MAP: Estimation of biophysical attributes
Dominic Hewitt
do6743he-s

May 2, 2021

I used this opportunity to apply what I'm learning about python, Jupyter notebooks and LaTeX. The following is a Jupyter notebook I created for doing this assignment with added commentary with LaTeX

1 loading the data

First I needed to import the required packages

[63]:
```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import rasterio
import numpy as np
from rasterio.plot import show, show_hist
from scipy.stats import linregress, describe, rv_histogram
```

Then I load the excel file into a GeoPandas Dataframe, I also set the CRS

[2]:
```
df = pd.read_excel('field_observation.xls')
point_data = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.X, df.Y))
point_data.set_crs(epsg=32644, inplace=True)
```

[2]:
```
      Plot       X        Y         nnx         nny  canopy density (%)  \
0        0  548171  3057513  273.033333  416.966667                 0.0
1        1  547415  3052674  247.833333  578.266667                81.0
2        2  547177  3052725  239.900000  576.566667                46.0
3        3  546887  3052673  230.233333  578.300000                84.8
4        4  546619  3052552  221.300000  582.333333                79.0
..     ...     ...      ...         ...         ...                 ...
367    367  549780  3065424  326.666667  153.266667                76.0
368    368  550326  3065855  344.866667  138.900000                81.0
369    369  549604  3065668  320.800000  145.133333                81.0
```

```
370   370  548628  3065963  288.266667  135.300000                75.1
371   371  548309  3065676  277.633333  144.866667                86.0

     light intensity                          geometry
0              18.40  POINT (548171.000 3057513.000)
1               2.20  POINT (547415.000 3052674.000)
2              11.80  POINT (547177.000 3052725.000)
3               1.90  POINT (546887.000 3052673.000)
4               2.30  POINT (546619.000 3052552.000)
..               ...                             ...
367             4.27  POINT (549780.000 3065424.000)
368             1.60  POINT (550326.000 3065855.000)
369             3.60  POINT (549604.000 3065668.000)
370             2.87  POINT (548628.000 3065963.000)
371             1.00  POINT (548309.000 3065676.000)

[372 rows x 8 columns]
```

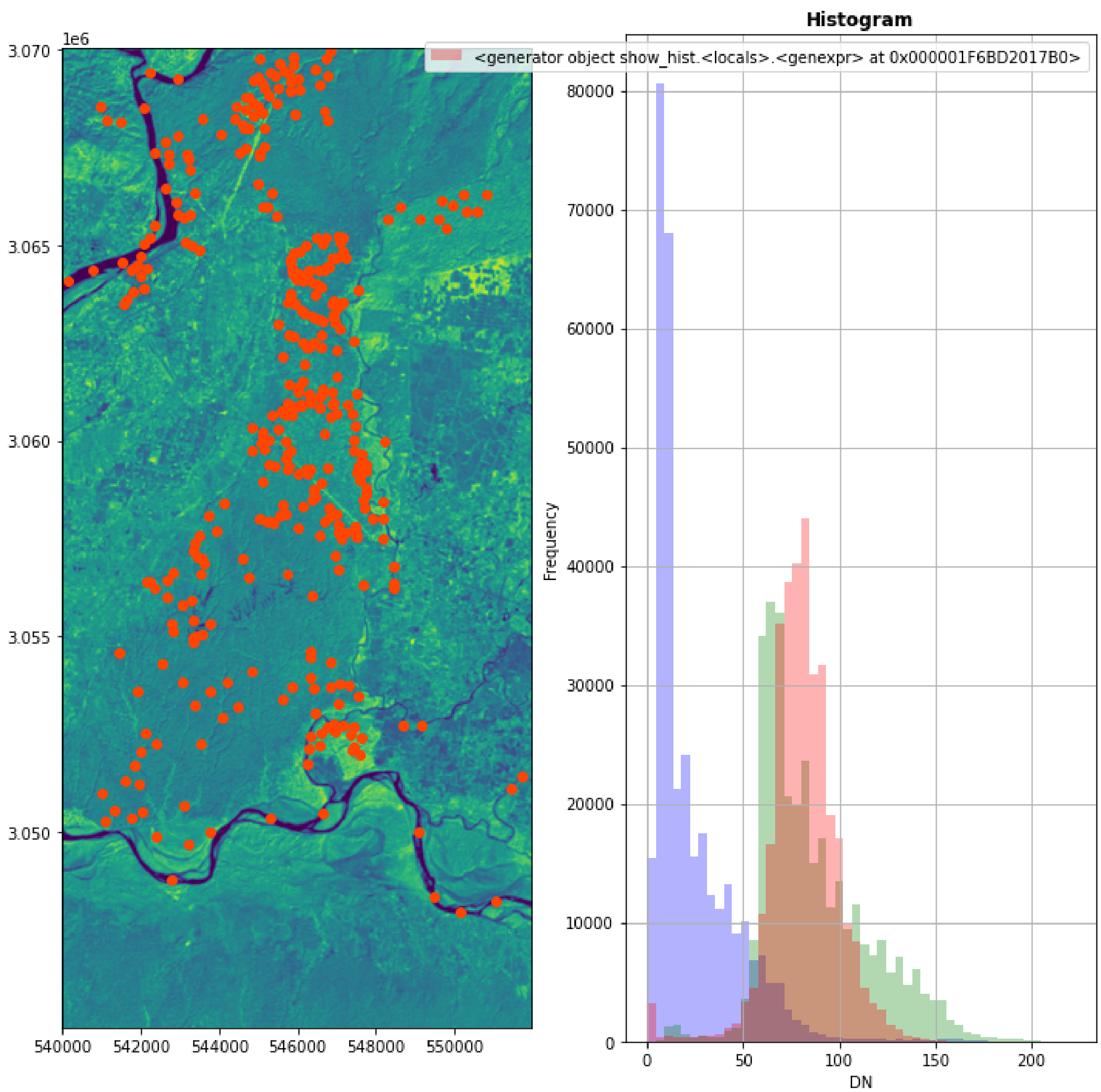
I then used GDAL to extract the required bands and convert the raster to .tif format.

[3]:
```
!gdal_translate np_20011024_refl.img -b 4 -b 5 -b 3 np_20011024_refl.tif
```

```
Input file size is 400, 834
0...10...20...30...40...50...60...70...80...90...100 - done.
```

Next I load the raster data and plot sample points on top to visualise the data
I also added a histogram of the bands DNs

[4]:
```
s = rasterio.open('np_20011024_refl.tif')
fig, (ax, axhist) = plt.subplots(1, 2, figsize=(12,12))
point_data.plot(ax=ax, color='orangered')
show(s, ax=ax)
show_hist(s, bins=50, histtype='stepfilled', lw=0.0, stacked=False, alpha=0.3,␣
 ↪ax=axhist)
plt.show()
```



2 Extract samples and calculate SVIs

I then used rasterio to to extract the DNs values from the raster, and then calculated the NDVI and SR values

[79]:
```
# extract raster values at points using rasterio .sample()
coords = [(x,y) for x, y in zip(point_data.X, point_data.Y)]
point_data['Raster Value'] = [x for x in s.sample(coords)]

# calculate NDVI
point_data['NDVI'] = (point_data['Raster Value'].str[0] - point_data['Raster␣
↪Value'].str[2]) / (point_data['Raster Value'].str[0] + point_data['Raster␣
↪Value'].str[2])
```

```
# calculate SR
point_data['SR'] = point_data['Raster Value'].str[0]/point_data['Raster Value'].
 ↪str[2]

# a look at the dataframe with extracted DNs, calculated NDVI ans SR indices␣
 ↪values
point_data.head()
```

[79]:
```
   Plot       X        Y         nnx         nny  canopy density (%)  \
0     0  548171  3057513  273.033333  416.966667                 0.0
1     1  547415  3052674  247.833333  578.266667                81.0
2     2  547177  3052725  239.900000  576.566667                46.0
3     3  546887  3052673  230.233333  578.300000                84.8
4     4  546619  3052552  221.300000  582.333333                79.0

   light intensity                        geometry    Raster Value      NDVI  \
0             18.4  POINT (548171.000 3057513.000)  [145, 152, 38]  0.584699
1              2.2  POINT (547415.000 3052674.000)    [128, 90, 3]  0.954198
2             11.8  POINT (547177.000 3052725.000)     [93, 76, 5]  0.897959
3              1.9  POINT (546887.000 3052673.000)    [132, 88, 8]  0.885714
4              2.3  POINT (546619.000 3052552.000)    [117, 82, 8]  0.872000

          SR
0   3.815789
1  42.666667
2  18.600000
3  16.500000
4  14.625000
```

I'd noticed later on in the analysis, that there was one or more of the SR values causing an error by giving an infinity value. So for the sake of continuity I with deal with it here

[9]:
```
point_data['SR'].describe()
```

[9]:
```
count    372.000000
mean            inf
std             NaN
min        0.000000
25%        4.826923
50%        8.286364
75%       11.500000
max             inf
Name: SR, dtype: float64
```

...and replace these inf values with Nan values

[10]:
```
point_data['SR'].replace(np.inf, np.nan, inplace=True)
```

3 Correlation coefficients and Scatter plots

The the correlation coefficients between single spectral bands and the forest canopy attributes and between SVIs and forest canopy attributes were calculated

[11]:
```
canopy_band4 = point_data['canopy density (%)'].corr(point_data['Raster Value'].
 ↪str[0])
canopy_band5 = point_data['canopy density (%)'].corr(point_data['Raster Value'].
 ↪str[1])
canopy_band3 = point_data['canopy density (%)'].corr(point_data['Raster Value'].
 ↪str[2])
canopy_NDVI = point_data['canopy density (%)'].corr(point_data['NDVI'])
canopy_SR = point_data['canopy density (%)'].corr(point_data['SR'])

light_band4 = point_data['light intensity'].corr(point_data['Raster Value'].
 ↪str[0])
light_band5 = point_data['light intensity'].corr(point_data['Raster Value'].
 ↪str[1])
light_band3 = point_data['light intensity'].corr(point_data['Raster Value'].
 ↪str[2])
light_NDVI = point_data['light intensity'].corr(point_data['NDVI'])
light_SR = point_data['light intensity'].corr(point_data['SR'])
```

...and added to a dict and convert to a dataframe for easy reading

[12]:
```
corr_dict = {'biological attributes': ['canopy density (%)', 'light intensity'],
             'Band 4': [canopy_band4, light_band4],
             'Band 5': [canopy_band5, light_band5],
             'Band 3': [canopy_band3, light_band3],
             'NDVI': [canopy_NDVI, light_NDVI],
             'SR': [canopy_SR, light_SR]}

corr_df = pd.DataFrame(corr_dict)
corr_df = corr_df.transpose()
corr_df
```

[12]:

	canopy density (%)	light intensity
Band 4	0.320603	-0.323536
Band 5	-0.216024	0.206974
Band 3	-0.53462	0.525831

```
NDVI                                  0.53777        -0.533861
SR                                   0.608042        -0.594625
```

Above we can see that, so far, SR (simple ratio) has the highest correlation with canopy cover and light intensity

I then tested the correlation between the two biophysical attributes

[13]:
```
point_data['canopy density (%)'].corr(point_data['light intensity'])
```

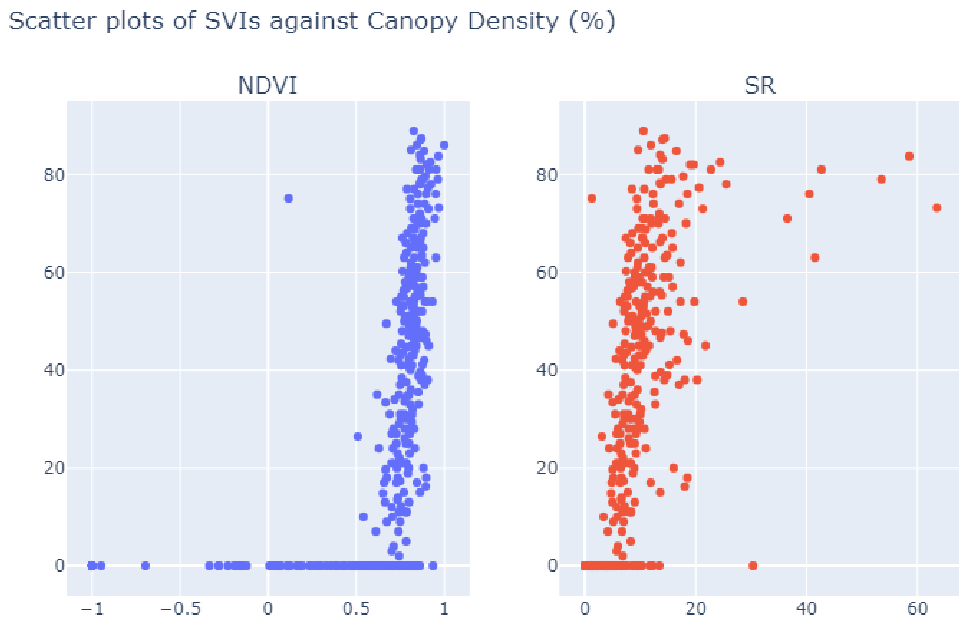
[13]:
```
-0.9886624290330429
```

As they correlate so highly I will from here on use only Canopy Density for workong with bio-physical attributes

Next, I plotted scatter plots of the SVIs against Canopy Density

[74]:
```
# plot SVI values against Canopy Density (%)
fig = make_subplots(rows=1, cols=2,
                    subplot_titles=('NDVI','SR'))
fig.add_trace(
    go.Scatter(x=point_data['NDVI'], y=point_data['canopy density (%)'],␣
 ↪mode='markers'),
    row=1, col=1
)
fig.add_trace(
    go.Scatter(x=point_data['SR'], y=point_data['canopy density (%)'],␣
 ↪mode='markers'),
    row=1, col=2
)
fig.update_layout(title_text="Scatter plots of SVIs against Canopy Density (%)",
                  showlegend=False
)

fig.show()
```



As per the instructions, I filtered out the zero values and also the outlier in the NDVI

```
[16]: point_data_new = point_data[(point_data['canopy density (%)'] > 0.0) &␣
      ↪(point_data['NDVI'] > 0.5)].copy()
```

and then recalculated the correlation coefficients

```
[20]: canopy_band4 = point_data_new['canopy density (%)'].corr(point_data_new['Raster␣
      ↪Value'].str[0])
      canopy_band5 = point_data_new['canopy density (%)'].corr(point_data_new['Raster␣
      ↪Value'].str[1])
      canopy_band3 = point_data_new['canopy density (%)'].corr(point_data_new['Raster␣
      ↪Value'].str[2])
      canopy_NDVI = point_data_new['canopy density (%)'].corr(point_data_new['NDVI'])
      canopy_SR = point_data_new['canopy density (%)'].corr(point_data_new['SR'])

      light_band4 = point_data_new['light intensity'].corr(point_data_new['Raster␣
      ↪Value'].str[0])
      light_band5 = point_data_new['light intensity'].corr(point_data_new['Raster␣
      ↪Value'].str[1])
```

```
light_band3 = point_data_new['light intensity'].corr(point_data_new['Raster␣
↪Value'].str[2])
light_NDVI = point_data_new['light intensity'].corr(point_data_new['NDVI'])
light_SR = point_data_new['light intensity'].corr(point_data_new['SR'])
```

[21]:
```
corr_dict = {'biological attributes': ['canopy density (%)', 'light intensity'],
             'Band 4': [canopy_band4, light_band4],
             'Band 5': [canopy_band5, light_band5],
             'Band 3': [canopy_band3, light_band3],
             'NDVI': [canopy_NDVI, light_NDVI],
             'SR': [canopy_SR, light_SR]}

corr_df = pd.DataFrame(corr_dict)
corr_df = corr_df.transpose()
corr_df
```

[21]:

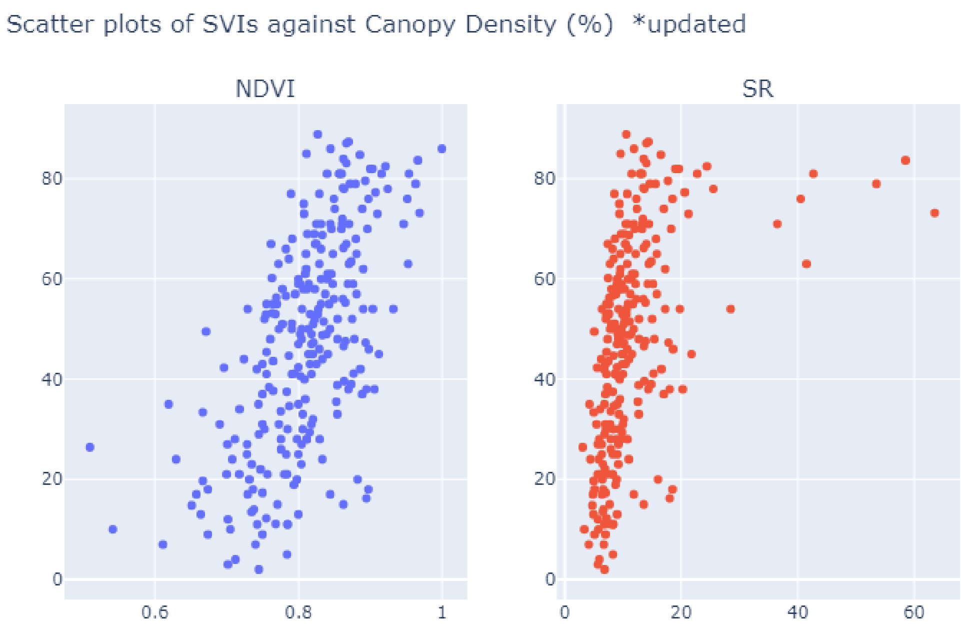
	0	1
biological attributes	canopy density (%)	light intensity
Band 4	0.326074	-0.30899
Band 5	-0.282964	0.286562
Band 3	-0.560606	0.533888
NDVI	0.617229	-0.588258
SR	0.458158	-0.431034

And I noticed the NDVI now shows a stronger relationship with Canopy Density

Re-plotting the scatter plots visualises the result

[76]:
```
# plot SVI values against Canopy Density (%)
fig = make_subplots(rows=1, cols=2,
                    subplot_titles=('NDVI','SR'))
fig.add_trace(
    go.Scatter(x=point_data_new['NDVI'], y=point_data_new['canopy density (%)'],␣
↪mode='markers'),
    row=1, col=1
)
fig.add_trace(
    go.Scatter(x=point_data_new['SR'], y=point_data_new['canopy density (%)'],␣
↪mode='markers'),
    row=1, col=2
)
fig.update_layout(title_text="Scatter plots of SVIs against Canopy Density (%) ␣
↪*updated",
                  showlegend=False
)
```

```
fig.show()
```



4 SVI Raster creation

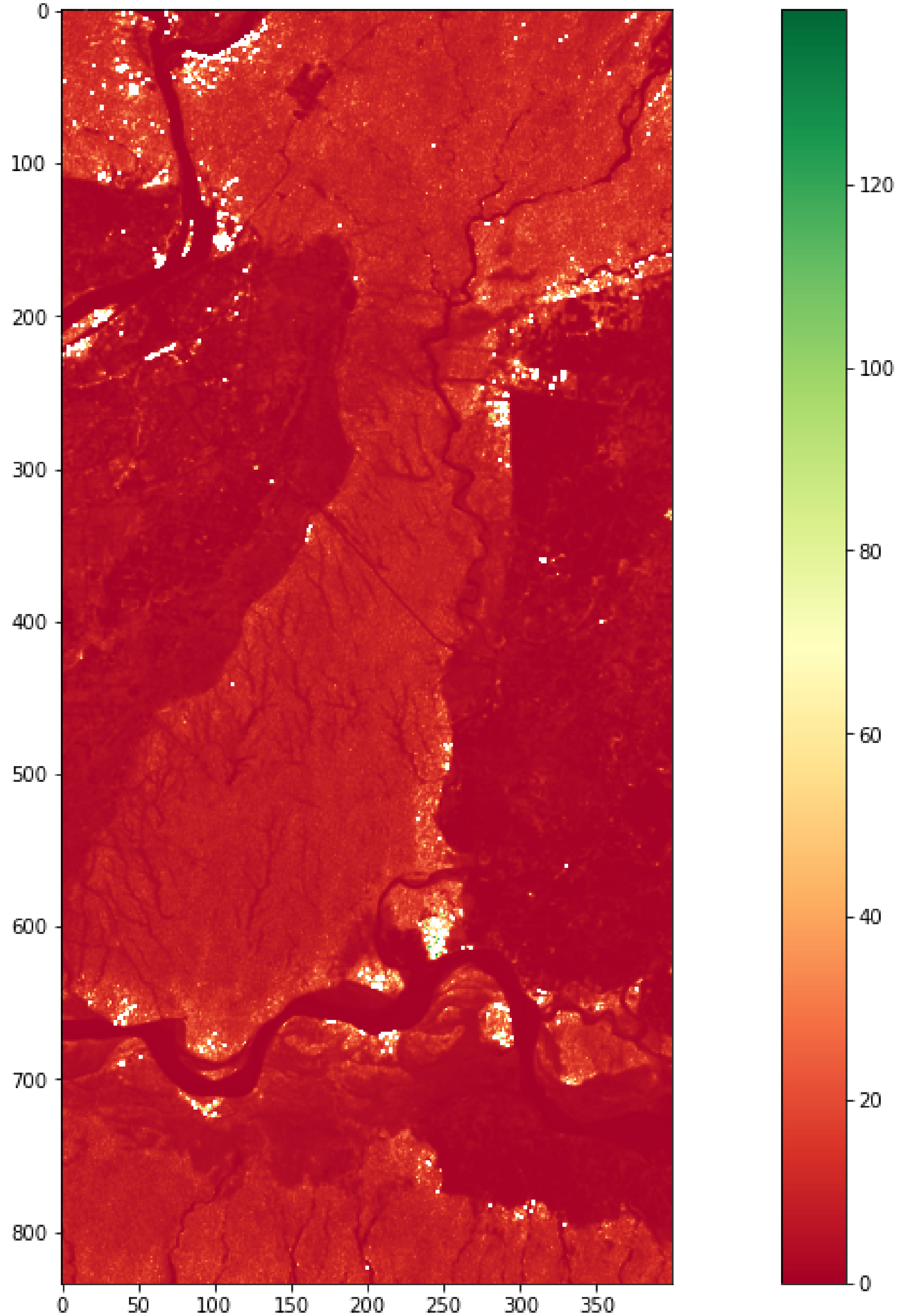
First, I set variables for each band I would be using

[23]:
```
red = s.read(3).astype(float)
nir = s.read(1).astype(float)
np.seterr(divide='ignore', invalid='ignore')   # ignore division errors
```

and proceeded to calculate SR for each pixel and render the layer

[24]:
```
SR = nir/red

plt.figure(figsize = (20,12))
plt.imshow(SR, cmap="RdYlGn")
plt.colorbar()
plt.show()
```

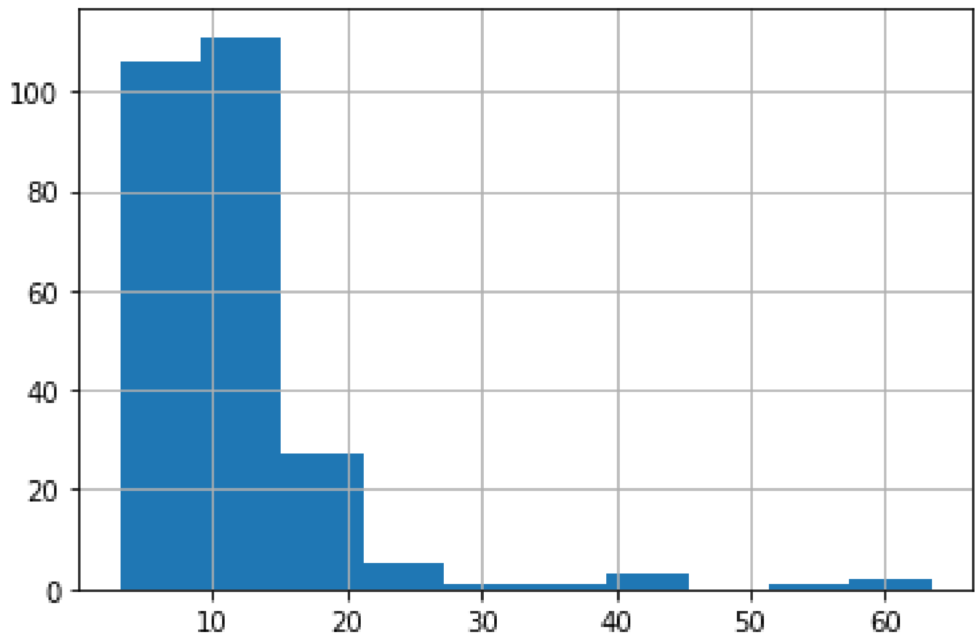


0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300
350
120
100
80
60
40
20
0

the distribution doesnt look right, maybe some erroneous values at the top end ...

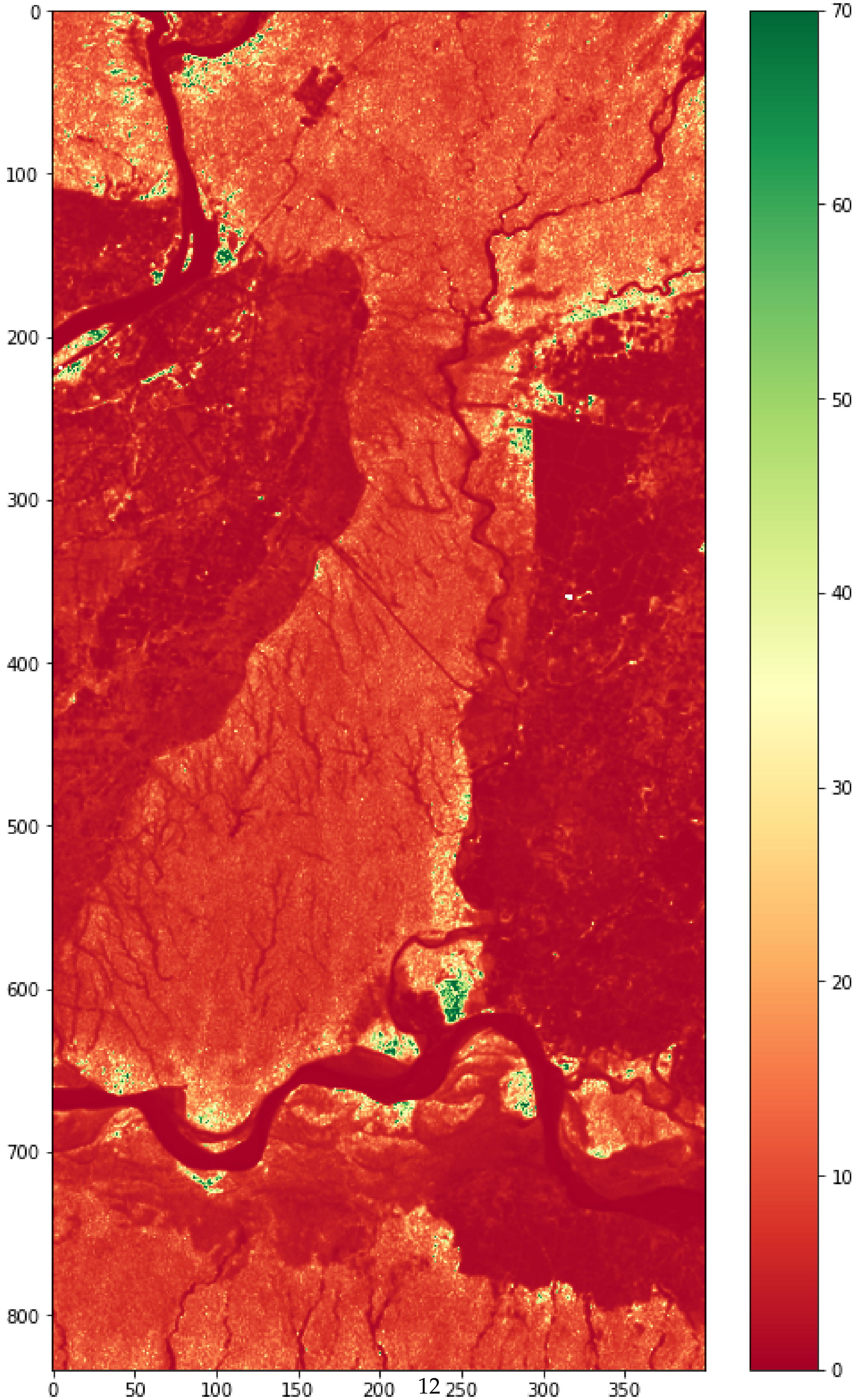
[25]:
```
point_data_new['SR'].hist() # look at the histogram of the point data
```

[25]: <AxesSubplot:>



After looking at the above graph, I re-plotted the raster but clipped the values to those present from the sample points

[59]:
```
# replot with clipped outliers
plt.figure(figsize = (9,14))
plt.imshow(np.clip(SR, 0, 70), cmap="RdYlGn")
plt.colorbar()
plt.show()
```

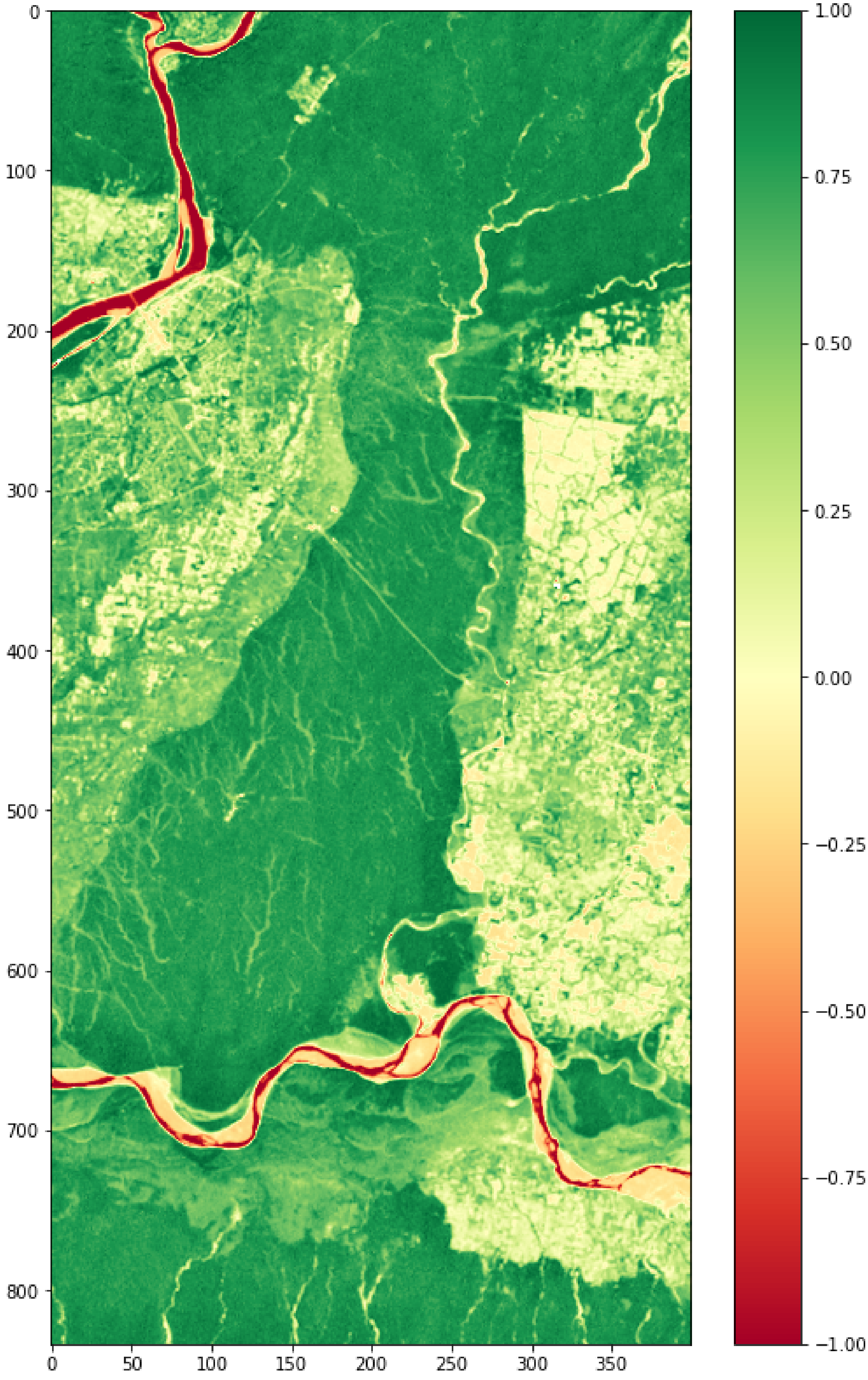

0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300
350
70
60
50
40
30
20
10
0

Then I did the same For NDVI

[58]:
```
NDVI = (nir-red)/(nir+red)

plt.figure(figsize = (9,14))
plt.imshow(NDVI, cmap="RdYlGn")
plt.colorbar()
plt.show()
```

and these values where within the -1 to 1 of the NDVI index



0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300
350
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

5 Regression equation

Using the python package Scipy, I calculated the linear line regression statistics for NDVI vs. Canopy Density

[29]:
```
reg = linregress(point_data_new['NDVI'],point_data_new['canopy density (%)'])
print(reg)
```

```
LinregressResult(slope=186.65945760658215, intercept=-103.7461581835557,
rvalue=0.6172289374610263, pvalue=1.7507047946226463e-28,
stderr=14.870961139036076, intercept_stderr=12.089842416897852)
```

...and built the regression equation

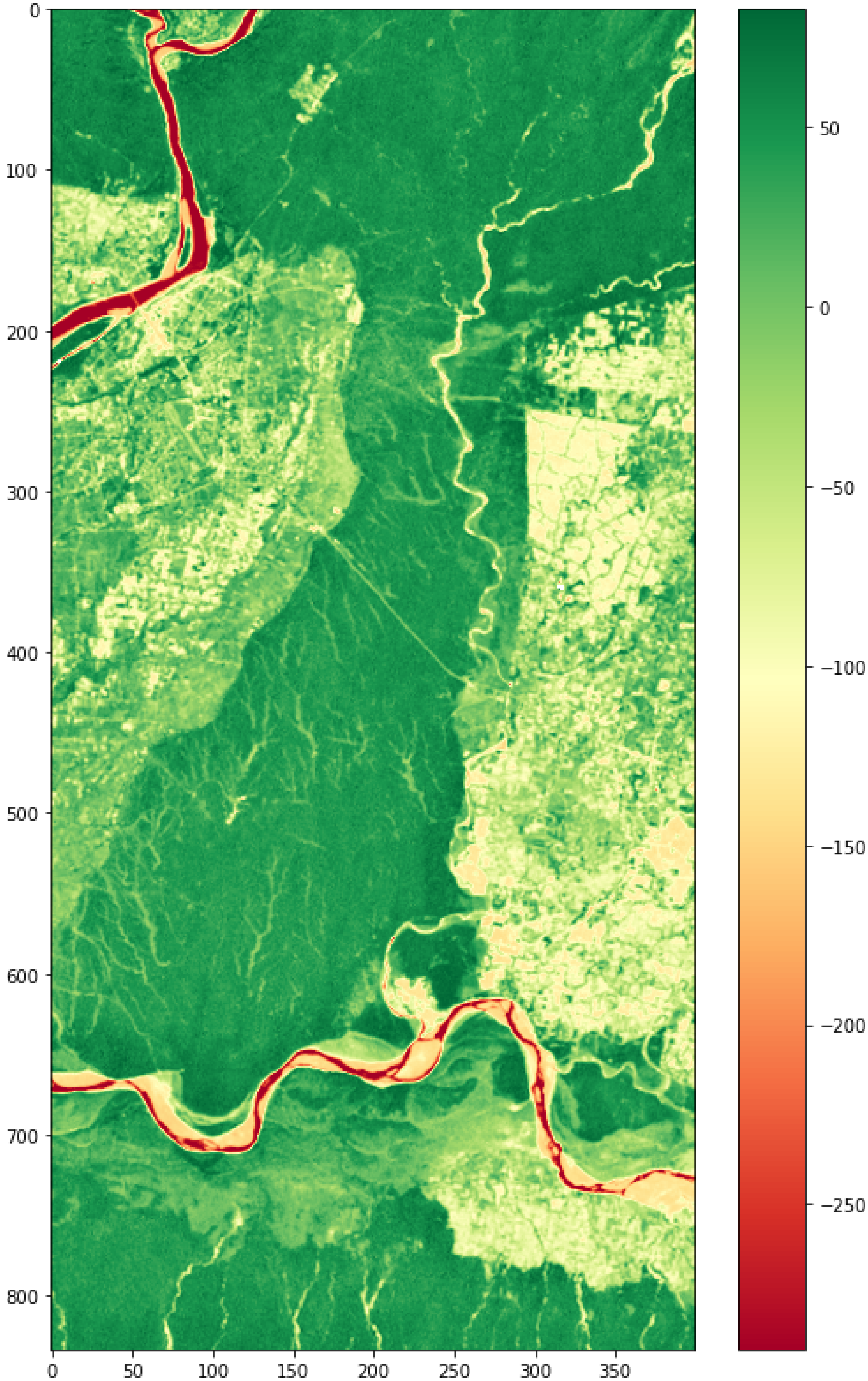
[30]:
```
# build regression equation
print(f'regression equation is: {reg.intercept} + ({reg.slope} * NDVI)')
```

```
regression equation is: -103.7461581835557 + (186.65945760658215 * NDVI)
```

Finally I applied the equation to the NDVI raster and rendered the result

[77]:
```
def NDVI2CD(NDVI):
    return reg.intercept + (reg.slope * NDVI)

plt.figure(figsize = (9,14))
plt.imshow(NDVI2CD(NDVI), cmap="RdYlGn")
plt.colorbar()
plt.show()
```


0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300
350
50
0
−50
−100
−150
−200
−250

Although the raster looks logical, the values are not the expected percentage range (between 0 and 100)
I think this is because there a significantly larger variance of DNs than that taken from the sample plots, including urban areas and waterbodies (which were excluded during analysis). I think a common method is to mask these areas before applying raster transformations/calculations

[37]:
```
plt.hist(NDVI2CD(NDVI))
```

[37]:
```
(array([[ 12.,   0.,   5., ..., 227., 278., 185.],
        [ 13.,   3.,   2., ..., 217., 274., 193.],
        [ 15.,   0.,   1., ..., 215., 305., 191.],
        ...,
        [  1.,   0.,   3., ..., 152., 101., 167.],
        [  1.,   1.,   1., ..., 169.,  96., 172.],
        [  1.,   1.,   0., ..., 159.,  95., 180.]]),
 array([-290.40561579, -253.07372427, -215.74183275, -178.40994123,
        -141.0780497 , -103.74615818,  -66.41426666,  -29.08237514,
           8.24951638,   45.5814079 ,   82.91329942]),
 <a list of 400 BarContainer objects>)
```