Filter.equals( {leftField: 'system:index', rightField: 'system:index'} );

var TheJOIN = ee.Join.simple( );

var MayJuneIMAGES = TheJOIN.apply( AprilJuneIMAGES, MayJulyIMAGES, TheFILTER );

print( AprilJuneIMAGES, MayJulyIMAGES, MayJuneIMAGES );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **RECASTING VALUES**

imageCollection.uint8 , .toUint8 , .byte , and .toByte unsigned 8-bit integers

Each of these operations replicates a specified image collection (of any pixel type) to create a new one whose pixel type is as indicated in red.

.uint16 and .toUint16 unsigned 16-bit integers

.uint32 and .toUint32 unsigned 32-bit integers

.int8 and .toInt8 signed 8-bit integers

.int16 , .toInt16 , .short , and .toShort signed 16-bit integers

.int32 , .toInt32 , .int , and .toInt signed 32-bit integers

.int64 , .toInt64 , .long , and .toLong signed 64-bit integers

.float and .toFloat 32-bit floating-point numbers

.double and .toDouble 64-bit floating-point numbers

newImageCollection = oldImageCollection.uint8( ) or .toUint8( ) or .byte( ) or .toByte( )

or .uint16( ) or .toUint16( )

or .uint32( ) or .toUint32( )

The new image collection

or .int8( ) or .toInt8( )

or .int16( ) or .toInt8( ) or .short( ) or .toShort( )

The specified image collection

or .int32( ) or .toInt32( ) or .int( ) or .toInt( )

or .int64( ) or .toInt64( ) or .long( ) or .toLong( )

or .float( ) or .toFloat( )

or .double( ) or .toDouble( )

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **RECASTING VALUES**

imageCollection.cast creates a new image collection of specified band-by-band types by replicating a specified image collection of any type.

newImageCollection = oldImageCollection.cast( bandTypes, *bandOrder* )

The specified types, given as a dictionary of band names and corresponding types. Types can be specified as PixelType objects or as any of the following strings: 'int8', 'int16', 'int32', 'int64', 'uint8', 'uint16', 'uint32', 'byte', 'short', 'int', 'long', 'float', or 'double.'

A list of (all) band names indicating the order in which

they are to be stored. Default: alphabetical order

The specified

image collection

The new

image

collection

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **RESETTING VALUES**

imageCollection.set creates new image collection by replicating a specified image collection

after setting or resetting specified properties to specified values.

newImageCollection = oldImageCollection.set ( pairsOfPropertiesAndValues )

The specified image collection

The new image collection

The specified properties and new values, given as a comma-separated sequence

(or a dictionary) of property name strings, each immediately followed by its new value.

var FirstIMAGE = ee.Image( 'LC8\_L1T/LC80150332014322LGN00' );

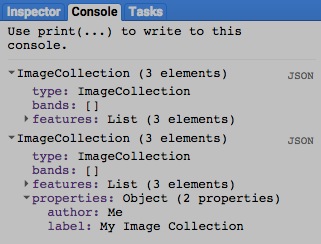
var SecondIMAGE = ee.Image( 'LC8\_L1T/LC80140322014219LGN00' );

var ThirdIMAGE = ee.Image( 'LC8\_L1T/LC80130312014276LGN00' );

var OldIMAGES = ee.ImageCollection( [ FirstIMAGE, SecondIMAGE, ThirdIMAGE ] );

var NewIMAGES = OldIMAGES.set( 'author','Me','label','My Image Collection' );

print( OldIMAGES.getInfo( ), NewIMAGES.getInfo( ) );

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**EDITING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) BY **RESETTING VALUES**

imageCollection.setMulti creates new image collection by replicating a specified image collection

after setting or resetting specified properties to specified values.

newImageCollection = oldImageCollection.setMulti ( dictionaryOfPropertiesAndValues )

The specified properties and new values, given as a

dictionary of property name strings and new values.

The specified image collection

The new image collection

var FirstIMAGE = ee.Image( 'LC8\_L1T/LC80150332014322LGN00' );

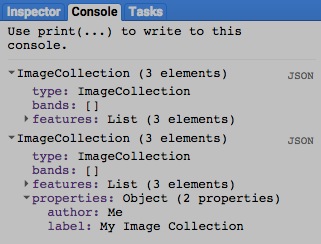
var SecondIMAGE = ee.Image( 'LC8\_L1T/LC80140322014219LGN00' );

var ThirdIMAGE = ee.Image( 'LC8\_L1T/LC80130312014276LGN00' );

var OldIMAGES = ee.ImageCollection( [ FirstIMAGE, SecondIMAGE, ThirdIMAGE ] );

var NewIMAGES = OldIMAGES.setMulti( {'author':'Me','label':'My Image Collection'} );

print( OldIMAGES.getInfo( ), NewIMAGES.getInfo( ) );

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) INTO IMAGES BY **MOSAICKING**

imageCollection.mosaic creates new image in by combining images from a specified image collection such that each pixel is set to

the value last ascribed to it by any of those images according to the order in which they are sorted.

newImage = oldImageCollection.mosaic ( )

The specified image collection

The new image collection

var FirstIMAGE = ee.Image( 'LE7\_L1T/LE70400351999208EDC00' );

var SecondIMAGE = ee.Image( 'LE7\_L1T/LE70400351999224EDC00' );

var ThirdIMAGE = ee.Image( 'LE7\_L1T/LE70400351999240EDC00' );

var FourthIMAGE = ee.Image( 'LE7\_L1T/LE70390351999185EDC00' );

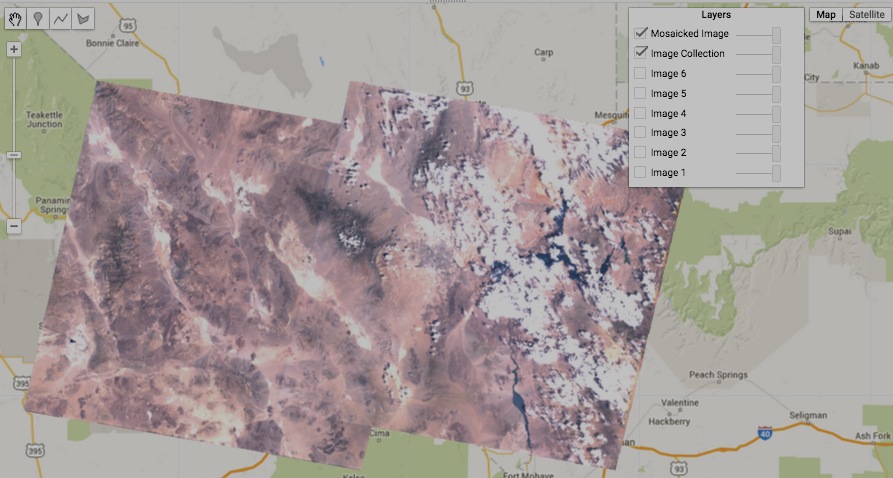
var FifthIMAGE = ee.Image( 'LE7\_L1T/LE70390351999201EDC00' );

var SixthIMAGE = ee.Image( 'LE7\_L1T/LE70390351999217EDC02' );

var TheIMAGES = ee.ImageCollection( [FirstIMAGE,SecondIMAGE,ThirdIMAGE,FourthIMAGE,FifthIMAGE,SixthIMAGE] );

var MosaickedIMAGE = TheIMAGES.mosaic();

print( TheIMAGES, MosaickedIMAGE );

Map.setCenter( -115.5267, 36.1334 , 8 );

Map.addLayer( FirstIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 1' );

Map.addLayer( SecondIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 2' );

Map.addLayer( ThirdIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 3' );

Map.addLayer( FourthIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 4' );

Map.addLayer( FifthIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 5' );

Map.addLayer( SixthIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Image 6' );

Map.addLayer( TheIMAGES,

{bands:['B3,B2,B1'], gain:1.5}, 'Image Collection' );

Map.addLayer( MosaickedIMAGE,

{bands:['B3,B2,B1'], gain:1.5}, 'Mosaicked Image' );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) INTO IMAGES WITH **LOGICAL** OPERATIONS

imageCollection.and and .or create new images in which each pixel is set to a Boolean value of true (1) if any (in the case of **or**) or all (in the case of **and**)

of its values in a specified image collection are true (non-zero). If not, the pixel is set to false (0).

newImage = oldImageCollection.and( ) or .or( )

The specified image collection

The new image

var NightLightIMAGES = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME\_LIGHTS').select('stable\_lights');

var SometimesLightIMAGE = NightLightIMAGES.or();

var AlwaysLightIMAGE = NightLightIMAGES.and();

print( NightLightIMAGES );

Map.setCenter(-90.703, 28.575, 6);

Map.addLayer( SometimesLightIMAGE, {min: 0, max: 1, palette: ['000000','ffff99'], opacity:0.5}, 'Sometimes Lit');

Map.addLayer( AlwaysLightIMAGE, {min: 0, max: 1, palette: ['333333','ffff55'], opacity:0.5}, 'Always Lit');



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) INTO IMAGES WITH **MATHEMATICAL** OPERATIONS

imageCollection.sum , .product ,.max , .min , .mean ,

all create a new image in which the value of every pixel is calculated

as a mathematical function of each pixels values on corresponding

bands of the images in a specified image collection.

.mode , .median , and .count

newImage = oldImageCollection.sum( ) or .product( ) or .max( ) or .min( ) or .mean( )

or .mode( ) or .median( ) or .count( )

The specified mathematical function

The specified

image collection

The new image

var TheGEOMETRY = ee.Geometry.Point( -111.489, 37.23 );

var AllIMAGES = ee.ImageCollection('LE7\_L1T').filterDate('2000-04-01', '2000-06-01');

var SomeIMAGES = AllIMAGES.filter( ee.Filter.bounds( TheGEOMETRY ) );

var MedianIMAGE = SomeIMAGES.median( );

var FirstIMAGE = ee.Image( 'LE7\_L1T/LE70370342000094EDC00' ); DO THIS VIA SELECTION INSTEAD

var SecondIMAGE = ee.Image( 'LE7\_L1T/LE70370342000110EDC00' );

var ThirdIMAGE = ee.Image( 'LE7\_L1T/LE70370342000126EDC00' );

var FourthIMAGE = ee.Image( 'LE7\_L1T/LE70370342000142EDC00' );

print( SomeIMAGES, MedianIMAGE );

Map.setCenter( -111.8388, 37.7523, 11 );

Map.addLayer( FirstIMAGE, {gain: '1.4, 1.4, 1.1'}, 'First IMAGE' );

Map.addLayer( SecondIMAGE, {gain: '1.4, 1.4, 1.1'}, 'Second IMAGE' );

Map.addLayer( ThirdIMAGE, {gain: '1.4, 1.4, 1.1'}, 'Third IMAGE' );

Map.addLayer( FourthIMAGE, {gain: '1.4, 1.4, 1.1'}, 'Fourth IMAGE' );

Map.addLayer( MedianIMAGE, {gain: '1.4, 1.4, 1.1'}, 'Median IMAGE' );

## 

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) INTO IMAGES WITH **MATHEMATICAL** OPERATIONS

imageCollection.formaTrend creates a new image of four floating-point bands respectively indicating 1) the long-term trend, 2) the short-term trend,

3) the t-test of the long term trend against the time series, and 4) the Bruce Hansen test of parameter stability for a time series (or, optionally, the trends of the ratio of a time series and a covariate) respectively represented by two specified single-band image collections: one representing a time series, and the other representing a covariate series. The long-term trend is estimated from the linear term of a regression on the full time series and the short-term trend.

newImage = timeSeriesImageCollection.formaTrend( *covariateImageCollection, windowSize* )

The specified image collection representing

an evenly spaced time series

The specified image collection representing

a covariate series. Default: null

The new

image

The short term trend analysis window

size, given as a number of images

var

EXTRA example of imageCollection.mean

var NightLightIMAGES = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME\_LIGHTS').select('stable\_lights');

var MeanIMAGE = NightLightIMAGES.mean( );

print( NightLightIMAGES );

Map.setCenter(129.331, 34.38, 6);

Map.addLayer( MeanIMAGE, {min: 0, max: 63, palette: ['000066','ff0000','ffff00','ffffff'], opacity:0.7}, 'Mean');

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**TRANSFORMING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) INTO IMAGES WITH **REDUCERS**

imageCollection.reduce creates a new image by applying a specified reducer to set of the values associated with each pixel by the set of images in a

specified image collection. For collections of multi-band images, each set of corresponding input bands (one from each image

in the collection) is reduced to a single output band, and all output bands are stored as part of a multi-band image.

newImage = oldImageCollection.reduce( reducer*, parallelScale* )

A number that can be increased (from 1.0) to reduce the likelihood

of exceeding limits on available computer memory. Default: 1.0

The specified reducer. If this reducer calls for

multiple inputs, the number of inputs must match

the number of bands **oldImageCollection**.

The specified

image collection

The new image, whose band names will depend on the specified reducer.

If that reducer calls for a single input, each new band name will identify both its original band and its reducer (e.g. 'B2\_mean').

If the reducer calls for multiple inputs, however, each new band name will identify its reducer followed by a band index number (e.g. 'mean', 'mean1', 'mean2').

var FirstIMAGE = ee.Image( 'LANDSAT/LE7/LE70190352000272GNC02' ).select( ['B3','B2','B1'] );

var SecondIMAGE = ee.Image( 'LANDSAT/LE7/LE70190352000288EDC00' ).select( ['B3','B2','B1'] );

var TheIMAGES = ee.ImageCollection.fromImages( [FirstIMAGE,SecondIMAGE] );

var TheREDUCER = ee.Reducer.max( );

var ReducedIMAGE = TheIMAGES.reduce( TheREDUCER );

print( FirstIMAGE, SecondIMAGE, TheIMAGES );

Map.setCenter( -84.1941, 35.9988, 12 );

Map.addLayer( FirstIMAGE, {bands: ['B3', 'B2', 'B1'], min:0, max:256 }, 'First Image' );

Map.addLayer( SecondIMAGE, {bands: ['B3', 'B2', 'B1'], min:0, max:256 }, 'Second Image' );

Map.addLayer( ReducedIMAGE, {bands: ['B3\_max','B2\_max','B1\_max'], min:0, max:256 }, 'Reduced Image' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS LISTS OF **PIXEL DATA**

imageCollection.getRegion creates a new list of items describing the image, longitude, latitude, time, and band-by-band value(s) of each pixel

(per band per image) of a specified image collection that lies within a specified geometry

newList = oldImageCollection.getRegion ( geometry, scale, *CRS*, *placement* )

The specified image collection

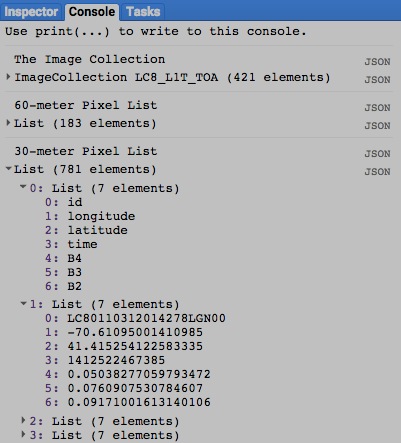
A list of six numbers that indicate how the specified image is to be situated with respect to the CRS. This cannot be used in conjunction with ***scale***. Default: Null

The specified geometry

The new list

A number which, when divided by the nominal size of a meter in the specified **CRS**, indicates the scale of the specified image. This cannot be used in conjunction with **placement** but one of those two must be specified. Default: Null

The cartographic reference system (CRS) to be used, given as an EPSG code or WKT. Default: the CRS of the specified image’s first band

var TheIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' )

.filterDate( '2014-10-05','2014-10-06' ).select( 3,2,1 );

var TheSITE = ee.Geometry.Polygon( [[-70.611,41.422],

[-70.611,41.415],[-70.603,41.415],[-70.603,41.422]] );

var CoarserLIST = TheIMAGES.getRegion( TheSITE, 60 );

var FinerLIST = TheIMAGES.getRegion( TheSITE, 30 );

Map.setCenter( -70.6069, 41.4184, 8 ); // The Vineyard

Map.addLayer( TheIMAGES, {bands:'B4,B3,B2', min:0, max:0.2,

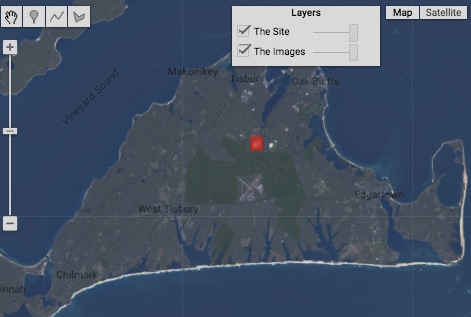
opacity:0.7, gamma:0.5}, 'The Images' );

Map.addLayer( TheSITE, {color:'ff0000'}, 'The Site' );

print( 'The Image Collection', TheIMAGES );

print( '60-meter Pixel List', CoarserLIST );

print( '30-meter Pixel List', FinerLIST );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS LISTS OF **PIXEL DATA**

imageCollection.first creates a new list element by reproducing the first image in a specified image collection.

newListElement = oldImageCollection.first ( )

The specified image collection

The new image

var AllIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-11-25','2014-11-26').select(0);

var FirstELEMENT = AllIMAGES.first( );

var FirstIMAGE = ee.Image( FirstELEMENT );

Map.centerObject( FirstIMAGE, 4 );

Map.addLayer( AllIMAGES, {palette:['000000','999999'], min:0, max:0.4 }, 'All Images' );

Map.addLayer( FirstIMAGE, {palette:['000000','ff0000'], min:0, max:0.4 }, 'The First Image' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS LISTS OF **IMAGES**

imageCollection.toList creates a new list of a specified number of consecutive images from a specified image collection.

newList = oldImageCollection.toList ( howManyImagesToList, *howManyImagesToSkip* )

The index of the first image to be listed. Default: 0

The number of consecutive images to be listed

The new list

The specified image collection

var ImageCOLLECTION = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-11-25','2014-11-26').select(0);

var ImageLIST = ImageCOLLECTION.toList( 5,0 );

var InitialIMAGE = ee.Image( ImageLIST.get(0) );

var NextIMAGE = ee.Image( ImageLIST.get(1) );

var FinalIMAGE = ee.Image( ImageLIST.get(2) );

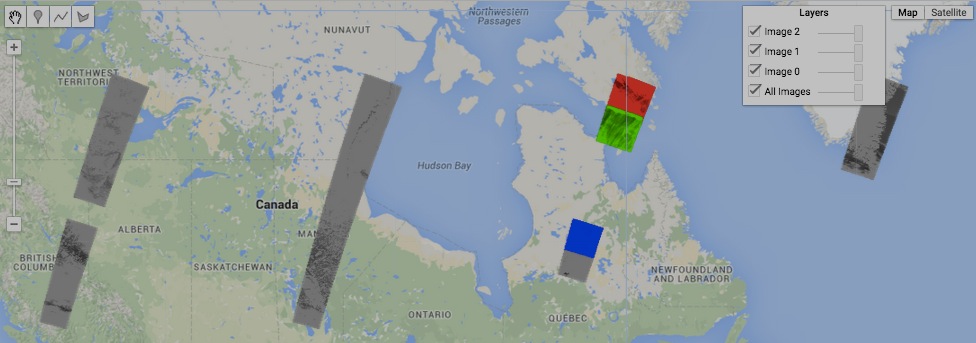
Map.centerObject( NextIMAGE, 4 );

Map.addLayer( ImageCOLLECTION, {palette:['000000','999999'], min:0, max:0.4 }, 'All Images' );

Map.addLayer( InitialIMAGE, {palette:['000000','ff0000'], min:0, max:0.4 }, 'Image 0' );

Map.addLayer( NextIMAGE, {palette:['000000','00ff00'], min:0, max:0.4 }, 'Image 1' );

Map.addLayer( FinalIMAGE, {palette:['000000','0000ff'], min:0, max:0.4 }, 'Image 2' );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS **GOOGLEMAP OVERLAYS**

imageCollection.getMap creates a new object containing a specified image collection, Map ID, and token from which to create a Google Maps overlay.

newObject = oldImageCollection.getMap ( settings)

Instructions identifying data-display options in the same manner as **Map.addlayer**

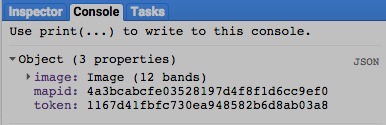
The specified image collection

The new object

var TheIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-05-01','2014-06-01');

var ThePACKAGE = TheIMAGES.getMap({ bands:'B4,B3,B2', min:0, max:0.2, opacity:0.7, gamma:0.5 });

print( ThePACKAGE );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS **IMAGES OF ARRAYS**

imageCollection.toArray creates a new image in which each pixel holds a two-dimensional array values drawn from a specified image collection.

Each array’s first dimension represents a series of images, while its second dimension represents a series of bands.

newImage = oldImageCollection.toArray ( )

The new image

The specified image

var TheIMAGES =

var ValueARRAY = TheIMAGE.toArray( 0 );

print( ValueARRAY );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**REPRODUCING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) AS **IMAGES OF ARRAYS**

imageCollection.toArrayPerBand creates a new image in which each pixel holds an two-dimensional array values drawn from

a specified image collection. Each array’s first dimension represents a series of images, while its second dimension represents a series of bands.

newImage = oldImageCollection.toArrayPerBand ( axis )

The specified axis, given as an integer. Default: 0

The specified image collection

The new image

var TheIMAGES =

var ValueARRAY = TheIMAGE.toArray( 0 );

print( ValueARRAY );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**COMPARING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

ee.Algorithms.IsEqual creates a new Boolean set to True (only) if the first of two specified image collections

is identical to the second in both structure and content.

newBoolean = ee.Algorithms.IsEqual ( 1stImageCollection, 2ndImageCollection )

The first specified image collection

The second specified image collection

The new Boolean

var The1stIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04' ).limit(15).select(3,2,1);

var The2ndIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04' ).limit(15).select(6,5,4);

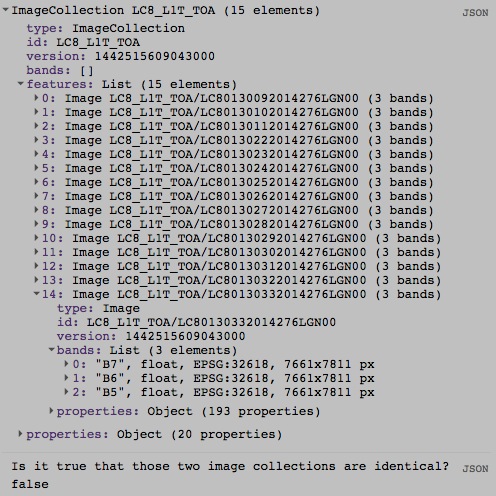
var TheBOOLEAN = ee.Algorithms.IsEqual( The1stIMAGES, The2ndIMAGES );

print( The1stIMAGES );

print( The2ndIMAGES );

print( 'Is it true that those two image collections are identical?', TheBOOLEAN );





[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PARALLEL PROCESSING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

imageCollection.map creates a image feature collection by applying a specified algorithm to each of the images in a specified image collection.

(Because this algorithm is called only once, it should not include imperative functions like **print** or rely on external states.)

newImageCollection = oldImageCollection.map ( algorithm)

The specified image collection

The new image collection

The specified algorithm

function TheALGORITHM( TypicalIMAGE )

{ var NewBAND = TypicalIMAGE.normalizedDifference( ['B5', 'B4'] );

var NewIMAGE = TypicalIMAGE.addBands( NewBAND );

return NewIMAGE;

}

var OldIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-03','2014-10-04').limit(15);

var NewIMAGES = OldIMAGES.map( TheALGORITHM );

print('Old Image Collection', OldIMAGES);

print('New Image Collection', NewIMAGES);

Map.setCenter( -73.3337, 40.7785, 10 );

Map.addLayer( NewIMAGES, {bands:['nd'], min:0, max:1, palette:'000000,11ff11',opacity:0.7}, 'NDVI');

****

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**DOCUMENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

ee.Algorithms.Describe and imageCollection.getInfo

each creates a JSON-compatible text object

representing a specified image collection.

newObject = ee.Algorithms.Describe( oldImageCollection )

and oldImageCollection.getInfo( )

The new object

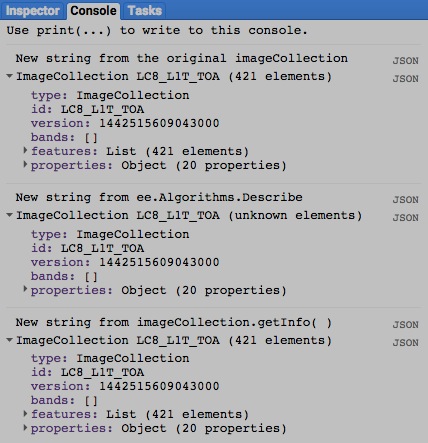
The specified image collection

var OldIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-05','2014-10-06').select(3,2,1);

print( 'New string from the original imageCollection', OldIMAGES );

print( 'New string from ee.Algorithms.Describe', ee.Algorithms.Describe( OldIMAGES ) );

print( 'New string from imageCollection.getInfo( )', OldIMAGES.getInfo( ) );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**DOCUMENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

imageCollection.toString and .serialize each creates a new string presenting information on a specified image collection.

newString = oldImageCollection.toString ( )

and oldImageCollection.serialize( )

The specified image collection

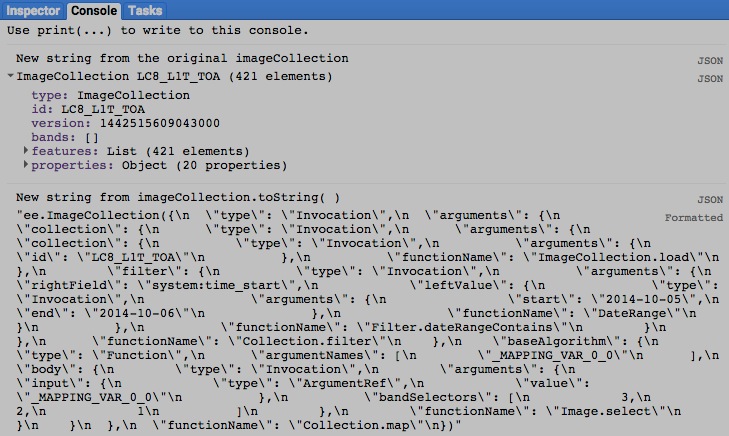
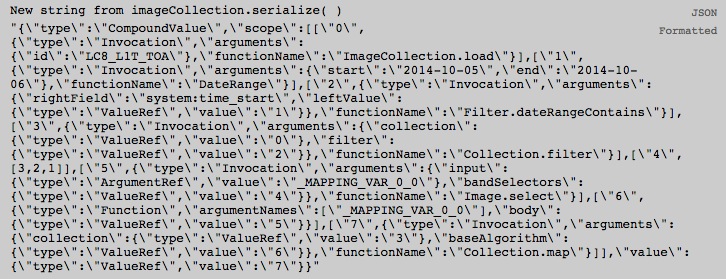
The new string

var OldIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-10-05','2014-10-06').select(3,2,1);

print( 'New string from the original imageCollection', OldIMAGES );

print( 'New string from imageCollection.toString( )', OldIMAGES.toString( ) );

print( 'New string from imageCollection.serialize( )', OldIMAGES.serialize( ) );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**DOCUMENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx)

ee.data.getList creates a new list of image types and asset IDs for each of a specified set of images in aspecified image collection.

newList = ee.data.getList ( *settings* )

Instructions indicating what to list, given as a curly bracketed and comma-separated sequence of the following **key: value** pairs.

**id : X** gives **X** as the quoted asset ID of the specified image collection.

**fields : X** gives **X** as a string of comma-separated substrings naming the fields to be included.

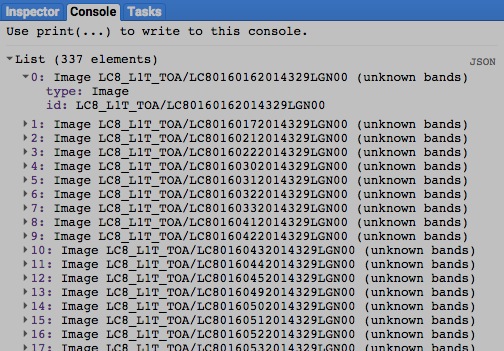
**starttime : X** gives **X** as the earliest date to be considered, given in milliseconds since the epoch, which can be calculated [here](http://www.epochconverter.com)..

**endtime : X** gives **X** as the latest date to be considered, given in milliseconds since the epoch, which can be calculated [here](http://www.epochconverter.com)..

The new list

var TheLIST = ee.data.getList({id:'LC8\_L1T\_TOA', starttime:1416873600000, endtime:1416960000000 } );

print( TheLIST );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **PRINT**

print ( imageCollection ) and console.log ( imageCollection ) present JSON-formatted text renditions of a specified

image collection in the console.

print( oldImageCollection ) or console.log( oldImageCollection )

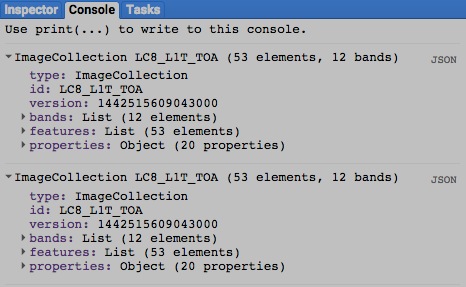
The specified image collection

var TheFEATURES = ee.FeatureCollection( 'ft:1dXsoWNDPeUEXzlQV52jaI4zeckuyIlHiUwox3K4' );

var TheIMAGES = ee.ImageCollection('LC8\_L1T\_TOA').filterDate('2014-05-01','2014-06-01').filterBounds(TheFEATURES);

print( TheIMAGES );

console.log( TheIMAGES );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **PRINT**

alert ( imageCollection ) and confirm( imageCollection ) presents the text representation of a specified string in a pop-up message box.

alert( oldImageCollection ) or confirm( oldImageCollection )

The specified image collection

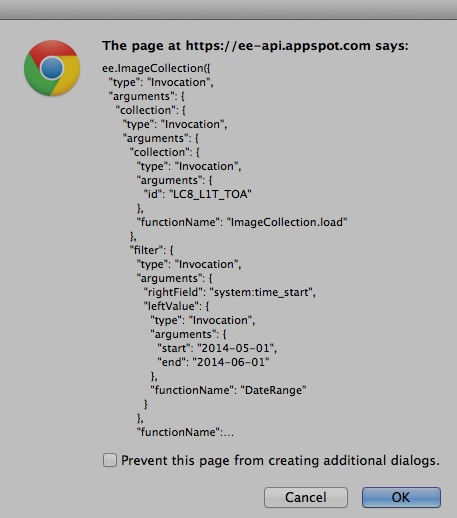
var TheFEATURES = ee.FeatureCollection( 'ft:1dXsoWNDPeUEXzlQV52jaI4zeckuyIlHiUwox3K4' );

var TheIMAGES = ee.ImageCollection('LC8\_L1T\_TOA').filterDate('2014-05-01','2014-06-01').filterBounds(TheFEATURES);

alert( TheIMAGES );

confirm( TheIMAGES );





[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **MAPS**

Map.addLayer ( imageCollection ) adds a specified image collection to the map as a layer with specified graphic settings.

Map.addLayer ( imageCollection*,  settings, name, visibility* )

Instructions indicating how to display the specified image collection, given as a curly bracketed and comma-separated sequence of the following **key: value** pairs.

**format: 'x'** gives **'x'**as one of the strings **'jpg'** or **'png'** to specify the digital format of the layer to be displayed.

**version: #** gives **#** as a number specifying the version of the image collection to be displayed. For the current version, give **#** as -1.

**bands: rgb** gives **rgb** as a list of the form **['r','g','b']**  (or as a concatenation of the form **'r, g, b'**) containing three strings that respectively

specify the names of the bands whose values are to be represented by the brightness of red (**r**), green (**g**), and blue (**b**) color components.

**gain: #, bias: ##** gives **#** as a number specifying an amount by which each value is to be multiplied and/or **##** as a number specifying an amount to be

added to each value before being displayed. **#** and **##** may be given as numbers (referring to all bands) or as lists of numbers (referring

to consecutive bands). **gain** and **bias** settingscannot be used in conjunction with **min** and **max** settings.

(Additional settings are described on the following page.)

A Boolean set to TRUE (only) if the layer being displayed should initially be toggled on (visible) rather than off (invisible).Visibility can also be adjusted interactively by using the map’s layer listing.

The specified image collection.

This is automatically and temporarily

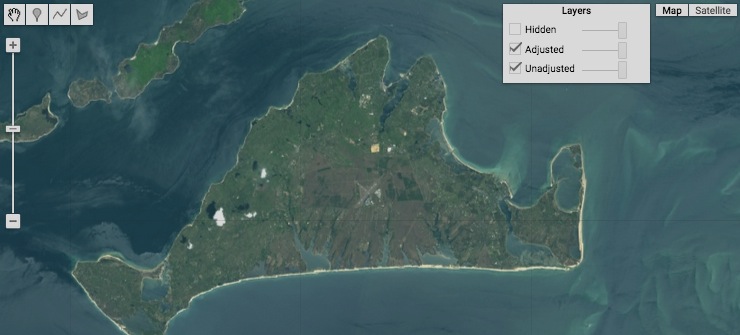
converted to an image ( by using

**imageCollection.mosaic** ) before

being added to the map

The name for the layer to be displayed, given as a string. This name will appear on the the map’s layer list listing.Default: "Layer N," where N is 1, 2, 3, and so on according to the order in which layers are added to the display.

var TheIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-05-01','2014-06-01');

Map.setCenter( -70.6126, 41.3923, 11 ); // Martha’s Vineyard

Map.addLayer ( TheIMAGES, null, 'Unadjusted' );

Map.addLayer ( TheIMAGES, { bands:['B4,B3,B2'],

gain:1000, bias:[ 0,20, 0] }, 'Adjusted' );

Map.addLayer ( TheIMAGES, { bands:['B3,B2,B1'],

gain:800, bias :[20, 0, 0] }, 'Hidden', false );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **MAPS**

Map.addLayer ( imageCollection ) … continued

Map.addLayer ( imageCollection*,  settings, name, visibility* )

Addition instructions on how to display the specified image collection

**min: #, max: ##** gives **#** as a number specifying the lower end of the range of values that are to be represented by whatever range of colors are called for by **palette** settings and/or **##** to specify the upper end of that value range. Values below **#** and above **##** are respectively represented by lowest and highest extremes of the color range. **#** and **##** may be given as numbers (referring to all bands) or as lists of numbers (referring to consecutive bands). **min** and **max** settingscannot be used in conjunction with **gain** and **bias** settings.

**gamma: #** gives **#** as a number ranging from 0 to 10 that calls for greater discrimination in the display of lower values (when # is higher) or higher values (when # is lower). **gamma** may be given either as a single number referring to all bands or as a list of consecutive numbers referring to (all) consecutive bands.

**opacity: #** gives **#** as a number ranging from 0 to 1 that calls for greater opacity in the display of values (when # is higher) or transparency (when # is

lower). Opacity can also be adjusted interactively by using the layer list in the upper right corner of the map.

**palette: X** gives **X** as the range of colors to be used to display the range of values on a single-band. Each color within this range is given as a string of three two-character codes that indicate the amount of redness, greenness, and blueness (in that order) comprising the color. The characters in each two-character code are digits that represent quantities ranging from 0 to 15 as follows: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Thus, A (or a) represents ten, B (or b) represents eleven, and so on while A0 represents 160, while A represents 161, and FF represents 255. To the right are some examples; each six-character code appears in the color it represents as described [here](http://www.w3schools.com/tags/ref_colormixer.asp). Ranges of color are given as lists (or concatenations) of these strings such that the order in which colors are listed determines the order in which they appear in the color range, each occupying an equal portion of that range.

**000000 505050**

**FF0000 800000**

**FFAA00 DDCCAA**

**FFFF00 808000**

**00FF00 008000**

**00FFFF 008080**

**0000FF 000080**

**FF00FF 800080**

**FFFFFF 808080**

**000000 505050**

**FF0000 800000**

**FFAA00 DDCCAA**

**FFFF00 808000**

**00FF00 008000**

**00FFFF 008080**

**0000FF 000080**

**FF00FF 800080**

**FFFFFF 808080**

var TheIMAGES = ee.ImageCollection( 'LC8\_L1T\_TOA' ).filterDate( '2014-05-01','2014-06-01'); // Martha’s Vineyard

Map.setCenter( -70.6126, 41.3923, 11 );

Map.addLayer ( TheIMAGES,

 { bands:'B4,B3,B2', min:0, max:0.2, opacity:0.7 }, 'Adjusted' );

Map.addLayer ( TheIMAGES,

{ bands:'B4,B3,B2', min:0, max:0.2, opacity:0.7, gamma:0.5 }, 'Lower Gamma' );

Map.addLayer ( TheIMAGES.select(0),

{ min:0, max:0.4, palette:'000000,cccccc,ff0000' }, 'Single Band' );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **CHARTS**

Chart.image.series presents a chart

- whose X axis identifies the values of a specified property of a specified image;

- whose Y axis identifies the values of that image’s bands; and

- whose charted data indicate values reduced within a specified region for each of a specified series of bands.

newChart = Chart.image.series( oldImageCollection*, region, regionReducer, scale, xProperty* )

The specified image collection

The specified region, given as a feature

or geometry. Default: all geometries

within the image collection.

The region reducer’s

pixel resolution, given

as a number of meters

The property from which

to draw X-axis labels.

Default: “system:itime\_start”

The new chart

The specified region reducer, which must compute a single value per band. Default: ee.Reducer( ).mean

var TheIMAGES = ee.ImageCollection( 'MODIS/MYD11A1' ).filterDate( '2014-09-01','2015-03-06' );

var TheIMAGES = TheIMAGES.select( ['LST\_Day\_1km','LST\_Night\_1km'] );

var TheFEATURES = ee.FeatureCollection( 'ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8' ); // U.S. States

var TheFEATURE = TheFEATURES.filterMetadata( 'Name','equals','New York' );

var TheCHART = Chart.image.series( TheIMAGES, TheFEATURE, ee.Reducer.mean( ), 1000);

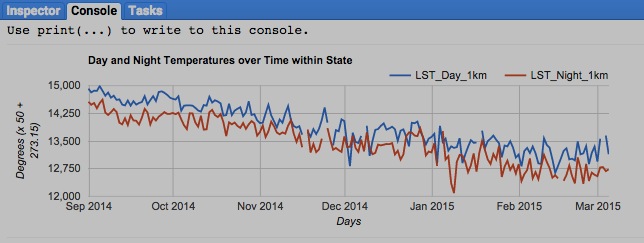
var TheCHART = TheCHART.setOptions( { title: 'Day and Night Temperatures over Time within State',

hAxis: { title: 'Days' },

vAxis: { title: 'Degrees (x 50 + 273.15)' } } );

print( TheCHART );

print( TheIMAGES.getInfo() );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **CHARTS**

Chart.image.seriesByRegion presents a chart

newChart =

- whose X axis identifies the values of a specified property of a specified image;

- whose Y axis identifies the values generated by applying a specified reducer to that image; and

- whose charted data indicate the reduced value of a specified band for each of a specified series of regions.

Chart.image.seriesByRegion( oldImageCollection*, regions, reducer, band, scale, xProperty, seriesProperty* )

The specified

reducer, which

must compute

a single value

per band. Default:

ee.Reducer( )

The region property

from which to draw

series labels.

Default: “system:index”

The reducer’s

pixel resolution,

given as a number of

meters

The specified image collection

The new chart

The specified region(s), given as a feature, a list of features,

a feature collection, a geometry, or a list of geometries.

Default: the entire image.

The property from which

to draw X-axis labels.

Default: “system:time\_start”

The specified band, given as a band-name string or a band number. Default: the initial band

var TheIMAGES = ee.ImageCollection( 'LANDSAT/LC8\_L1T\_8DAY\_TOA' ).filterDate( '2014-01-01','2014-12-31' );

var TheFEATURES = ee.FeatureCollection( 'ft:1G3RZbWoTiCiYv\_LEwc7xKZq8aYoPZlL5\_KuVhyDM' );

var TheFEATURES = TheFEATURES.filterMetadata( 'city\_name','contains','Wash.' ); // Selected Cities

var TheCHART = Chart.image.seriesByRegion(TheIMAGES, TheFEATURES, ee.Reducer.mean( ), 'B1', 30);

var TheCHART = TheCHART.setSeriesNames(['Olympia', 'Seattle', 'Spokane']);

var TheCHART = TheCHART.setOptions( { title: 'Reflectance over Time by City',

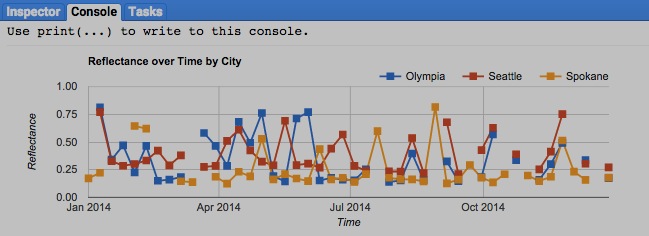
pointShape: 'square',

pointSize: 10,

hAxis: { title: 'Time' },

vAxis: { title: 'Reflectance' } } );

print( TheCHART );



[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **CHARTS**

Chart.image.doySeries presents a chart

- whose X axis identifies a specified range of days of the year;

- whose Y axis identifies the values generated by applying two specified reducers to the bands of a specified image collection; and

- whose charted data indicate values reduced by region and by day (over the years) for each of a specified series of bands.

newChart =

Chart.image.doySeries( oldImageCollection*, region, regionReducer, scale, yearReducer, start, stop* )

The region reducer’s

pixel resolution, given

as a number of meters

Initial day of the year, given as an integer between

1and 366

Final day of

the year, given as an integer between

**startDay**

and 366

The new chart

The specified

image collection

The specified region, given as a feature

or geometry.Default: all geometries

within the image collection.

The specified region reducer, which must compute a single value per band. Default: ee.Reducer( ).mean

The specified reducer summarizing each day’s annual regionReducer outputs with a single value per band. Default: ee.Reducer( ).mean

var TheIMAGES = ee.ImageCollection( 'MODIS/MYD11A1').filterDate( '2013-01-01','2015-01-01' );

var TheIMAGES = TheIMAGES.select( ['LST\_Day\_1km','LST\_Night\_1km'] );

var TheFEATURES = ee.FeatureCollection( 'ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8' ); // U.S. States

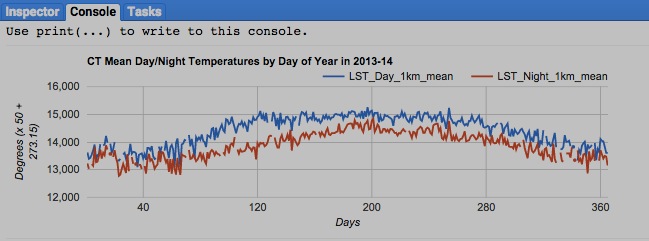
var TheFEATURE = TheFEATURES.filterMetadata( 'Name','equals','Connecticut' );

var TheCHART = Chart.image.doySeries(TheIMAGES, TheFEATURE, ee.Reducer.mean( ), 2000);

var TheCHART = TheCHART.setOptions( { title: 'CT Mean Day/Night Temperatures by Day of Year in 2013-14',

hAxis: { title: 'Days' },

vAxis: { title: 'Degrees (x 50 + 273.15)' } } );

print( TheCHART );

print( TheIMAGES.getInfo() );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **CHARTS**

Chart.image.doySeriesByRegion

presents a chart - whose X axis identifies a specified range of days of the year;

- whose Y axis identifies the values generated by applying two specified reducers to a specified band of a specified image collection; and

- whose charted data indicate values reduced by region and by day (over the years) for each of a specified series of bands.

newChart = Chart.image.doySeriesByRegion

( oldImageCollection*,* band, *regions, regionReducer, scale, yearReducer, seriesProperty, start, stop* )

The specified

image

collection

The specified band,

given by band-name

string or by number

The region reducer’s

pixel resolution, given

as a number of meters

The feature property from which to draw series labels.

Default: “system:index”

Final

day of

the year, given as

an integer between

**startDay**

and 366

The new

chart

The specified region(s), given as a feature, a list of features, a feature

collection, a geometry, or a list of geometries. Default: entire image.

The specified reducer summarizing each day’s annual **regionReducer**

outputs with a single value per band. Default: ee.Reducer( ).mean

Initial day of the year, given as

an integer between 1and 366

The specified region reducer, which must compute a single value per band. Default: ee.Reducer( ).mean

var TheIMAGES = ee.ImageCollection('MODIS/MYD11A1').filterDate('2013-01-01','2015-01-01');

var TheIMAGES = TheIMAGES.select( ['LST\_Day\_1km','LST\_Night\_1km'] );

var TheFEATURES = ee.FeatureCollection( 'ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8' ); // U.S. States

var TheFEATURE = TheFEATURES.filterMetadata( 'Name','contains','North' );

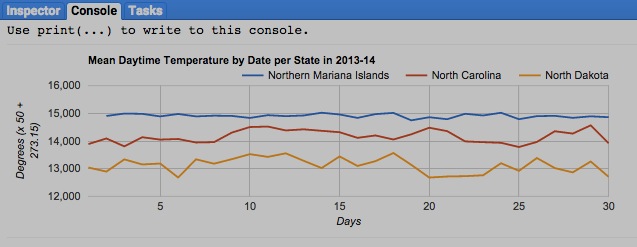
var TheCHART = Chart.image.doySeriesByRegion( TheIMAGES, 0, TheFEATURE, ee.Reducer.mean( ), 10000,

ee.Reducer.mean( ),'Name', 1, 30);

var TheCHART = TheCHART.setOptions( { title: 'Mean Daytime Temperature by Date per State in 2013-14',

hAxis: { title: 'Days' },

vAxis: { title: 'Degrees (x 50 + 273.15)' } } );



print( TheCHART );

print( TheIMAGES.getInfo() );

[GOOGLE EARTH ENGINE](EE01%20Earth%20Engine%20(EE).docx) [APPLICATION PROGRAMMING INTERFACE](EE05%20%20%20The%20EE%20API.docx) [CAPABILITIES](EE07%20%20%20%20%20%20API%20Capabilities.docx)

**PRESENTING** [IMAGE COLLECTION](#_top) [VARIABLES](EE13%20%20%20%20%20%20%20%20%20Variables.docx) IN **CHARTS**

Chart.image.doySeriesByYear

presents a chart - whose X axis identifies a specified range of days of the year;

- whose Y axis identifies the values generated by applying two specified reducers to a specified band of a specified image collection; and

- whose charted data indicate values reduced by region and by day (over the years) for each of a specified series of years.

newChart = Chart.image.doySeriesYear

( oldImageCollection*,* band, *region, regionReducer, scale, dayYearReducer, startDay, stopDay* )

The specified

image

collection

The specified band,

given by band-name

string or by number

The region reducer’s

pixel resolution, given

as a number of meters

The feature property from which to draw series labels.

Default: “system:index”

Final

day of

the year, given as

an integer between

**startDay**

and 366

The new

chart

The specified region(s), given as a feature, a list of features, a feature

collection, a geometry, or a list of geometries. Default: entire image.

The specified reducer summarizing day-year values over images

with a single value per band. Default: ee.Reducer( ).mean

Initial day of the year, given as

an integer between 1and 366

The specified region reducer, which must compute a single value per band. Default: ee.Reducer( ).mean

var TheIMAGES = ee.ImageCollection('MODIS/MYD11A1').filterDate('2013-01-01','2015-01-01');

var TheIMAGES = TheIMAGES.select( ['LST\_Day\_1km','LST\_Night\_1km'] );

var TheFEATURES = ee.FeatureCollection('ft:1fRY18cjsHzDgGiJiS2nnpUU3v9JPDc2HNaR7Xk8'); // U.S. States

var TheFEATURE = TheFEATURES.filterMetadata( 'Name','contains','Island');

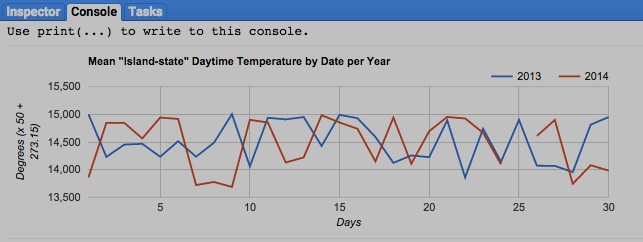
var TheCHART = Chart.image.doySeriesByYear( TheIMAGES, 0, TheFEATURE, ee.Reducer.mean( ), 10000,

ee.Reducer.mean( ), 1, 30);

var TheCHART = TheCHART.setOptions( { title: 'Mean "Island-state" Daytime Temperature by Date per Year',

hAxis: { title: 'Days' },

vAxis: { title: 'Degrees (x 50 + 273.15)' } } );



print( TheCHART );

print( TheIMAGES.getInfo() );