cartoee Documentation

Kel Markert

Jan 17, 2019

Introduction:

`cartoee` is a simple Python package used for making publication quality maps from Earth Engine results using Cartopy without having to export results from Earth Engine.

This packages aims to do only one thing well: getting processing results from Earth Engine into a publication quality mapping interface. `cartoee` simply gets results from Earth Engine and plots it with the correct geographic projections leaving `ee` and `cartopy` to do more of the processing and visualization.

CHAPTER 1

Overview

`cartoee` is a simple Python package used for making publication quality maps from Earth Engine results using Cartopy without having to download results.

This packages aims to do only one thing well: getting processing results from Earth Engine into a publication quality mapping interface. `cartoee` simply gets results from Earth Engine and plots it with the correct geographic projections leaving `ee` and `cartopy` to do more of the processing and visualization.

A typical Earth Engine workflow includes:

1. Processing your data on Earth Engine
2. Exporting your data from Earth Engine
3. Creating maps of your results

Here, we omit the 2nd step and merge steps 1 and 3 into one step. This allows users to process their data using the Python Earth Engine API and quickly create a map.

1.1 Installation

`cartoee` is available to install via `pip`. To install the package, you can use pip install for your Python environment:

```
pip install cartoee
```

Or, you can install the package manually from source code using the following commands:

```
git clone https://github.com/kmarkert/cartoee.git
cd cartoee
python setup.py install
```

1.1.1 Dependencies

`cartoee` is built using pure Python code however relies on a few dependencies (earthengine-api and cartopy) that are available. Users are referred to the dependencies documentation for full installation instructions:

- Matplotlib installation
- Earth Engine API installation
- Cartopy installation

The easiest way to get dependencies installed correctly is to use conda for matplotlib and cartopy. The Earth Engine API requires installation through pip:

```
conda install -c conda-forge matplotlib cartopy
pip install google-api-python-client
pip install earthengine-api
# make sure the latest oauth2client library is installed
pip install --upgrade oauth2client
# autheticate earthengine python api
earthengine authenticate
```

CHAPTER 2

A simple plotting example

[1]:
```
import ee
import cartoee as cee
import cartopy.crs as ccrs

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

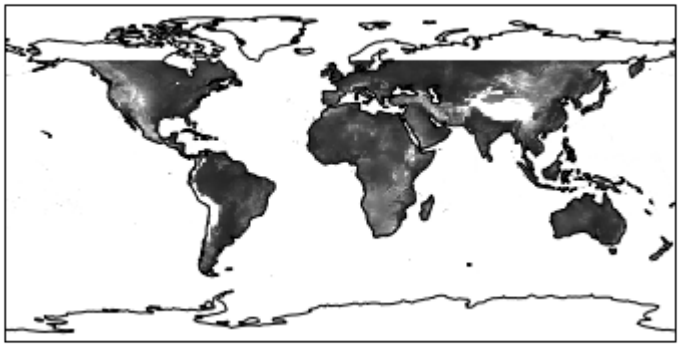
[2]:
```
ee.Initialize()
```

2.1 Plotting an EE Image on a map

[3]:
```
# get an earth engine image
srtm = ee.Image("CGIAR/SRTM90_V4")
```

[4]:
```
# specify visualization parameters and region to map
visualization = {'min':-500,'max':3000,'bands':'elevation'}
bbox = [-180,-90,180,90]
```

[5]:
```
# plot the result using cartoee
ax = cee.getMap(srtm,region=bbox,visParams=visualization)

ax.coastlines()
plt.show()
```

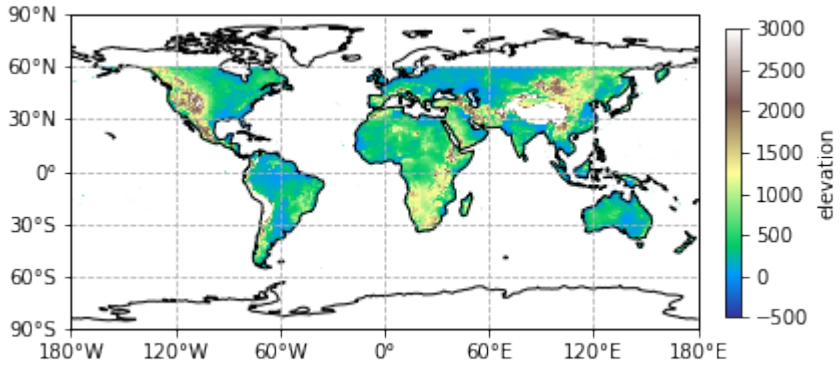


This is a basic example where we are simply getting the image result and rendering it on a map with the correct geographic projection. Usually we want to make out maps a little prettier...

2.2 Customizing maps

Here this example shows how we can plot an EE image with a specific colormap (from matplotlib), add a colorbar, and stylize our map with `cartopy`.

[6]:
```
from cartopy.mpl.gridliner import LATITUDE_FORMATTER, LONGITUDE_FORMATTER

# plot the map
ax = cee.getMap(srtm,cmap='terrain',region=bbox,visParams=visualization)
# add a color bar using cartoee
cb = cee.addColorbar(ax,loc='right',cmap='terrain',visParams=visualization)

ax.coastlines()

# set gridlines and spacing
xticks = [-180,-120,-60,0,60,120,180]
yticks = [-90,-60,-30,0,30,60,90]
ax.gridlines(xlocs=xticks, ylocs=yticks,linestyle='--')

# set custom formatting for the tick labels
ax.xaxis.set_major_formatter(LONGITUDE_FORMATTER)
ax.yaxis.set_major_formatter(LATITUDE_FORMATTER)

# set tick labels
ax.set_xticks([-180,-120,-60, 0, 60, 120, 180], crs=ccrs.PlateCarree())
ax.set_yticks([-90, -60, -30, 0, 30, 60, 90], crs=ccrs.PlateCarree())

plt.show()
```

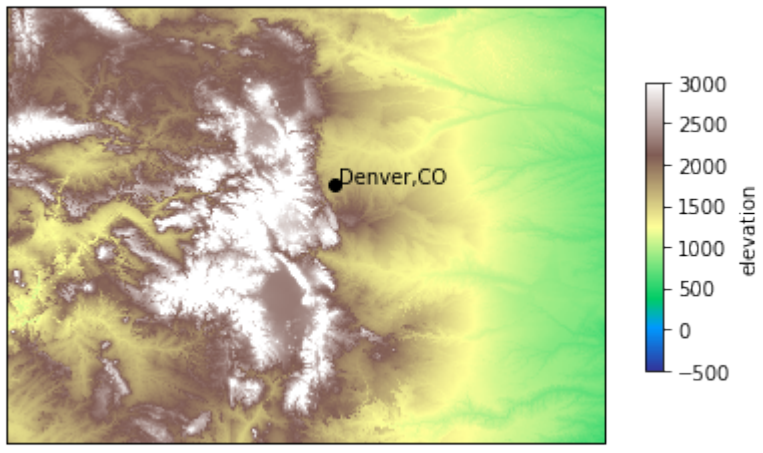


Now we have a map with some aesthetics! We provided a 'cmap' keyword to our `cee.getMap` function to provide color and used the same visualization and cmap parameters to provide a colorbar that is consistent with our EE image.

2.3 Plotting a specific region

A lot of times we don't have global results and want to make a map for a specific region. Here we can customize where we make a map and plot our EE image results.

[7]:
```
# specify new region over Colorado
# showing the Great Continental Divide that splits the state
newRegion = [-111,35,-100,43]

ax = cee.getMap(srtm,cmap='terrain',region=newRegion,visParams=visualization,
                dims=2000)

# add a colorbar to our map
cb = cee.addColorbar(ax,loc='right',cmap='terrain',visParams=visualization)

# add a marker for Denver,CO
ax.plot(-104.9903,39.7392,'ko')
ax.text(-104.9,39.78,'Denver,CO')

plt.show()
```



In this example, we took a global image (SRTM data) and clipped it down to the region required for plotting. This is

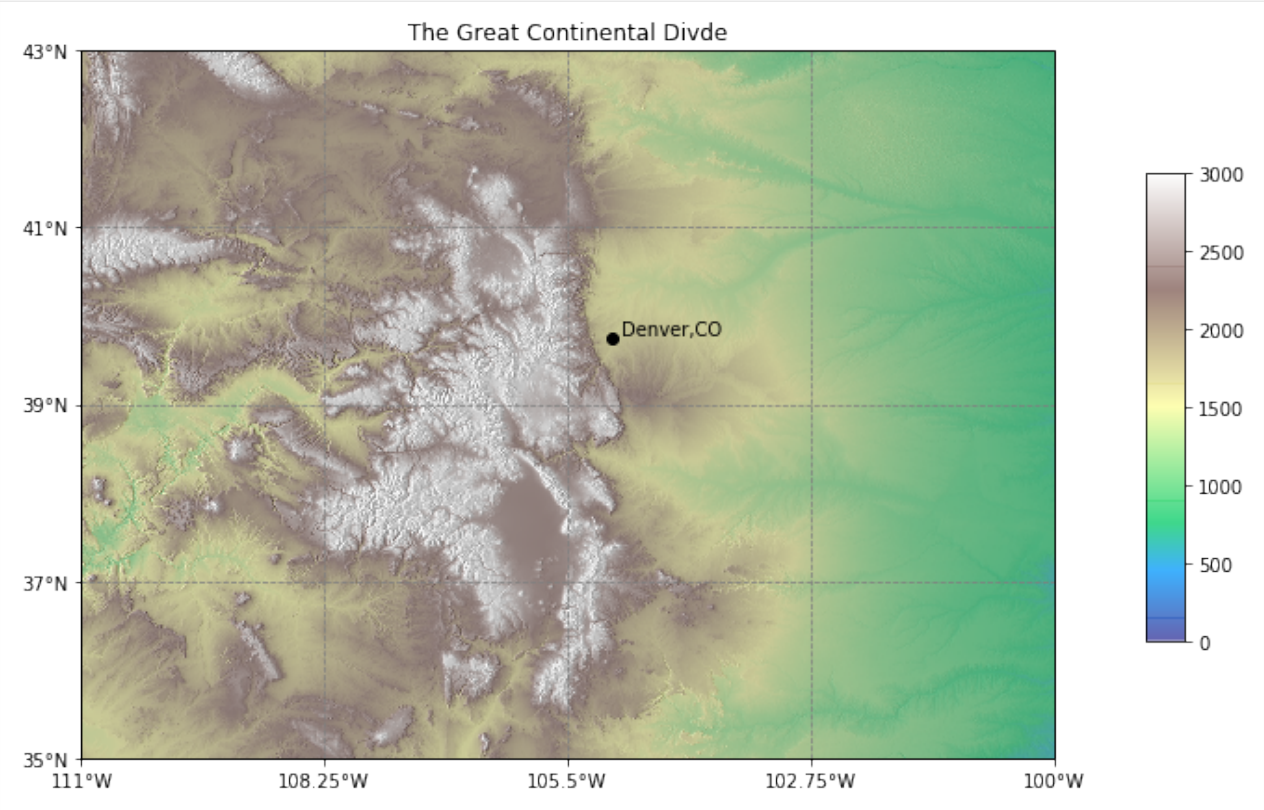
helpful for showing map insets or focusing on particular regions of your results.

2.4 Adding multiple images to a map

Many times we don't just have one image to show and `cartoee` allows for multiple images to be added to a map, and here is how.

[8]:
```
# send a processing request to EE
# calcualte a hillshade from SRTM data and specify the visualization
hillshade = ee.Terrain.hillshade(srtm,azimuth=285,elevation=30)

#create new visualization parameters for the hillshade and elevation data
hsVis = {'min':25,'max':200,'palette':'000000,ffffff'}
elvVis = {'min':0,'max':3000,'opacity':0.5}
```

[9]:
```
# set up a blank map
fig = plt.figure(figsize=(15,7))
ax = plt.subplot(projection=ccrs.PlateCarree())

# plot our hillshade on the blank map
# *note: we are using the cee.addLayer function here and
#        passing our map into addLayer as a keyword. This
#        will get that map with the image overlayed
ax = cee.addLayer(hillshade,ax=ax,region=newRegion
                  ,visParams=hsVis,dims=[2000,1000])

# plot SRTM data over the hillshade with some adjusted visualization parameters
# *note: we are passing the map variable again as a keyword
#        into the addLayer function to stack images on each other
ax = cee.addLayer(srtm,ax=ax, cmap='terrain',region=newRegion,
              visParams=elvVis,dims=[2000,1000])

cax = ax.figure.add_axes([0.8,0.25,0.02,0.5])
cb = cee.addColorbar(ax,cax=cax,cmap='terrain',visParams=elvVis)

# add some styling to make our map publication ready
xticks = np.linspace(-111,-100,5)
yticks = np.linspace(35,43,5)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.gridlines(xlocs=xticks, ylocs=yticks,linestyle='--',color='gray')
# set custom formatting for the tick labels
ax.xaxis.set_major_formatter(LONGITUDE_FORMATTER)
ax.yaxis.set_major_formatter(LATITUDE_FORMATTER)

# set a title so we know where it is
ax.set_title('The Great Continental Divde')

# add a marker for Denver,CO
ax.plot(-104.9903,39.7392,'ko')
ax.text(-104.9,39.78,'Denver,CO')

plt.show()
```



Now we have a nice map show elevation with some nice hillshade styling underneath it!

CHAPTER 3

Working with projections in cartoee

[1]:
```
import ee
import cartoee as cee
import cartopy.crs as ccrs

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
ee.Initialize()
```

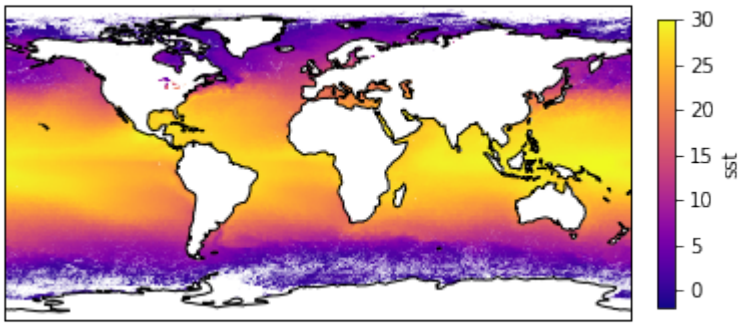
3.1 Plotting an image on a map

Here we are going to show another example of creating a map with EE results. We will use global sea surface temperature data for 2018.

[3]:
```
# get an earth engine image of ocean data for 2018
ocean = ee.ImageCollection('NASA/OCEANDATA/MODIS-Terra/L3SMI')\
        .filter(ee.Filter.date('2018-01-01', '2019-01-01')).median()
```

[4]:
```
# set parameters for plotting
# will plot the Sea Surface Temp with specific range and colormap
visualization = {'bands':'sst','min':-2,'max':30}
# specify region to focus on
bbox = [-180,-90,180,90]
```

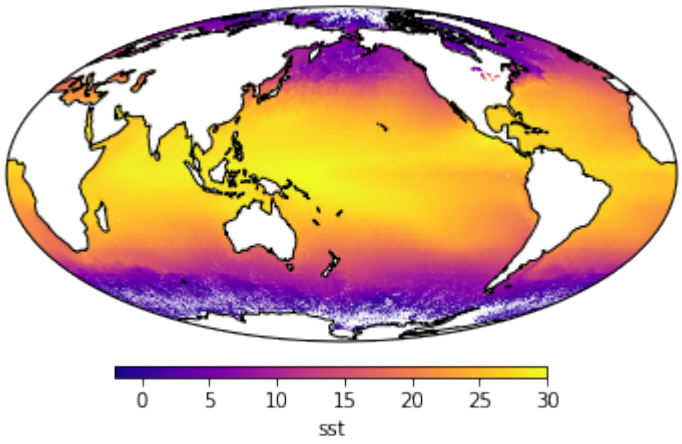
[5]:
```
# plot the result with cartoee using a PlateCarre projection (default)
ax = cee.getMap(ocean,cmap='plasma',visParams=visualization,region=bbox)
cb = cee.addColorbar(ax,loc='right',cmap='plasma',visParams=visualization)

ax.coastlines()
plt.show()
```



3.2 Mapping with different projections

You can specify what ever projection is available within `cartopy` to display the results from Earth Engine. Here are a couple examples of global and regions maps using the sea surface temperature example. Please refer to the `Cartopy projection documentation <https://scitools.org.uk/cartopy/docs/latest/crs/projections.html>‘__ for more examples with different projections.

[6]:
```
# create a new Mollweide projection centered on the Pacific
projection = ccrs.Mollweide(central_longitude=-180)

# plot the result with cartoee using the Mollweide projection
ax = cee.getMap(ocean,visParams=visualization,region=bbox,
                cmap='plasma',proj=projection)
cb = cee.addColorbar(ax,loc='bottom',cmap='plasma',visParams=visualization,
                     orientation='horizontal')


ax.coastlines()
plt.show()
```

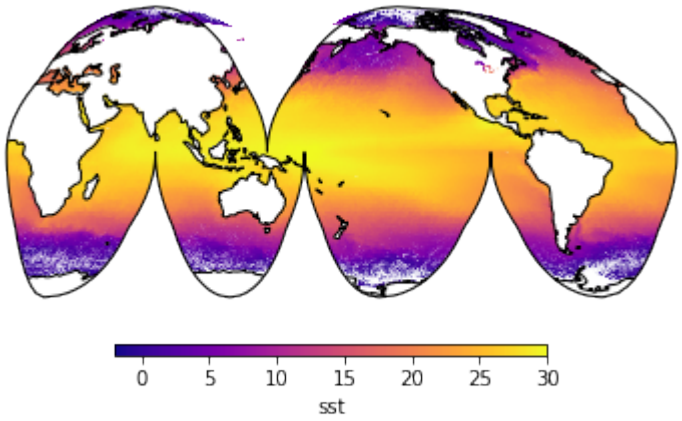


[7]:
```
# create a new Goode homolosine projection centered on the Pacific
projection = ccrs.InterruptedGoodeHomolosine(central_longitude=-180)

# plot the result with cartoee using the Goode homolosine projection
ax = cee.getMap(ocean,visParams=visualization,region=bbox,
```

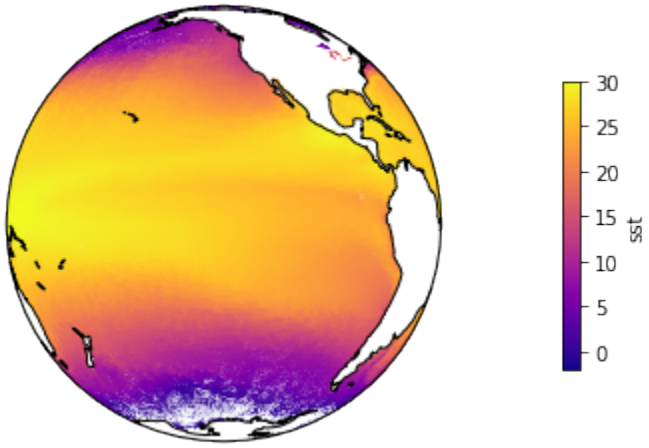
(continues on next page)

(continued from previous page)

```
                  cmap='plasma',proj=projection)
cb = cee.addColorbar(ax,loc='bottom',cmap='plasma',visParams=visualization,
                     orientation='horizontal')

ax.coastlines()
plt.show()
```



[8]:
```
# create a new orographic projection focused on the Pacific
projection = ccrs.Orthographic(-130,-10)

# plot the result with cartoee using the orographic projection
ax = cee.getMap(ocean,visParams=visualization,region=bbox,
                cmap='plasma',proj=projection)
cb = cee.addColorbar(ax,loc='right',cmap='plasma',visParams=visualization,
                     orientation='vertical')
ax.coastlines()
plt.show()
```

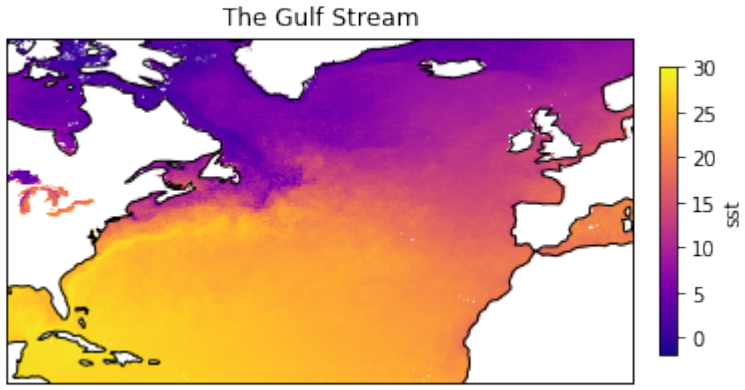


[9]:
```
# Create a new region to focus on
natlantic = [-90,15,10,70]

# plot the result with cartoee focusing on the north Atlantic
ax = cee.getMap(ocean,cmap='plasma',visParams=visualization,region=natlantic)
cb = cee.addColorbar(ax,loc='right',cmap='plasma',visParams=visualization,)
```

(continues on next page)

(continued from previous page)

```
ax.coastlines()
ax.set_title('The Gulf Stream')
plt.show()
```



CHAPTER 4

Customizing colorbars in cartoee

[1]:
```
import ee
import cartoee as cee
import cartopy.crs as ccrs

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
ee.Initialize()
```

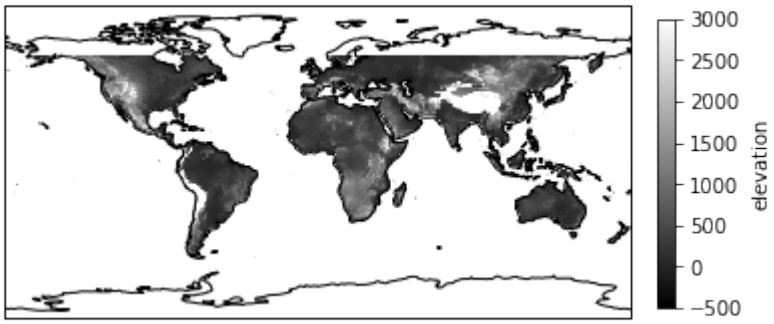
4.1 Creating a map with a colorbar

`Cartoee` provides a simple way to create a map from EE, however, the visualization parameters one passes to EE do not directly correspond to matplotlib. To accomidate the difference and make it simple to user, `cartoee` has a function `addColorbar()` that takes a mixture of EE and matplotlib parameters. Let's see some examples.

[3]:
```
# get an earth engine image
srtm = ee.Image("CGIAR/SRTM90_V4")
```

[4]:
```
# specify visualization parameters and region to map
# *note: the visualization variable is what EE takes to visualize the data
visualization = {'min':-500,'max':3000,'bands':'elevation'}
bbox = [-180,-90,180,90]
```

[5]:
```
# plot the result using cartoee
ax = cee.getMap(srtm,cmap='gray',region=bbox,visParams=visualization)
cb = cee.addColorbar(ax,loc='right',cmap='gray',visParams=visualization)

ax.coastlines()
plt.show()
```

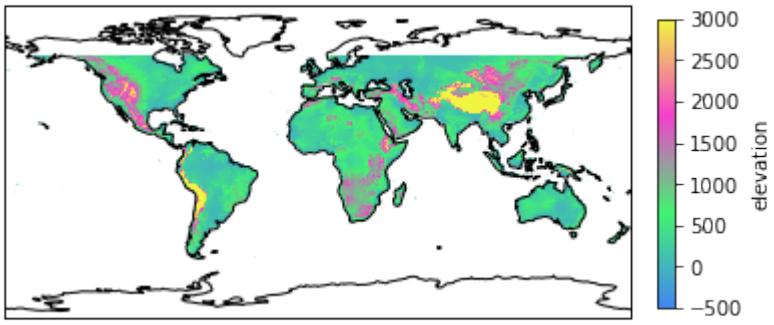


So, here we created a map of SRTM data and added a colorbar. Simple enough, right? Earth Engine will default to a grayscale colormap if no palette is given. Matplotlib, on the other hand, uses the viridis colormap. When using cartoee, it is be to simply be explicit when it comes to the cmap keyword (helps you make sure the map and colorbar are the same).

Now let's see an example with a custom palette that is not available in the matplotlib's arsenal of colormaps.

4.2 Playing with colorbars

[6]:
```
# visualization parameters with an extremely wacky palette
visualization = {'min':-500,'max':3000,'bands':'elevation',
                 'palette':'#4286f4,#41f471,#f441cd,#f1f441'}

# plot the result using cartoee
# *note: cartoee will not let you plot the data if cmap is specify with
#        a palette in the visParams
ax = cee.getMap(srtm,region=bbox,visParams=visualization)
cb = cee.addColorbar(ax,loc='right',visParams=visualization)

ax.coastlines()
plt.show()
```



In this example we provided a custom colormap as a comma-seprated string of hex color codes in the visualization dictionary with the palette key. When we pass the visualization parameters into the `addColorbar` function we get that palette as a nice colorbar.

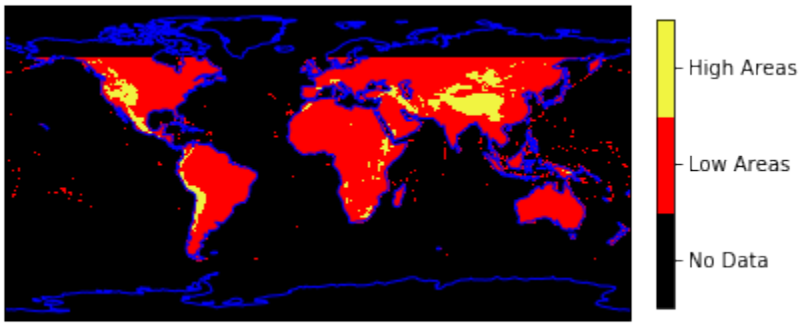
We can also make discrete colorbars... let's see how that works.

[7]:
```
# classify the SRTM image into 3 classes, nodata, <=1500, and >1500
classified = ee.Image(0).where(srtm.lte(1500),1)\
```

(continues on next page)

(continued from previous page)

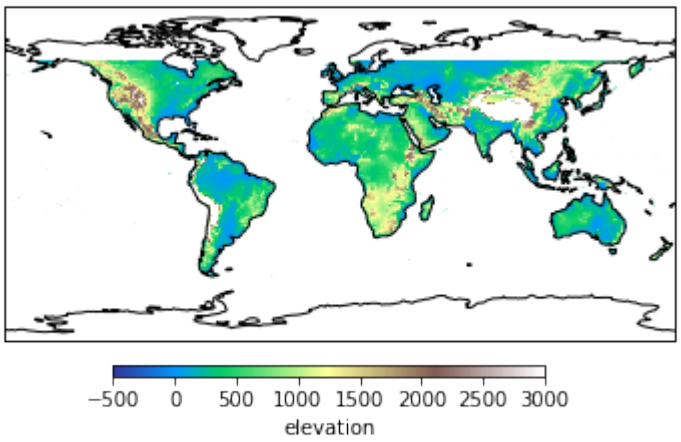
```
                        .where(srtm.gt(1500),2)

# visualization parameters with 3 colors in the palette
visualization = {'min':0,'max':2,'palette':'#000000,#FF0000,#f1f441'}

ax = cee.getMap(classified,region=bbox,visParams=visualization)
# create a colorbar but set the discrete keyword to true for categorical data
cb = cee.addColorbar(ax,loc='right',visParams=visualization,discrete=True)

# customize where the ticks are on the colorbar
cb.set_ticks([0.333,0.9999,1.666])
# and set custom labels
cb.set_ticklabels(['No Data','Low Areas','High Areas'])

ax.coastlines(color='blue')
plt.show()
```



4.3 Colorbar positioning

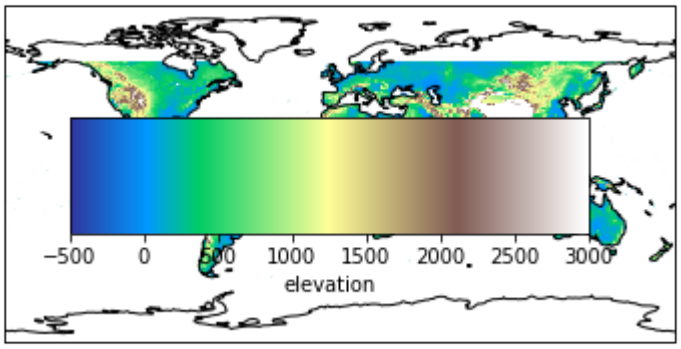
There are many more fun things you can do with the colors but what about position? Yes, you can select a position to display to colorbar. Or, better yet bring your own location

[8]:
```
# going back to the original visualization parameters
visualization = {'min':-500,'max':3000,'bands':'elevation'}

# plot the result using cartoee
ax = cee.getMap(srtm,cmap='terrain',region=bbox,visParams=visualization)
# create the colorbar but set the location to bottom
cb = cee.addColorbar(ax,loc='bottom',cmap='terrain',visParams=visualization,
                     orientation='horizontal')

ax.coastlines()
plt.show()
```



`Cartoee` comes with predetermined locations for the colorbars ('left', 'right', 'bottom', and 'top') to help make it easy for users to create a plot. Sometimes you just want to put something where you want it when really customizing map, and you can.

[9]:
```
# plot the result using cartoee
ax = cee.getMap(srtm,cmap='terrain',region=bbox,visParams=visualization)

# create a new axes to add the colorbar to in the middle of the map
cax = ax.figure.add_axes([0.2,0.4,0.6,0.2])

# create the colorbar but set the cax keyword
cb = cee.addColorbar(ax,cax=cax,cmap='terrain',visParams=visualization,
                     orientation='horizontal')

ax.coastlines()
plt.show()
```



This is obviously a joke to demostrate that you can add the colorbar wherever you would like. This gives you the basics of colorbars in cartoee which allows users to do more fun things and make custom publication quality maps with EE!

CHAPTER 5

Multiple maps using subplots

[1]:
```
import ee
import cartoee as cee
import cartopy.crs as ccrs

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

[2]:
```
ee.Initialize()
```

5.1 Seasonal NDVI example

In this example we will compute the seasonal average NDVI values for the globe and plot the four seasons on the same figure.

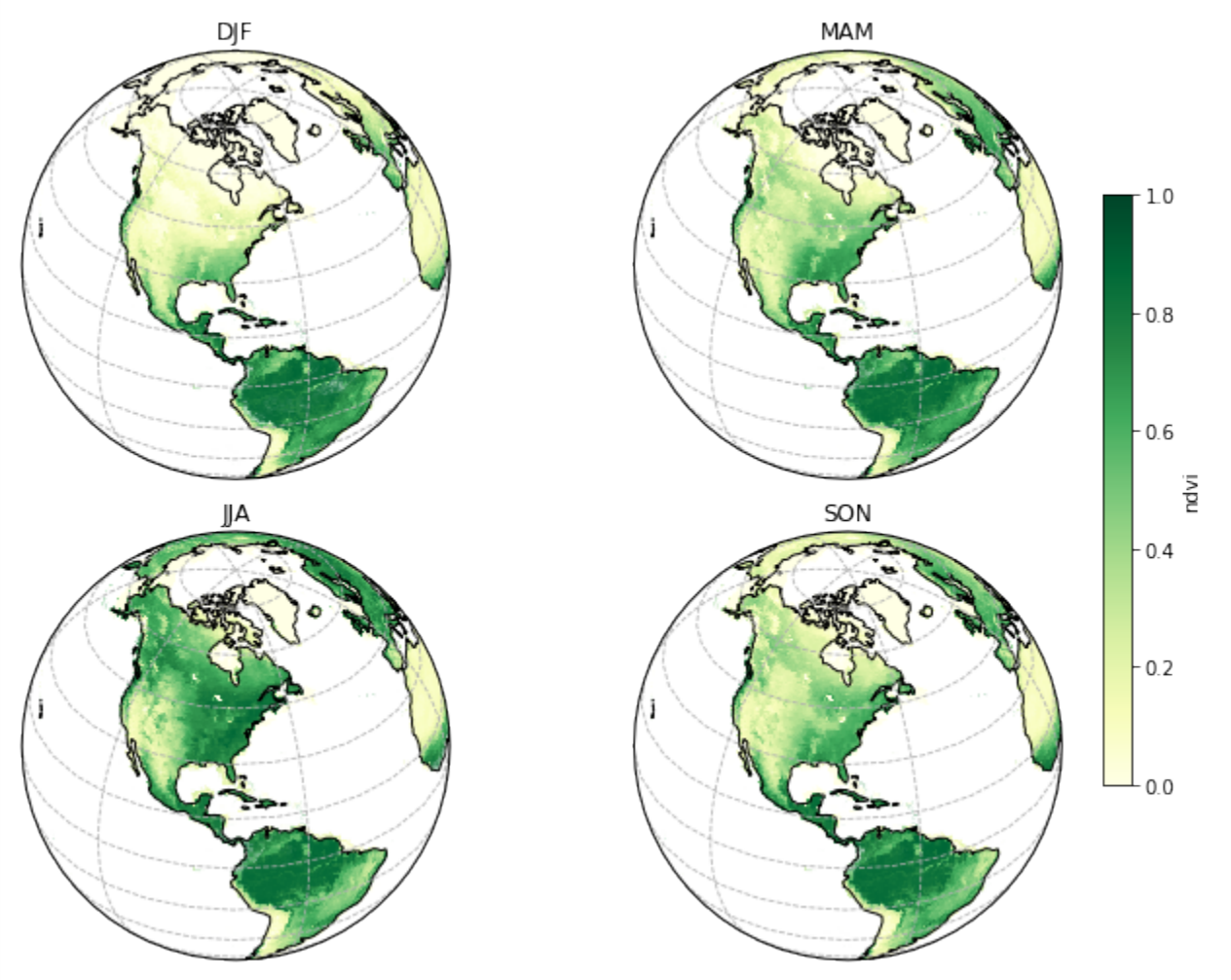
[3]:
```
# function to add NDVI band to imagery
def calc_ndvi(img):
    ndvi = img.normalizedDifference(['Nadir_Reflectance_Band2','Nadir_Reflectance_
↪Band1'])
    return img.addBands(ndvi.rename('ndvi'))
```

[4]:
```
# MODIS Nadir BRDF-Adjusted Reflectance with NDVI band
modis = ee.ImageCollection('MODIS/006/MCD43A4')\
        .filterDate('2010-01-01','2016-01-01')\
        .map(calc_ndvi)
```

[5]:
```
# set parameters for plotting
ndviVis = {'min':0,'max':1,'bands':'ndvi'}
bbox = [-180,-60,180,90]
```

[6]:
```
# get land mass feature collection
land = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')

# calculate seasonal averages and clip to land features
djf = modis.filter(ee.Filter.calendarRange(12,3,'month')).mean().clip(land)
mam = modis.filter(ee.Filter.calendarRange(3,6,'month')).mean().clip(land)
jja = modis.filter(ee.Filter.calendarRange(6,9,'month')).mean().clip(land)
son = modis.filter(ee.Filter.calendarRange(9,12,'month')).mean().clip(land)
```

[7]:
```
# set up a blank map with multiple subplots
fig,ax = plt.subplots(ncols=2,nrows=2,figsize=(10,7),
                      subplot_kw={'projection': ccrs.Orthographic(-80,35)})

# format images and subplot titles with same dimensions as subplots
imgs = np.array([[djf,mam],[jja,son]])
titles = np.array([['DJF','MAM'],['JJA','SON']])

for i in range(len(imgs)):
    for j in range(len(imgs[i])):
        ax[i,j] = cee.addLayer(imgs[i,j],ax=ax[i,j],
                               region=bbox,dims=500,
                               visParams=ndviVis,cmap='YlGn'
                              )
        ax[i,j].coastlines()
        ax[i,j].gridlines(linestyle='--')
        ax[i,j].set_title(titles[i,j])

plt.tight_layout()

cax = fig.add_axes([0.9, 0.2, 0.02, 0.6])
cb = cee.addColorbar(ax[i,j],cax=cax,cmap='YlGn',visParams=ndviVis)

plt.show()
```

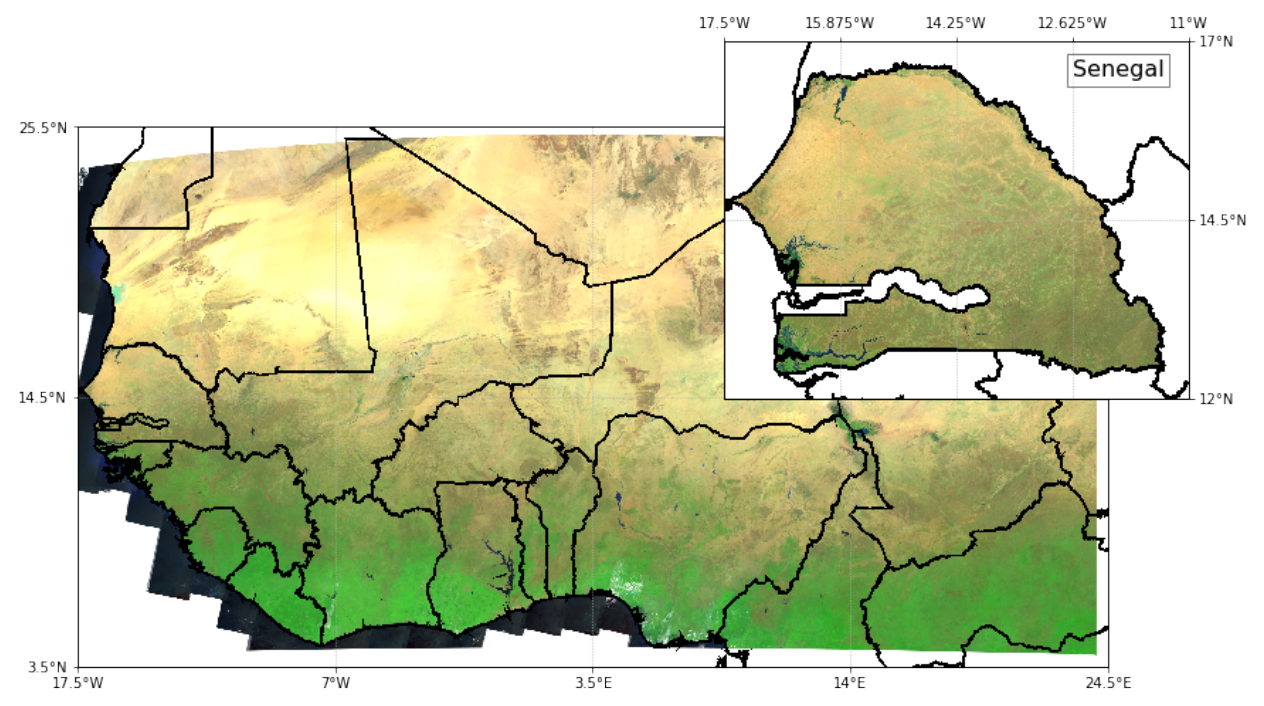


5.2 Map inset example

In this example we are going to take a regional image from EE, plot the entire region, and plot a smaller country within the region as a subset. This specific example will create a map for West Africa showing a Landsat mosaic and create an inset map of Senegal using the same Landsat mosaic.

[8]:
```
# get a country FeatureCollection
countries = ee.FeatureCollection('USDOS/LSIB_SIMPLE/2017')
# filter the countries for a specific county
country = 'Senegal'
senegal = ee.Feature(countries.filter(ee.Filter.eq('country_na',country)).first())
# create an ee.Image from a FeatureCollection
adminImg = ee.Image().paint(countries,color='#000000',width=4)

# get a Landsat mosaic for West Africa
waLandsat = ee.Image('projects/servir-wa/regional_west_africa/Landsat_SR/Landsat_SR_
↪prewet_2012_2014')
# specify the visualization parameters
lsVis = {'min':50,'max':5500,'gamma':1.5,'bands':'swir2,nir,green'}
```

[9]:
```
# import some styling functions
from cartopy.mpl.gridliner import LATITUDE_FORMATTER, LONGITUDE_FORMATTER

# setup blank figure
fig = plt.figure(figsize=(15,7))

# region for the main map
mainBox = [-17.5,3.5,24.5,25.5]

# set blank map for the main plot and add layers
ax_main = fig.add_subplot(1, 1, 1,projection=ccrs.PlateCarree())
ax_main = cee.addLayer(waLandsat,visParams=lsVis,region=mainBox,dims=2500,ax=ax_main)
ax_main = cee.addLayer(adminImg,region=mainBox,dims=1500,ax=ax_main)

# main map styling
xmain = np.linspace(-17.5,24.5,5)
ymain = np.linspace(3.5,25.5,3)
ax_main.gridlines(xlocs=xmain, ylocs=ymain,linestyle=':')
# set custom formatting for the tick labels
ax_main.xaxis.set_major_formatter(LONGITUDE_FORMATTER)
ax_main.yaxis.set_major_formatter(LATITUDE_FORMATTER)
# set tick labels
ax_main.set_xticks(xmain, crs=ccrs.PlateCarree())
ax_main.set_yticks(ymain, crs=ccrs.PlateCarree())


# region for the map inset
insetBox = [-17.5,12,-11,17]

# setup inset map and add layers
ax_inset = fig.add_axes([0.45, 0.5, 0.6, 0.5],projection=ccrs.PlateCarree())
ax_inset = cee.addLayer(waLandsat.clip(senegal),visParams=lsVis,region=insetBox,
↪dims=2500,ax=ax_inset)
ax_inset = cee.addLayer(adminImg,region=insetBox,ax=ax_inset)

# inset map styling
xinset = np.linspace(-17.5,-11,5)
yinset = np.linspace(12,17,3)
ax_inset.gridlines(xlocs=xinset, ylocs=yinset,linestyle=':')
# set custom formatting for the tick labels
ax_inset.xaxis.set_major_formatter(LONGITUDE_FORMATTER)
ax_inset.yaxis.set_major_formatter(LATITUDE_FORMATTER)
ax_inset.xaxis.tick_top()
ax_inset.yaxis.tick_right()
# set inset tick labels
ax_inset.set_xticks(xinset, crs=ccrs.PlateCarree())
ax_inset.set_yticks(yinset, crs=ccrs.PlateCarree())

# add some text to the inset as a pseudo-title
ax_inset.text(-12.63,16.5,country,fontsize=16,bbox=dict(facecolor='white', alpha=0.5,
↪edgecolor='k'))

plt.show()
```


17.5°W
15.875°W
14.25°W
12.625°W
11°W
17°N
14.5°N
12°N
Senegal
25.5°N
14.5°N
3.5°N
17.5°W
7°W
3.5°E
14°E
24.5°E

CHAPTER 6

cartoee API

This page provides an overview of cartoee's API. For detailed examples using the package please refer to the documentation on the docs page.

cartoee.plotting.**addColorbar**(*ax*, *loc=None*, *visParams=None*, *discrete=False*, ***kwargs*)

Add a colorbar tp the map based on visualization parameters provided

Parameters

- **ax** (*cartopy.mpl.geoaxes.GeoAxesSubplot | cartopy.mpl.geoaxes.GeoAxes*) – required cartopy GeoAxesSubplot object to add image overlay to
- **loc** (*str, optional*) – string specifying the position
- **visParams** (*dict, optional*) – visualization parameters as a dictionary. See https://developers.google.com/earth-engine/image_visualization for options
- ****kwargs** – remaining keyword arguments are passed to colorbar()

Returns matplotlib colorbar object

Return type cb (matplotlib.colorbar.ColorbarBase)

Raises

- **Warning** – If 'discrete' is true when "palette" key is not in visParams
- **ValueError** – If *ax* is not of type cartopy.mpl.geoaxes.GeoAxesSubplot
- **ValueError** – If 'cmap' or "palette" key in visParams is not provided
- **ValueError** – If "min" in visParams is not of type scalar
- **ValueError** – If "max" in visParams is not of type scalar
- **ValueError** – If 'loc' or 'cax' keywords are not provided
- **ValueError** – If 'loc' is not of type str or does not equal available options

cartoee.plotting.**addLayer**(*imgObj*, *ax*, *dims=None*, *region=None*, *cmap=None*, *visParams=None*)

Add an Earth Engine image to a cartopy plot.

Parameters

- **imgObj** (*ee.image.Image*) – Earth Engine image result to plot.
- **ax** (*cartopy.mpl.geoaxes.GeoAxesSubplot | cartopy.mpl.geoaxes.GeoAxes*) – required cartopy GeoAxesSubplot object to add image overlay to
- **dims** (*list | tuple | int, optional*) – dimensions to request earth engine result as [WIDTH,HEIGHT]. If only one number is passed, it is used as the maximum, and the other dimension is computed by proportional scaling. Default None and infers dimesions
- **region** (*list | tuple, optional*) – geospatial region of the image to render in format [E,S,W,N]. By default, the whole image
- **cmap** (*str, optional*) – string specifying matplotlib colormap to colorize image. If cmap is specified visParams cannot contain 'palette' key
- **visParams** (*dict, optional*) – visualization parameters as a dictionary. See https://developers.google.com/earth-engine/image_visualization for options

Returns cartopy GeoAxesSubplot object with Earth Engine results displayed

Return type ax (cartopy.mpl.geoaxes.GeoAxesSubplot)

Raises

- **ValueError** – If *dims* is not of type list, tuple, or int
- **ValueError** – If *imgObj* is not of type ee.image.Image
- **ValueError** – If *ax* if not of type cartopy.mpl.geoaxes.GeoAxesSubplot '

cartoee.plotting.**buildPalette**(*cmap*, *n=256*)

Creates hex color code palette from a matplotlib colormap

Parameters

- **cmap** (*str*) – string specifying matplotlib colormap to colorize image. If cmap is specified visParams cannot contain 'palette' key
- **n** (*int, optional*) – Number of hex color codes to create from colormap. Default is 256

Returns list of hex color codes from matplotlib colormap for n intervals

Return type palette (list)

cartoee.plotting.**getMap**(*imgObj*, *proj=<cartopy.crs.PlateCarree object>*, ***kwargs*)

Wrapper function to create a new cartopy plot with project and adds Earth Engine image results

Parameters

- **imgObj** (*ee.image.Image*) – Earth Engine image result to plot
- **proj** (*cartopy.crs, optional*) – Cartopy projection that determines the projection of the resulting plot. By default uses an equirectangular projection, PlateCarree
- ****kwargs** – remaining keyword arguments are passed to addLayer()

Returns cartopy GeoAxesSubplot object with Earth Engine results displayed

Return type ax (cartopy.mpl.geoaxes.GeoAxesSubplot)

Contact: Cartoee is on GitHub at https://github.com/kmarkert/cartoee. Please report issues there.

Python Module Index

C

Index