Satellite Data Analysis in Python

Python programme

Sentinel-2 satellite image analysis of Australian wildfire

Study Area : Canberra, Australia

In [1]:
```
# Load the Drive helper and mount your Google Drive as a drive in the virt
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [2]:
```
# Change working directory
import os
os.chdir("/content/drive/My Drive/coursework/")
```

In [4]:
```
# Install rasterio and earthpy packages into colab
!pip install rasterio
!pip install earthpy

# Import necessary libraries

from glob import glob

import earthpy as et
import earthpy.spatial as es
import earthpy.plot as ep

import rasterio as rio
from rasterio import plot

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

import numpy as np

np.seterr(divide='ignore', invalid='ignore')
%matplotlib inline
```

```
Requirement already satisfied: rasterio in /usr/local/lib/python3.7/dist-pa
ckages (1.2.1)
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.7/di
st-packages (from rasterio) (7.1.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packa
ges (from rasterio) (1.19.5)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-
packages (from rasterio) (0.7.1)
Requirement already satisfied: click-plugins in /usr/local/lib/python3.7/di
```

```
st-packages (from rasterio) (1.1.1)
Requirement already satisfied: snuggs>=1.4.1 in /usr/local/lib/python3.7/di
st-packages (from rasterio) (1.4.7)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-pac
kages (from rasterio) (2020.12.5)
Requirement already satisfied: affine in /usr/local/lib/python3.7/dist-pack
ages (from rasterio) (2.3.0)
Requirement already satisfied: attrs in /usr/local/lib/python3.7/dist-packa
ges (from rasterio) (20.3.0)
Requirement already satisfied: pyparsing>=2.1.6 in /usr/local/lib/python3.7
/dist-packages (from snuggs>=1.4.1->rasterio) (2.4.7)
Requirement already satisfied: earthpy in /usr/local/lib/python3.7/dist-pac
kages (0.9.2)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-pa
ckages (from earthpy) (2.23.0)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.7/di
st-packages (from earthpy) (1.19.5)
Requirement already satisfied: rasterio in /usr/local/lib/python3.7/dist-pa
ckages (from earthpy) (1.2.1)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dis
t-packages (from earthpy) (0.16.2)
Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.
7/dist-packages (from earthpy) (3.2.2)
Requirement already satisfied: geopandas in /usr/local/lib/python3.7/dist-p
ackages (from earthpy) (0.9.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
t-packages (from requests->earthpy) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
usr/local/lib/python3.7/dist-packages (from requests->earthpy) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
7/dist-packages (from requests->earthpy) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
.7/dist-packages (from requests->earthpy) (2020.12.5)
Requirement already satisfied: click-plugins in /usr/local/lib/python3.7/di
st-packages (from rasterio->earthpy) (1.1.1)
Requirement already satisfied: attrs in /usr/local/lib/python3.7/dist-packa
ges (from rasterio->earthpy) (20.3.0)
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.7/di
st-packages (from rasterio->earthpy) (7.1.2)
Requirement already satisfied: affine in /usr/local/lib/python3.7/dist-pack
ages (from rasterio->earthpy) (2.3.0)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.7/dist-
packages (from rasterio->earthpy) (0.7.1)
Requirement already satisfied: snuggs>=1.4.1 in /usr/local/lib/python3.7/di
st-packages (from rasterio->earthpy) (1.4.7)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.7/di
st-packages (from scikit-image->earthpy) (1.4.1)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.7/di
st-packages (from scikit-image->earthpy) (7.0.0)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/d
ist-packages (from scikit-image->earthpy) (2.4.1)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/di
st-packages (from scikit-image->earthpy) (2.5)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.
7/dist-packages (from scikit-image->earthpy) (1.1.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dis
t-packages (from matplotlib>=2.0.0->earthpy) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib>=2.0.0->earthpy) (2
.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
```

```
7/dist-packages (from matplotlib>=2.0.0->earthpy) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/pytho
n3.7/dist-packages (from matplotlib>=2.0.0->earthpy) (2.8.1)
Requirement already satisfied: shapely>=1.6 in /usr/local/lib/python3.7/dis
t-packages (from geopandas->earthpy) (1.7.1)
Requirement already satisfied: fiona>=1.8 in /usr/local/lib/python3.7/dist-
packages (from geopandas->earthpy) (1.8.18)
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/d
ist-packages (from geopandas->earthpy) (1.1.5)
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.7/di
st-packages (from geopandas->earthpy) (3.0.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.7
/dist-packages (from networkx>=2.0->scikit-image->earthpy) (4.4.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-package
s (from cycler>=0.10->matplotlib>=2.0.0->earthpy) (1.15.0)
Requirement already satisfied: munch in /usr/local/lib/python3.7/dist-packa
ges (from fiona>=1.8->geopandas->earthpy) (2.5.0)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dis
t-packages (from pandas>=0.24.0->geopandas->earthpy) (2018.9)
```

Australian wildfires

The entire world attention was drawn to Australia in the early days of 2020, where country's endangered species struggled for survival in the face of catastrophic fires.Koalas drinking from firefighters' water bottle image went viral, making them the global face of a tragedy that could kill three billion animals.The fire season, nicknamed "Black Summer," resulted in more wildlife deaths and near-extinctions than any other single occurrence in Australian history. Australia's Black Summer serves as a stark warning to the international community as global warming accelerates and destructive wildfires become more frequent. However, my research focused on Australian wildfires, specifically in and around Canberra, using Sentinel-2, level-2 data. I have acquired pre fire data Jan'2020 and post fire data March'2020. In this analysis, I will be analysising the satellite images with some aspects of pre-fire and post-fire as follows

1. Vegetation and soil indices
2. Water indices
3. Geology indices

Finally, I will be calculating the Normalised Burn Ratio (NBR) for pre-fire and post-fire and find the diffrences between pre-fire and post-fire NBR and classify according to burn severity.

Sentinal-2

The sentinel-2A was launched in 2015 and second version sentinel-2B was launched in 2017 stationed at sun-synchronous orbit. Its objective is to keep track of changes in land

surface conditions with 10 days repeat cycle. Every five days together, they cover all of the Earth's land surfaces, vast islands, and inland and coastal waters.

Sentinal-2 spectral band has the 13 bands as follows

	Spectral Band	Centre Wavelength (nm)	Band Width (nm)	Spatial Resolution (nm)
B1	Coastal aerosol	443	20	60
B2	Blue (B)	490	65	10
B3	Green (G) [1]	560	35	10
B4	Red (R) [1]	665	30	10
B5	Red-edge 1 (Re1) [1]	705	15	20
B6	Red-edge 2 (Re2) [1]	740	15	20
B7	Red-edge 3 (Re3) [1]	783	20	20
B8	Near infrared (NIR) [1]	842	115	10
B8a	Near infrared narrow (NIRn) [1]	865	20	20
B9	Water vapor	945	20	60
B10	Shortwave infrared/Cirrus	1380	30	60
B11	Shortwave infrared 1 (SWIR1)	1910	90	20
B12	Shortwave infrared 2 (SWIR2)	2190	180	20

image source : usgs

Pre-Fire Data

The pre-fire data was acquired in January'2020, The below code was used to read the data by glob( ) and the layers are stacked using numpy.stack( ) with three-dimensional array.

In [5]:
```
# open prefire band-8 as separate single-band raster band
imagePath = '/content/drive/My Drive/coursework/S2A_MSIL2A_20200130T000231_
band8 = rio.open(imagePath+'T55HFV_20200130T000231_B08_10m.jp2', driver='JI
```

In [6]:
```
# Reproject the prefire band-8 in target path
import os
source_path = " -of jp2 " + "/content/drive/My Drive/coursework/S2A_MSIL2A_
target_path = "/content/drive/My Drive/coursework/S2A_MSIL2A_20200130T00023

source_proj = "\'+proj=utm +zone=19 +ellps=WGS84 +datum=WGS84 +units=m  \'
new_proj = "+proj=utm +zone=18 +south +ellps=WGS84 +datum=WGS84 +units=m "
target_proj = " -t_srs " + '\''+ new_proj + '\''

cmd = "gdalwarp -overwrite -s_srs " + \
                        source_proj + \
                        target_proj + \
                        source_path + \
                        " " + \
                        target_path

print(cmd)

os.system(cmd)
```

```
gdalwarp -overwrite -s_srs '+proj=utm +zone=19 +ellps=WGS84 +datum=WGS84 +u
nits=m  '  -t_srs '+proj=utm +zone=18 +south +ellps=WGS84 +datum=WGS84 +uni
ts=m ' -of jp2 /content/drive/My Drive/coursework/S2A_MSIL2A_20200130T00023
1_N0213_R030_T55HFV_20200130T020413.SAFE/GRANULE/L2A_T55HFV_A024051_2020013
0T000233/IMG_DATA/R10m/T55HFV_20200130T000231_B08_10m.jp2 /content/drive/My
Drive/coursework/S2A_MSIL2A_20200130T000231_N0213_R030_T55HFV_20200130T0204
13.SAFE/GRANULE/L2A_T55HFV_A024051_20200130T000233/IMG_DATA/R20m/T55HFV_202
00130T000231_B08_20m.jp2
```

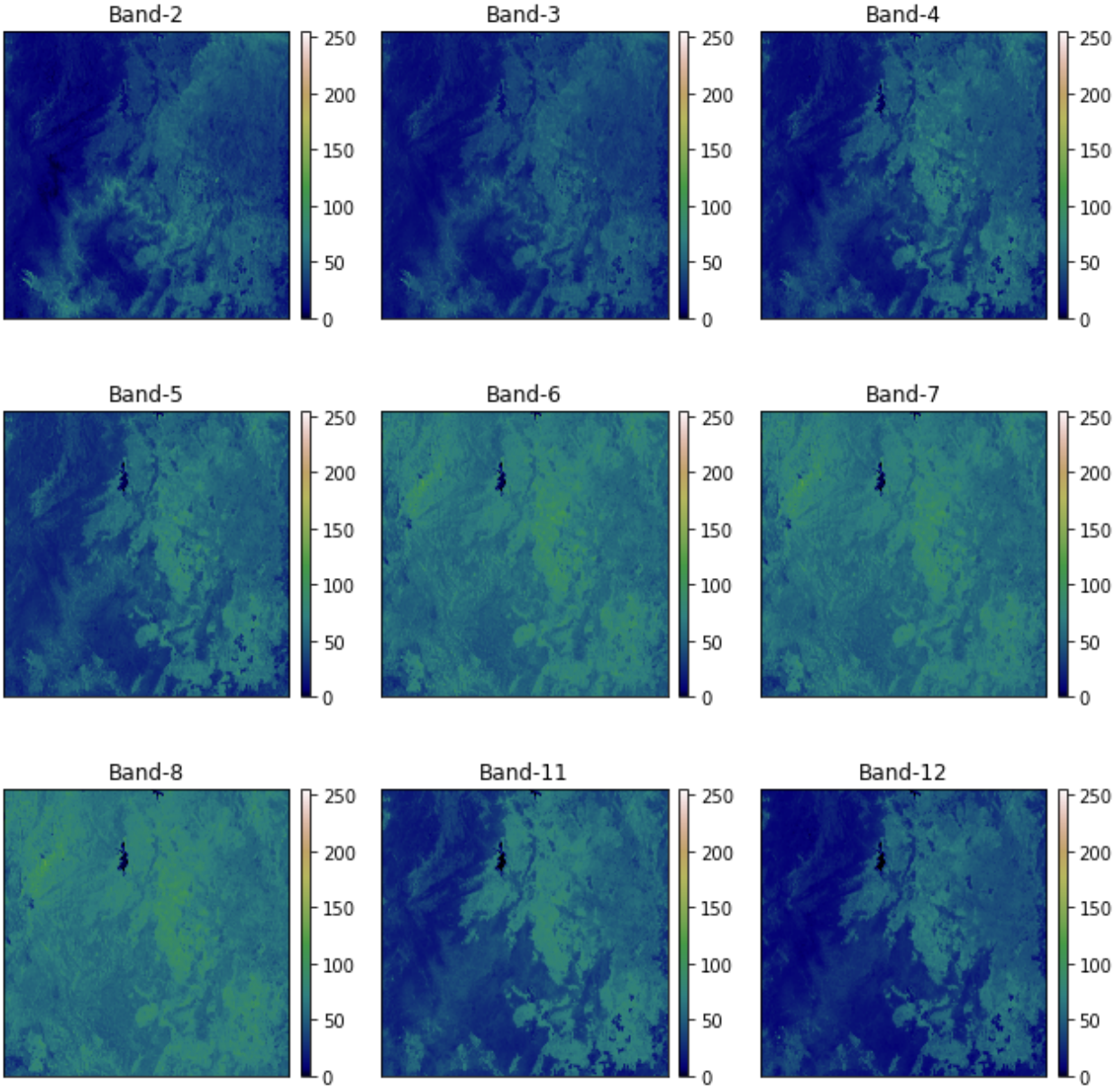
Out[6]:
```
512
```

In [7]:
```
# Import the pre-fire spectral bands using glob()

prefire_bands = glob("S2A_MSIL2A_20200130T000231_N0213_R030_T55HFV_2020013(
prefire_bands.sort()
prefire_bands = prefire_bands[:8] + [prefire_bands[-1]] + prefire_bands[8:-

# Stack Bands using numpy()
layer = []

for i in prefire_bands:
  with rio.open(i, 'r') as f:
    layer.append(f.read(1))

prefire_stack = np.stack(layer)

# Print spectral size and no. of bands stacked
print(f'Height: {prefire_stack.shape[1]}\nWidth: {prefire_stack.shape[2]}\n
```

```
Height: 5490
Width: 5490
Bands: 9
```

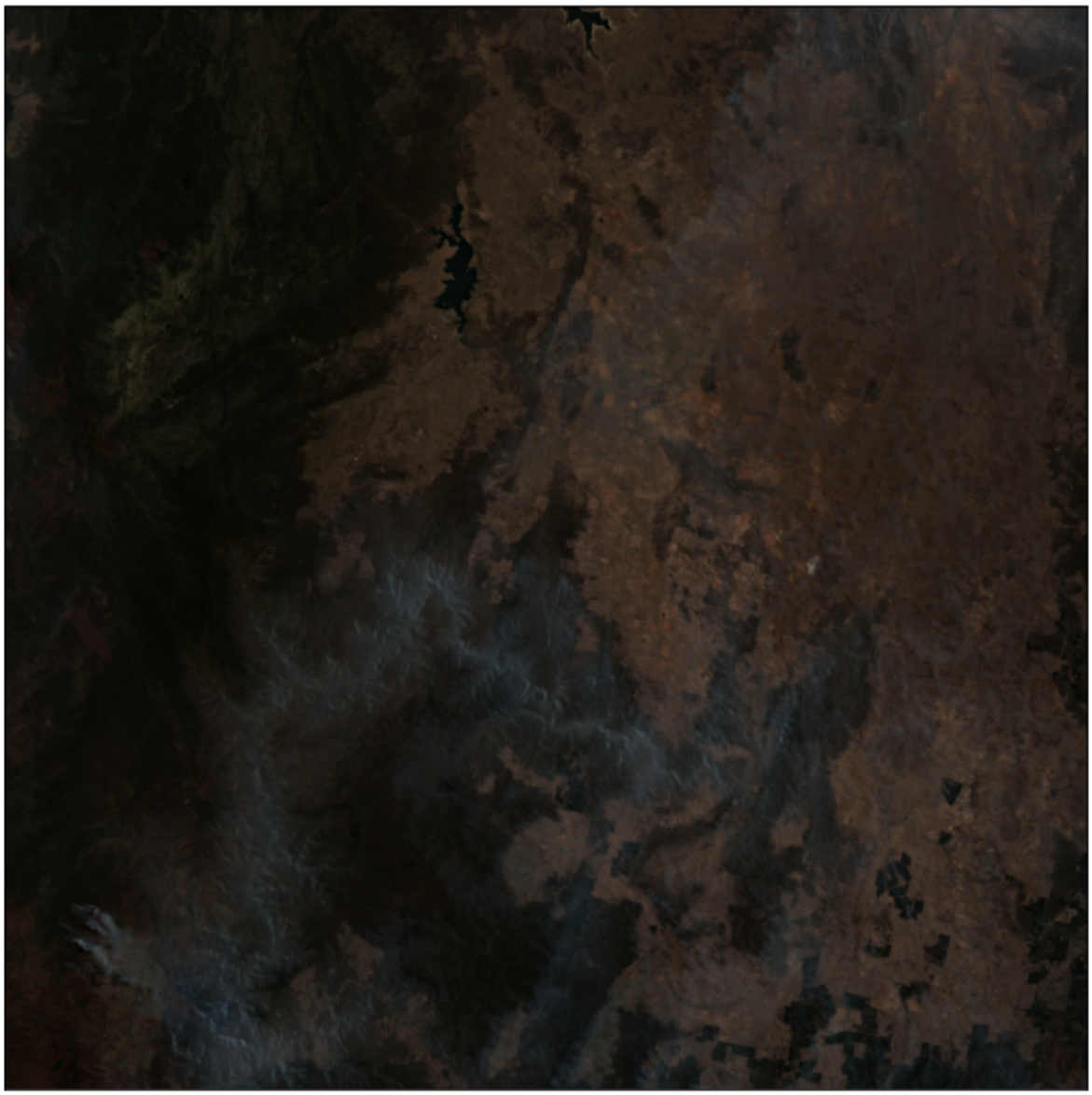
The below code will help us to visualise the pre-fire stacked layer using earthpy.plot_bands( ).

In [10]:
```
# Plot the spectral bands using earthpy()
# Here I have selected the only specific bands

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(prefire_stack,
              title=['Band-2', 'Band-3', 'Band-4', 'Band-5', 'Band-6', 'Ba
              cols=3,
              figsize=(9, 9),
              scale=True,
              cbar=True,
              cmap='gist_earth')

# show the band by matplotlib.pyplot()
plt.show()
```



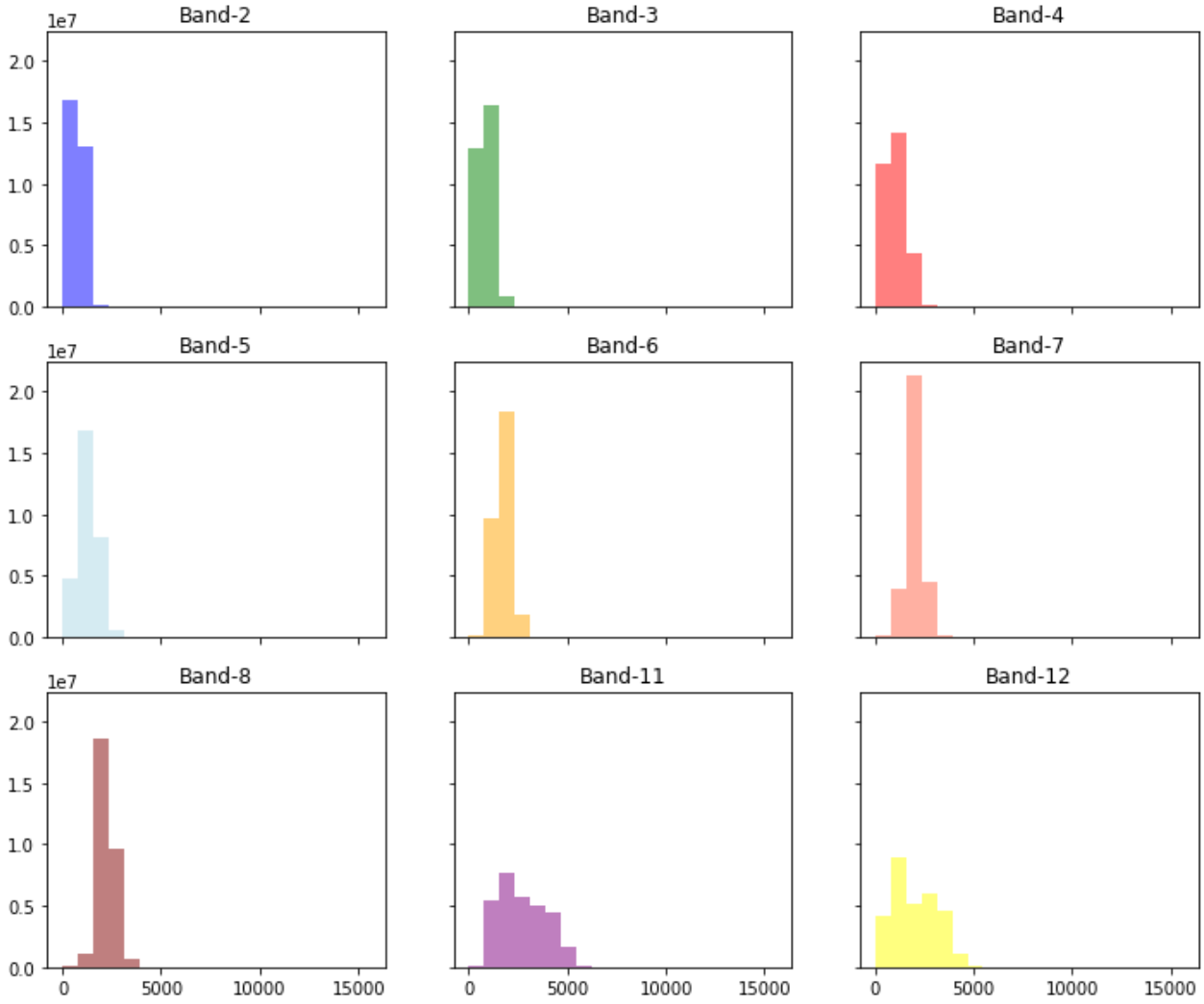
Let's visualise the natural color of the pre-fire satellite image using earthpy.plot_rgb( ).

In [11]:
```
# Visualise the RGB band composite of the image
# In stack array Red = 2, Green = 1, Blue = 0

# Plot the bands using earthpy.plot.plot_bands()
rgb = ep.plot_rgb(prefire_stack,
                  rgb=(2,1,0),
                  figsize=(10, 10))

# Show the band by matplotlib.pyplot()
plt.show()
```



In below code will help us to visualise the distribution of the data in the bands histogram using earthypy.hist( ).

In [12]:
```
# Pre-fire spectral band histogram

# Plot the histogram using earthpy.plot.hist()
ep.hist(prefire_stack,
        title=['Band-2', 'Band-3', 'Band-4', 'Band-5', 'Band-6', 'Band-7',
        colors=['blue', 'green', 'red', 'lightblue', 'orange', 'tomato','ma
        cols=3,
        alpha=0.5,
        figsize = (12, 10))

# show the histogram by matplotlib.pyplot()
plt.show()
```



We have imported and visualised the prefire data now, we will import postfire spectral bands. The pre-fire data was acquired in March'2021, The below code was used to read the data by glob( ) and the layers are stacked using numpy.stack( ) with three-dimensional array.

In [13]:
```
# open postfire band-8 as separate single-band raster band

imagePath = '/content/drive/My Drive/coursework/S2A_MSIL2A_20210315T000241_
band8 = rio.open(imagePath+'T55HFV_20210315T000241_B08_10m.jp2', driver='JI
```

In [14]:
```
# Reproject the postfire band-8 in target path

import os
source_path = " -of jp2 " + "/content/drive/My Drive/coursework/S2A_MSIL2A_
target_path = "/content/drive/My Drive/coursework/S2A_MSIL2A_20210315T0002

source_proj = "\'+proj=utm +zone=19 +ellps=WGS84 +datum=WGS84 +units=m  \'
new_proj = "+proj=utm +zone=18 +south +ellps=WGS84 +datum=WGS84 +units=m "
target_proj = " -t_srs " + '\''+ new_proj + '\''

cmd = "gdalwarp -overwrite -s_srs " + \
                        source_proj + \
                        target_proj + \
                        source_path + \
                        " " + \
                        target_path

print(cmd)

os.system(cmd)
```

```
gdalwarp -overwrite -s_srs '+proj=utm +zone=19 +ellps=WGS84 +datum=WGS84 +u
nits=m  '  -t_srs '+proj=utm +zone=18 +south +ellps=WGS84 +datum=WGS84 +uni
ts=m ' -of jp2 /content/drive/My Drive/coursework/S2A_MSIL2A_20210315T00024
1_N0214_R030_T55HFV_20210315T020346.SAFE/GRANULE/L2A_T55HFV_A029914_2021031
5T000240/IMG_DATA/R10m/T55HFV_20210315T000241_B08_10m.jp2 /content/drive/My
Drive/coursework/S2A_MSIL2A_20210315T000241_N0214_R030_T55HFV_20210315T0203
46.SAFE/GRANULE/L2A_T55HFV_A029914_20210315T000240/IMG_DATA/R20m/T55HFV_202
10315T000241_B08_20m.jp2
```

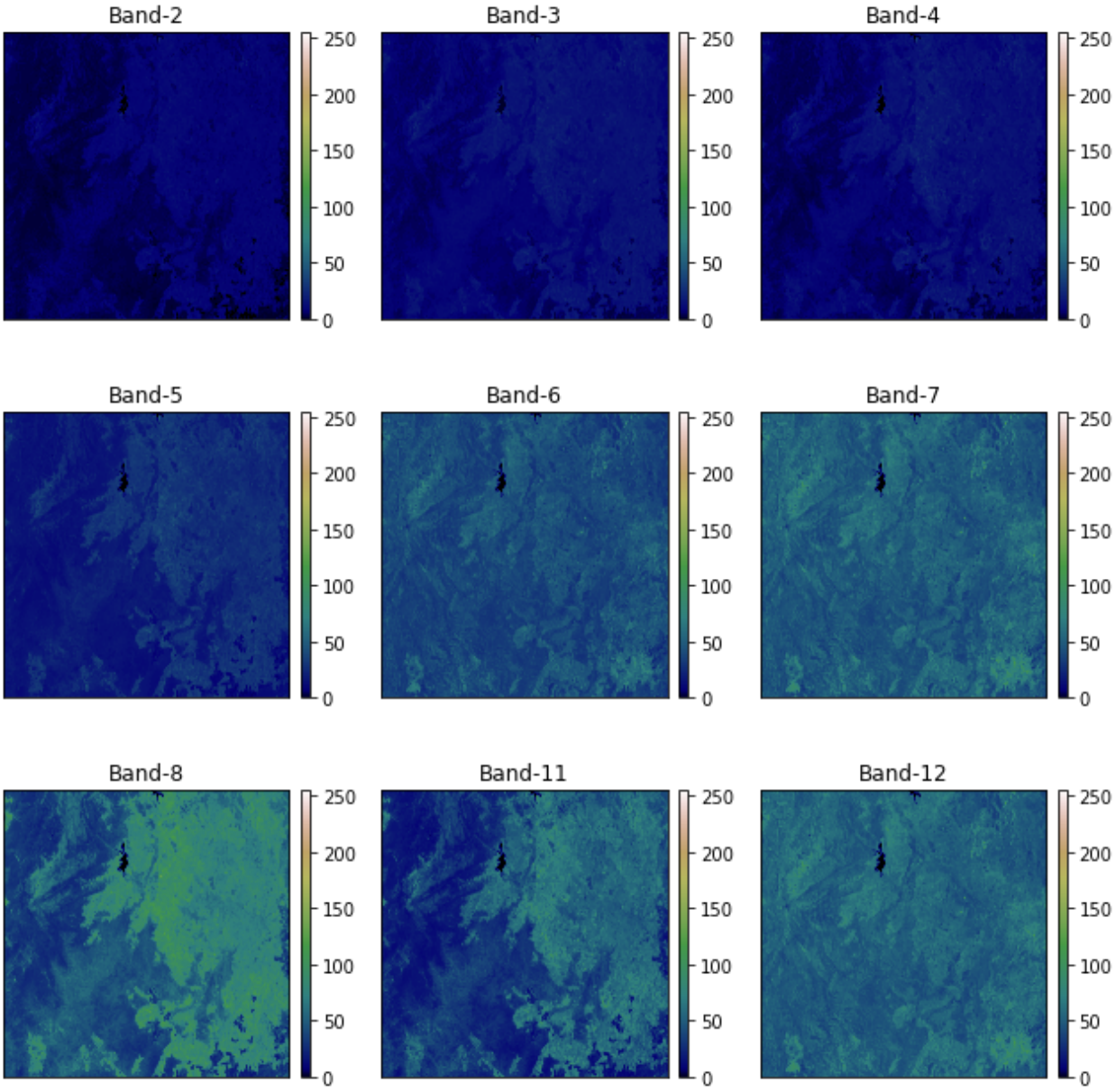
Out[14]: 512

In [15]:
```
# Import the post-fire spectral bands using glob()
postfire_bands = glob("S2A_MSIL2A_20210315T000241_N0214_R030_T55HFV_202103
postfire_bands.sort()
postfire_bands = postfire_bands[:8] + [postfire_bands[-1]] + postfire_bands

# Stack Bands using numpy()
layer = []

for i in postfire_bands:
  with rio.open(i, 'r') as f:
    layer.append(f.read(1))

postfire_stack = np.stack(layer)

# Print spectral size and no. of bands stacked
print(f'Height: {postfire_stack.shape[1]}\nWidth: {postfire_stack.shape[2]}
```

```
Height: 5490
Width: 5490
Bands: 9
```

The below code will help us to visualise the post-firestacked layer using earthpy.plot_bands( ).

In [16]:
```
# Plot the spectral bands using earthpy
# I have selected the only specific bands

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(postfire_stack,
              title=['Band-2', 'Band-3', 'Band-4', 'Band-5', 'Band-6', 'Ba
              cols=3,
              figsize=(9, 9),
              scale=True,
              cbar=True,
              cmap='gist_earth')

# show the band by matplotlib.pyplot()
plt.show()
```

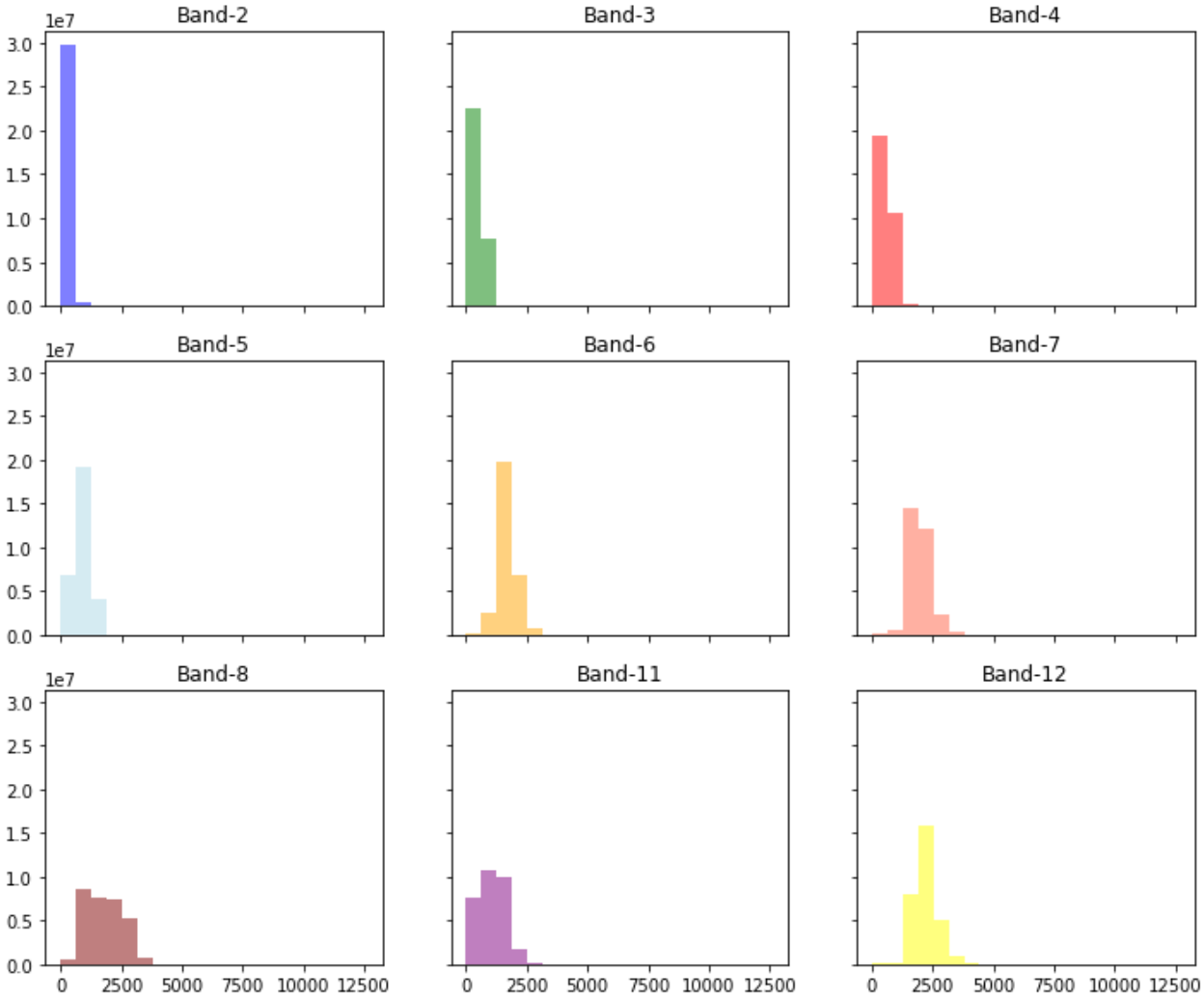


Let's visualise the natural color of the post-fire satellite image using earthpy.plot_rgb( ).

In [17]:
```
# Visualise the RGB band composite of the image
# In stack array Red = 2, Green = 1, Blue = 0

# Plot the bands using earthpy.plot.plot_bands()
rgb = ep.plot_rgb(postfire_stack,
                  rgb=(2,1,0),
                  figsize=(10, 10))

# Show the band by matplotlib.pyplot()
plt.show()
```

In below code will help us to visualise the distribution of the data in the bands histogram using earthypy.hist( ).

In [18]:
```
# Post-fire spectral band histogram

# Plot the histogram using earthpy.plot.hist()
ep.hist(postfire_stack,
        title=['Band-2', 'Band-3', 'Band-4', 'Band-5', 'Band-6', 'Band-7',
        colors=['blue', 'green', 'red', 'lightblue', 'orange', 'tomato','ma
        cols=3,
        alpha=0.5,
        figsize = (12, 10))

# Plot the histogram by matplotlib.pyplot()
plt.show()
```



In above analysis, we have imported, reprojected, visualised the spectral bands individually & Natural color and the Histogram for the both Pre-fire and Post-fire data. Now, we will analysis the some indices like soil and vegetation, water and the geology. By analysing this we can more precision on the factors affected by the fires in Australia.

Soil and Vegetation Indices

Dense vegetation will be brighter in the index picture, while poor vegetation will have smaller value and desert landscape will be dark. Although the intensity of images is affected by shading from landscape variance (hills and valleys), the indices are designed such that the colour of a subject is highlighted instead of the intensity or brightness of the object.

Normalized difference vegetation index (NDVI)

The Normalized Difference Vegetation Index (NDVI) measures the difference between near-infrared light, which vegetation strongly reflects, and red light, which vegetation strongly reflects. The NDVI value is always between -1 and 1.

The sentinel-2 NDVI will calculate by the below formula

NDVI = ((NIR - Red)/(NIR + Red))

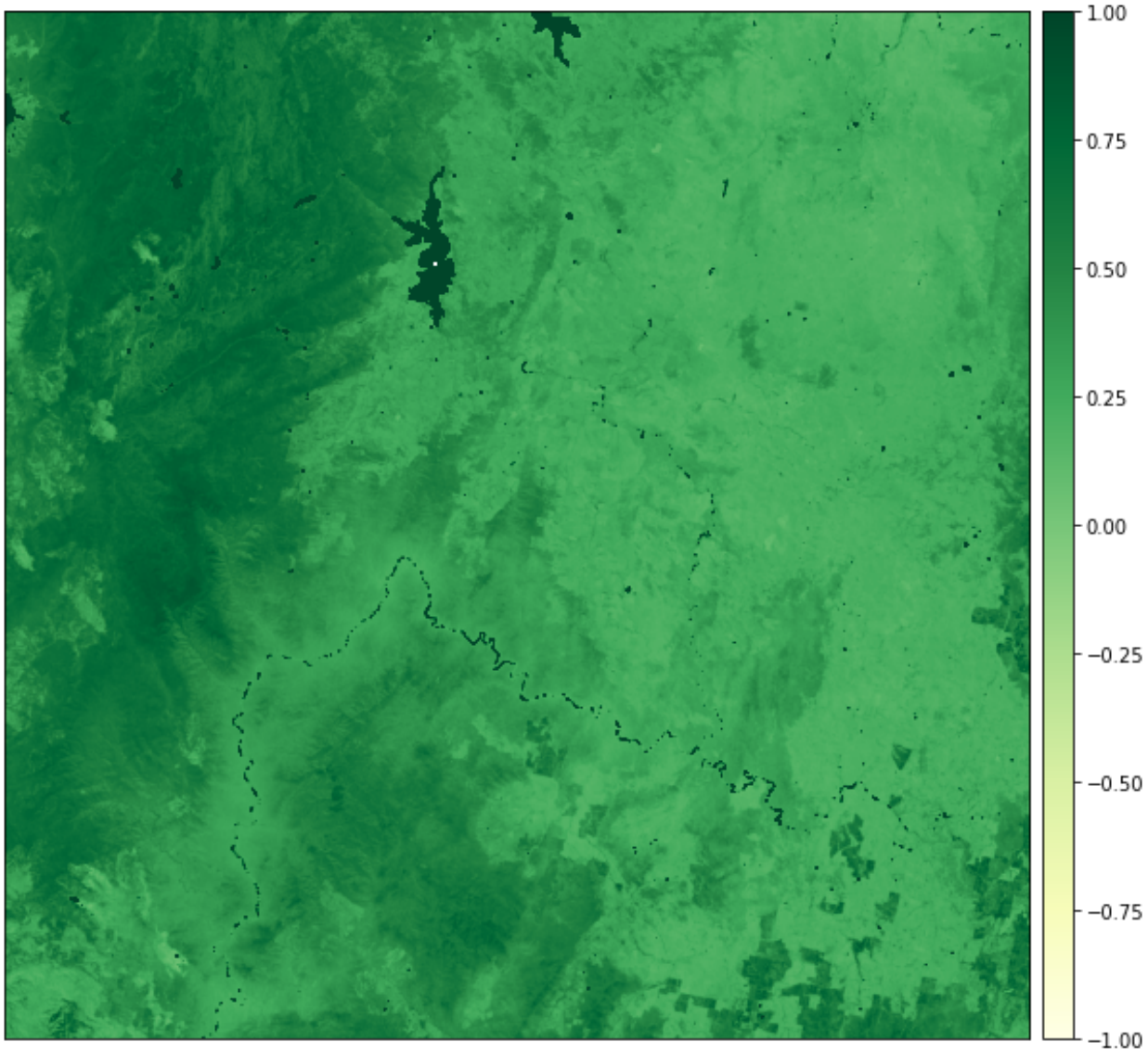
NIR - Near Infrared (Band - 8)

Red - Red band (Band - 4)

Water is most likely when the values are negative. If your NDVI value is close to +1, however, it's likely that you're looking at dense green leaves. When the NDVI is close to zero, nevertheless, there are no green leaves and the area can be urbanised.

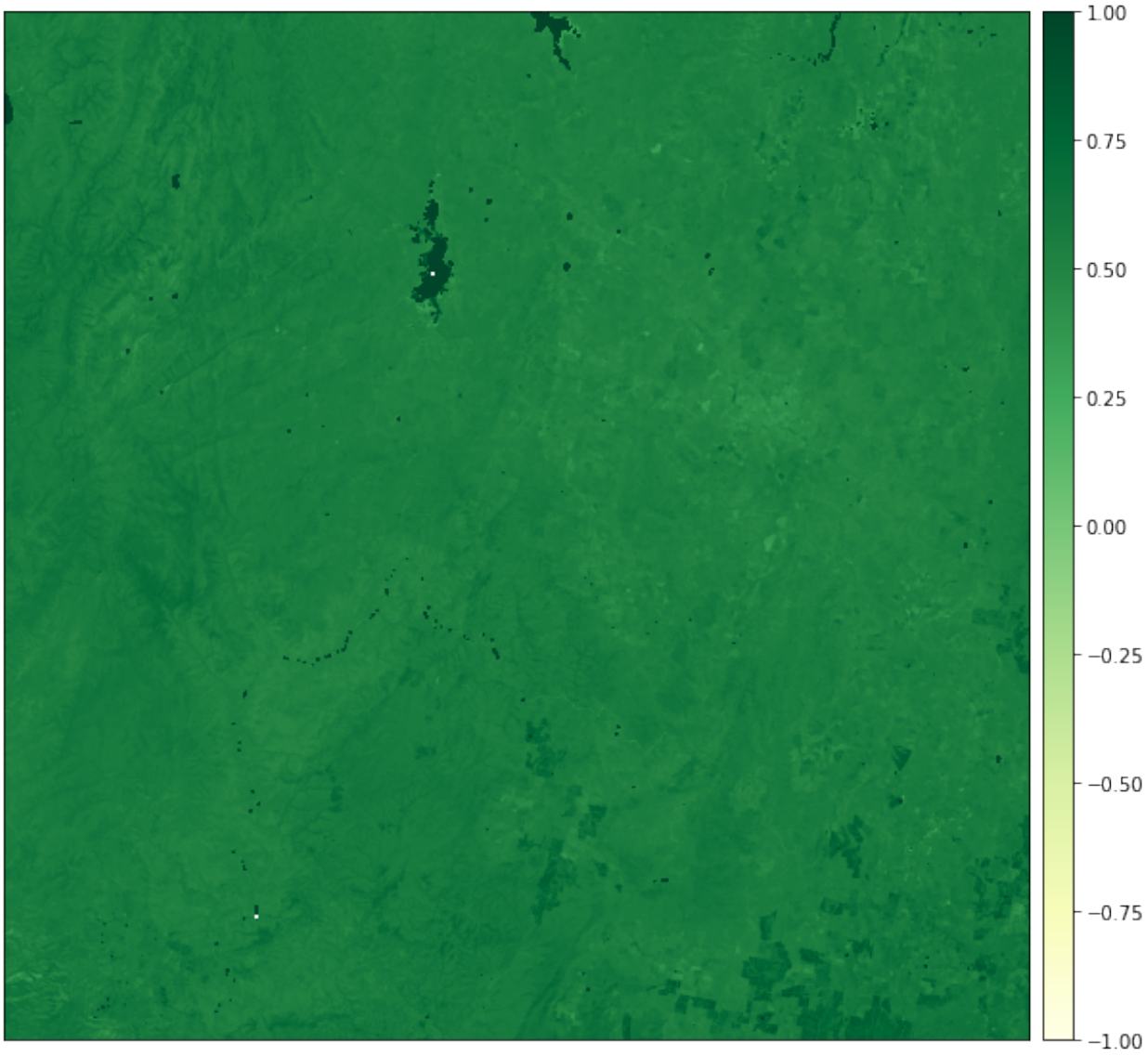
Let's see the NDVI on pre-fire satellite image.

In [38]:
```
# Calculate the NDVI using earthypy.spatial.normalised_diff() for prefire
# The band used here is band 8 and band 4
ndvi = es.normalized_diff(prefire_stack[6], prefire_stack[2])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(ndvi,
              cmap="YlGn",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let's see NDVI on post-fire satellite image.

In [39]:
```
# Calculate the NDVI using earthypy.spatial.normalised_diff() for postfire
# The band used here is band 8 and band 4

ndvi = es.normalized_diff(postfire_stack[6], postfire_stack[2])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(ndvi,
              cmap="YlGn",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

After the comparisons of the both the results, the fire that leads to some loss of the vegetation and some areas after these year we can see some vegetation over the regions.

Soil-Adjusted Vegetation Index (SAVI)

The Soil-Adjusted Vegetation Index (SAVI) is a vegetation index that uses a soil-brightness adjustment factor to try to reduce soil brightness influences. This is widely used in dry areas with little vegetation.

The sentinel - 2 SAVI calulations as follows.

SAVI = ((NIR - Red) / (NIR + Red + L)) x (1 + L)

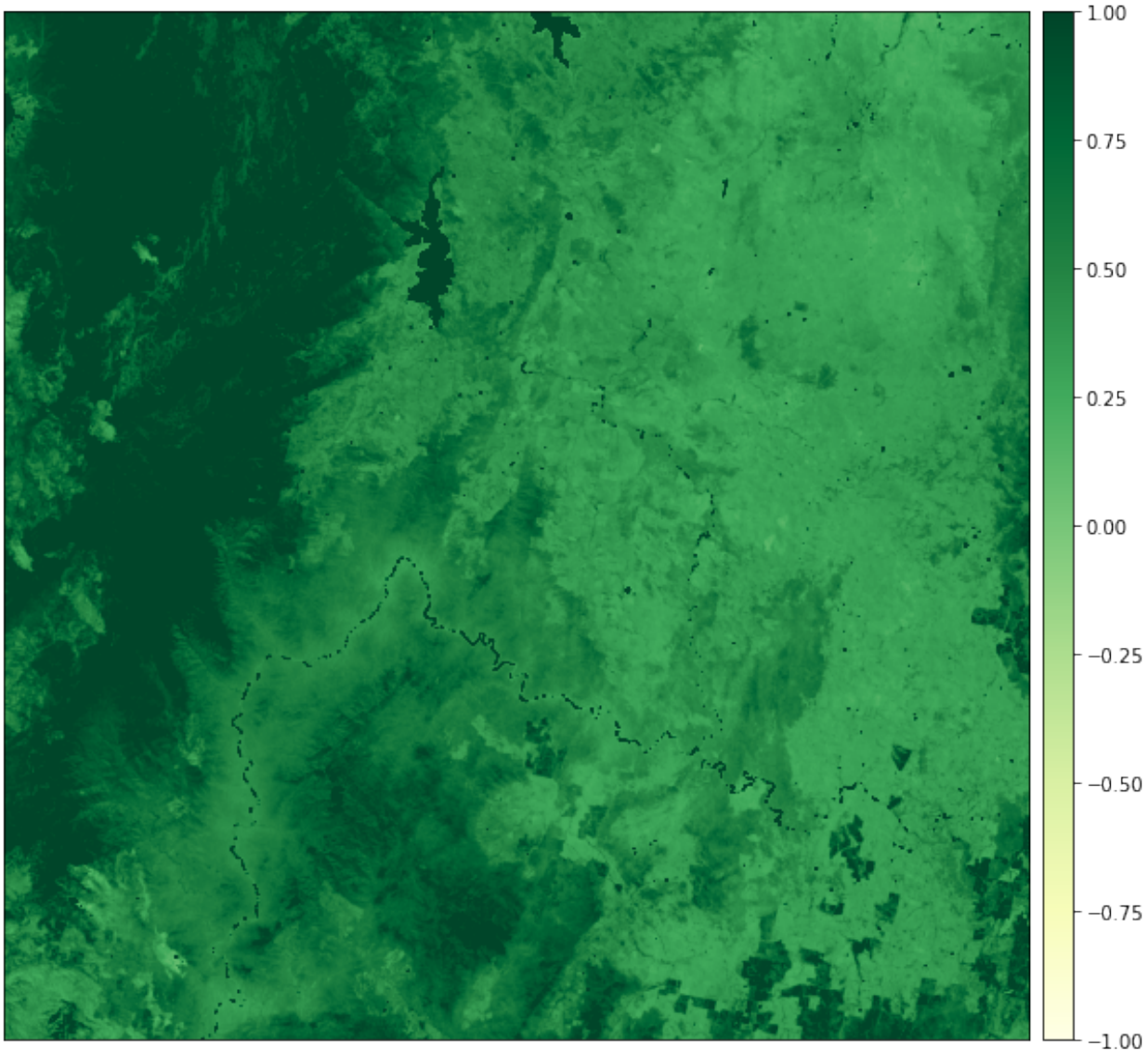
NIR - Near Infrared (Band - 8)

Red - Red band (Band - 4)

The L value varies depending on how much green vegetation is present. L=1 in areas where there is no green vegetation cover, L=0.5 in areas where there is low green vegetation cover, and L=0 in areas where there is very high vegetation cover (which is equivalent to the NDVI method). This index returns values ranging from -1.0 to 1.0

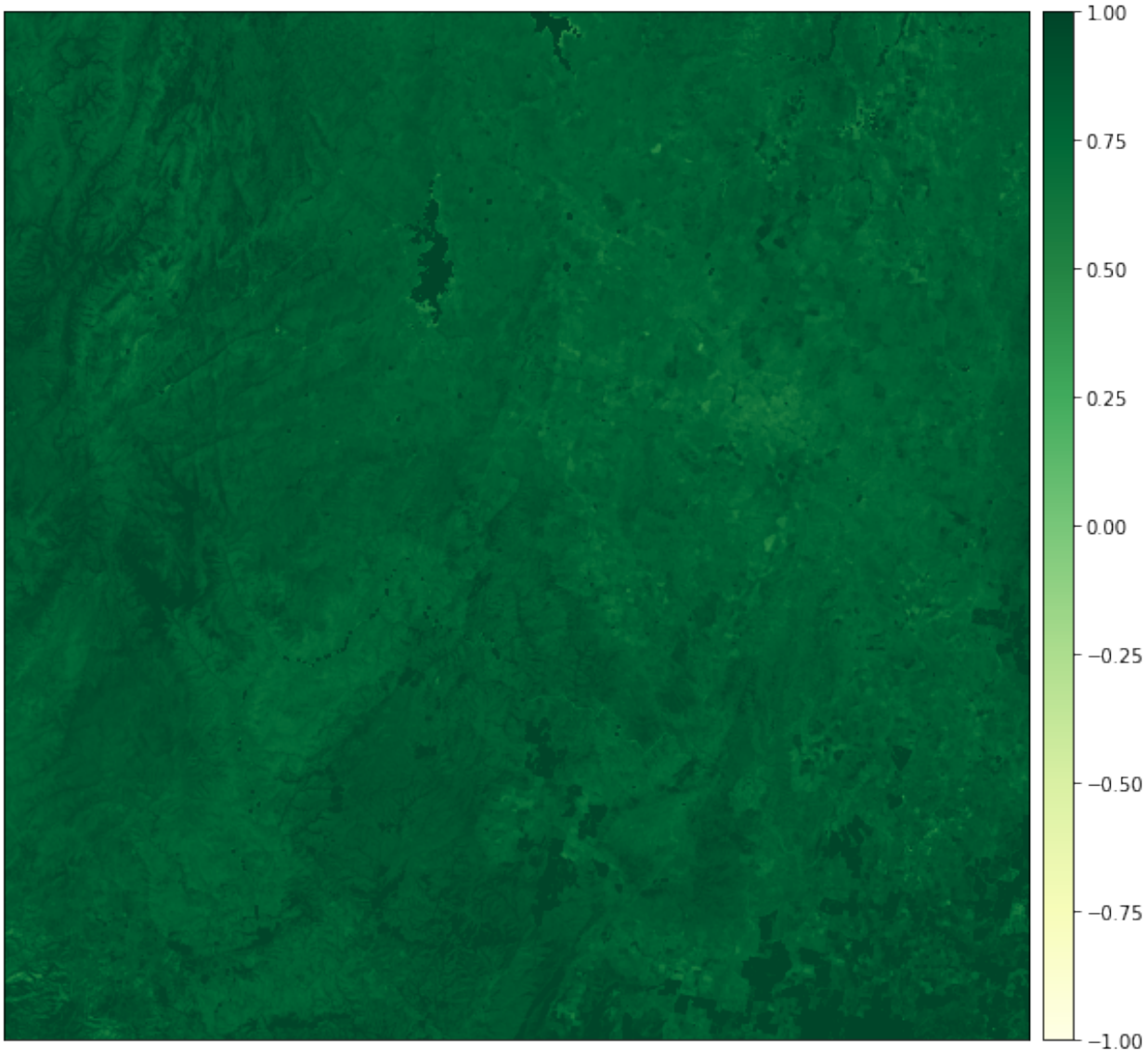
Let us calculate the SAVI for the pre-fire image

In [37]:
```
# Calculate the SAVI for prefire statellite images
# The band used here is band 8 and band 4
# Here, we use L = 0.5

savi = ((prefire_stack[6] - prefire_stack[2]) / (prefire_stack[6] + prefire

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(savi,
              cmap="YlGn",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let's Calculate the SAVI for postfire data.

In [40]:
```
# Calculate the SAVI for postfire statellite image
# The band used here is band 8 and band 4
# Here, we use L = 0.5

savi = ((postfire_stack[6] - postfire_stack[2]) / (postfire_stack[6] + post

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(savi,
              cmap="YlGn",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

The comparison between prefire and postfire SAVI images, we can clearly see that some areas it has lost the vegetation, in some places, it has gained some vegetaion in March'2021 after those wildfires.

Visible Atmospherically Resistant Index (VARI)

The Visible Atmospheric Resistant Index (VARI) is intended to highlight vegetation in the visible spectrum while minimising illumination variations and atmospheric effects. It works well for RGB or colour images since it employs all three colour bands.

The VARI will calculate as follows,

VARI = (Green - Red)/ (Green + Red - Blue)

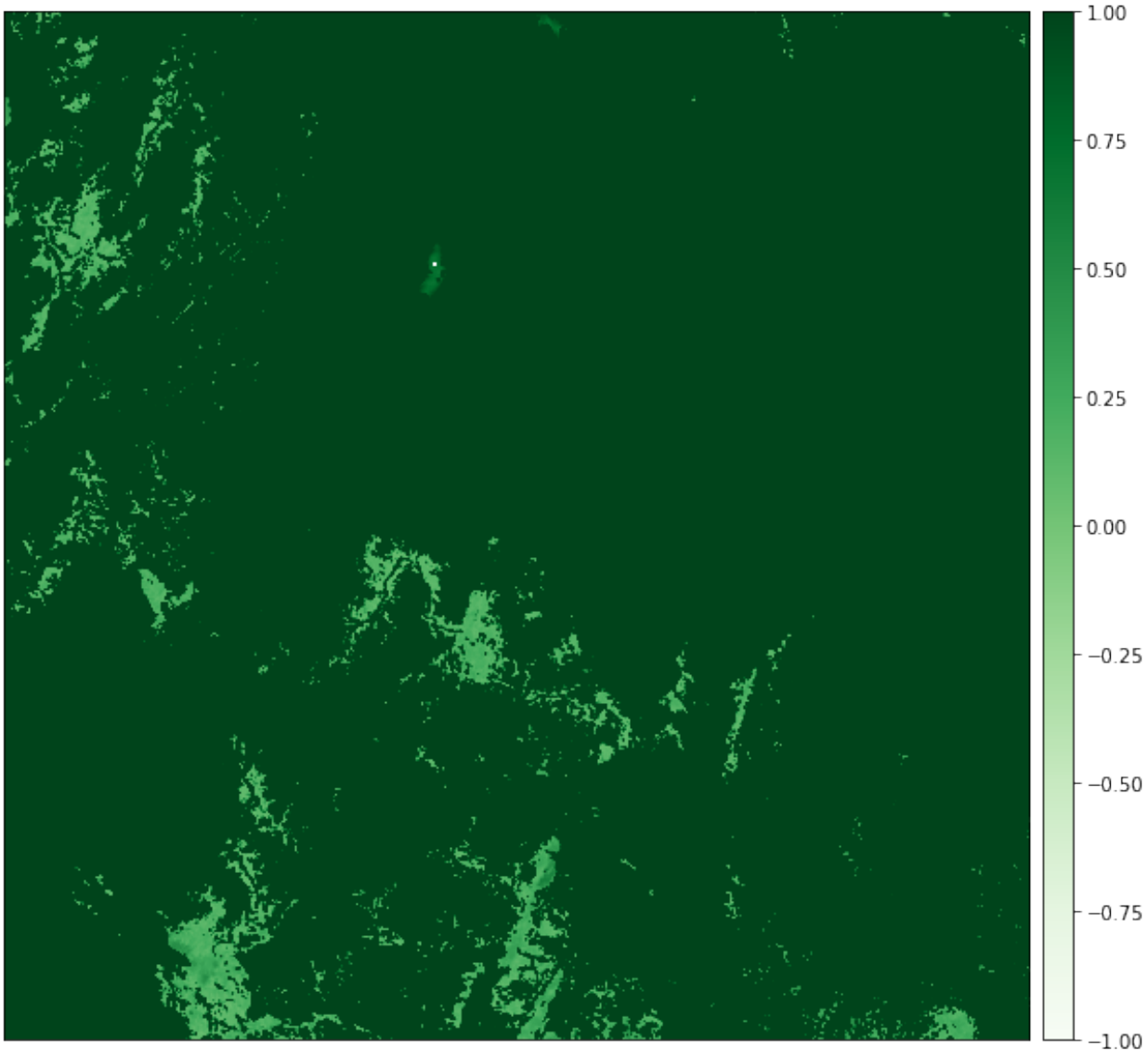
Green - Green Band (Band - 3)

Red - Red Band (Band - 4)

Blue - Blue Band (Band - 2)

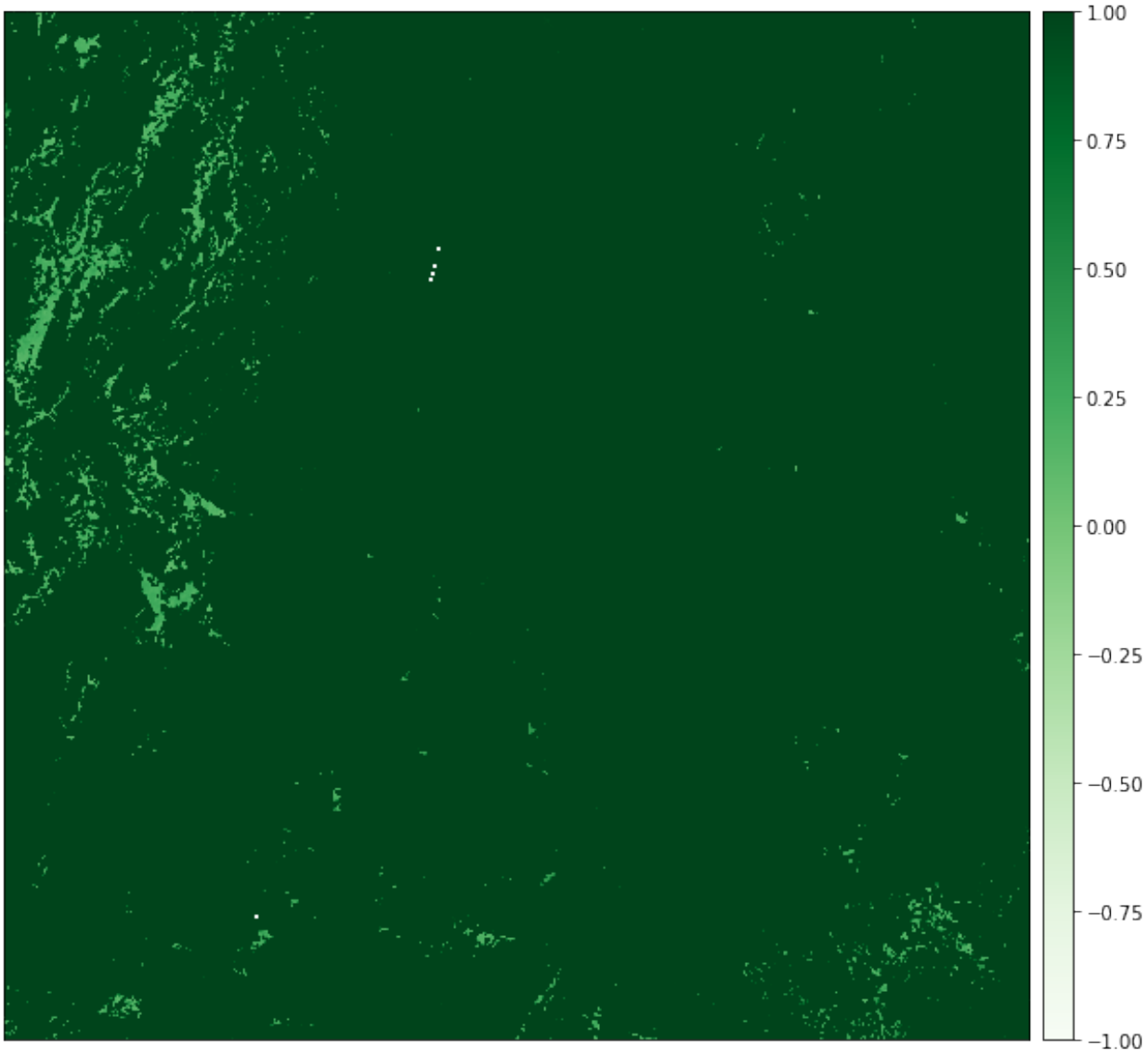
Let's see the VARI for the pre-fire satellite images

In [49]:
```
# Calculate the VARI for prefire statellite image
# The band used here is band 2, 3 and 4
vari = (prefire_stack[1] - prefire_stack[2])/ (prefire_stack[1] + prefire_s

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(vari,
              cmap="Greens",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let's Calculate the VARI for post fire data.

In [48]:
```
# Calculate the VARI for prefire statellite image
# The band used here is band 2, 3 and 4
vari = (postfire_stack[1] - postfire_stack[2])/ (postfire_stack[1] + postf

# Plot the data using earthpy.plot.plot_bands()
ep.plot_bands(vari,
              cmap="Greens",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

After the comparison of the these images, it shows the vegetation has been gained over these years. In March'2021, It gaining the vegetation which is lost for the fires. Now, we will move on to the water indices.

Water Indices

Changes in surface water are a primary indicator of environmental, climatic, and anthropogenic influences. There are many techniques for the surface water extraction, amongst the index-based methods are very popular and cost effectiveness.

Modified Normalized Difference Water Index (MNDWI)

For the enhancement of open water properties, the Modified Normalized Difference Water Index (MNDWI) employs green and SWIR bands. It also reduces built-up area characteristics, which are often linked to open water in other indices.

MNDWI will calculated as follows:

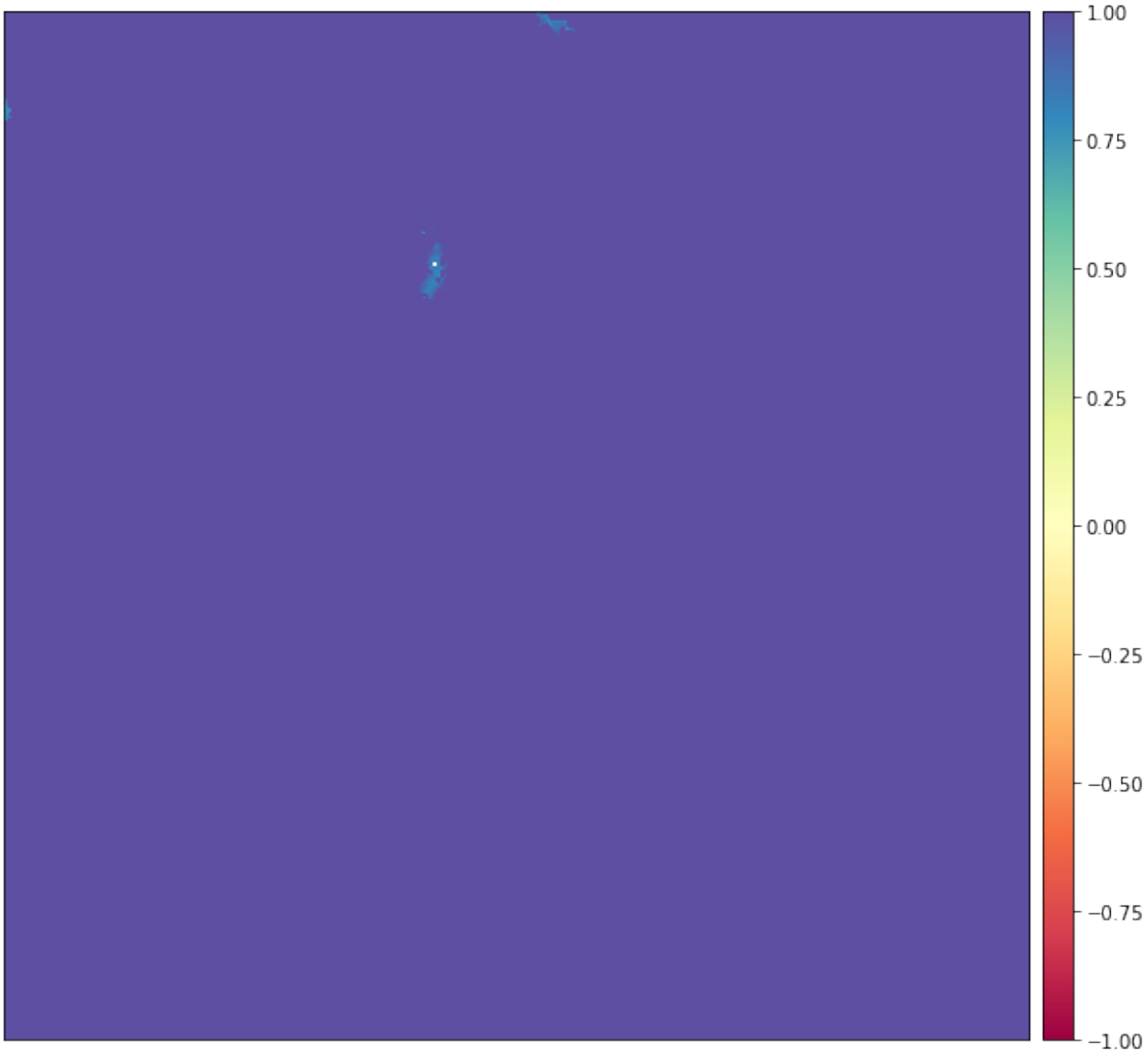
MNDWI = (Green - SWIR1) / (Green + SWIR1)

Green - Green band (Band - 3)

SWIR1 - Shortwave Infrared (Band - 11)

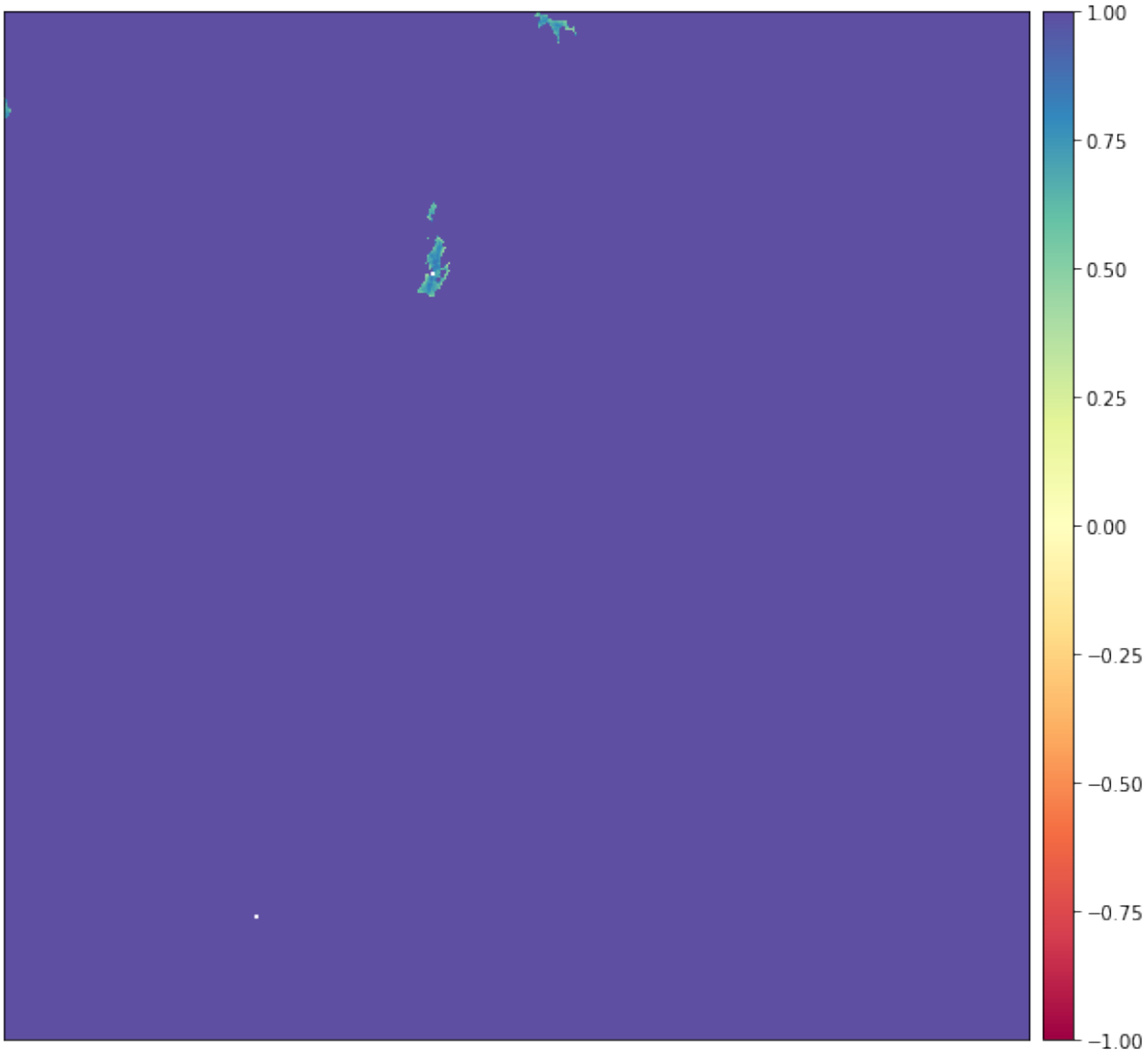
Let's us calculate the MNDWI for pre fire satellite in below

In [65]:
```
# Calculate the MNDWI using earthypy.spatial.normalised_diff() for prefire
# The band used here is band 3 and band 11

mndwi = es.normalized_diff(prefire_stack[1], prefire_stack[7])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(mndwi,
              cmap="Spectral",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let's us see for the postfire images.

In [56]:
```
# Calculate the MNDWI using earthypy.spatial.normalised_diff() for postfir
# The band used here is band 3 and band 11
mndwi = es.normalized_diff(postfire_stack[1], postfire_stack[7])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(mndwi,
              cmap="Spectral",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

The comparison between these two images, Its shows that in some areas they have retained the water resources, in some areas they lost the water level due to fires.

Normalized Difference Moisture Index (NDMI)

The Normalized Difference Moisture Index (NDMI) is a measure of how sensitive vegetation is to moisture levels. It's used to keep track of droughts and fuel levels in fire-prone areas. It produces a ratio that mitigates illumination and atmospheric effects by mixing NIR and SWIR bands.

The calculation for the NDMI as follows,

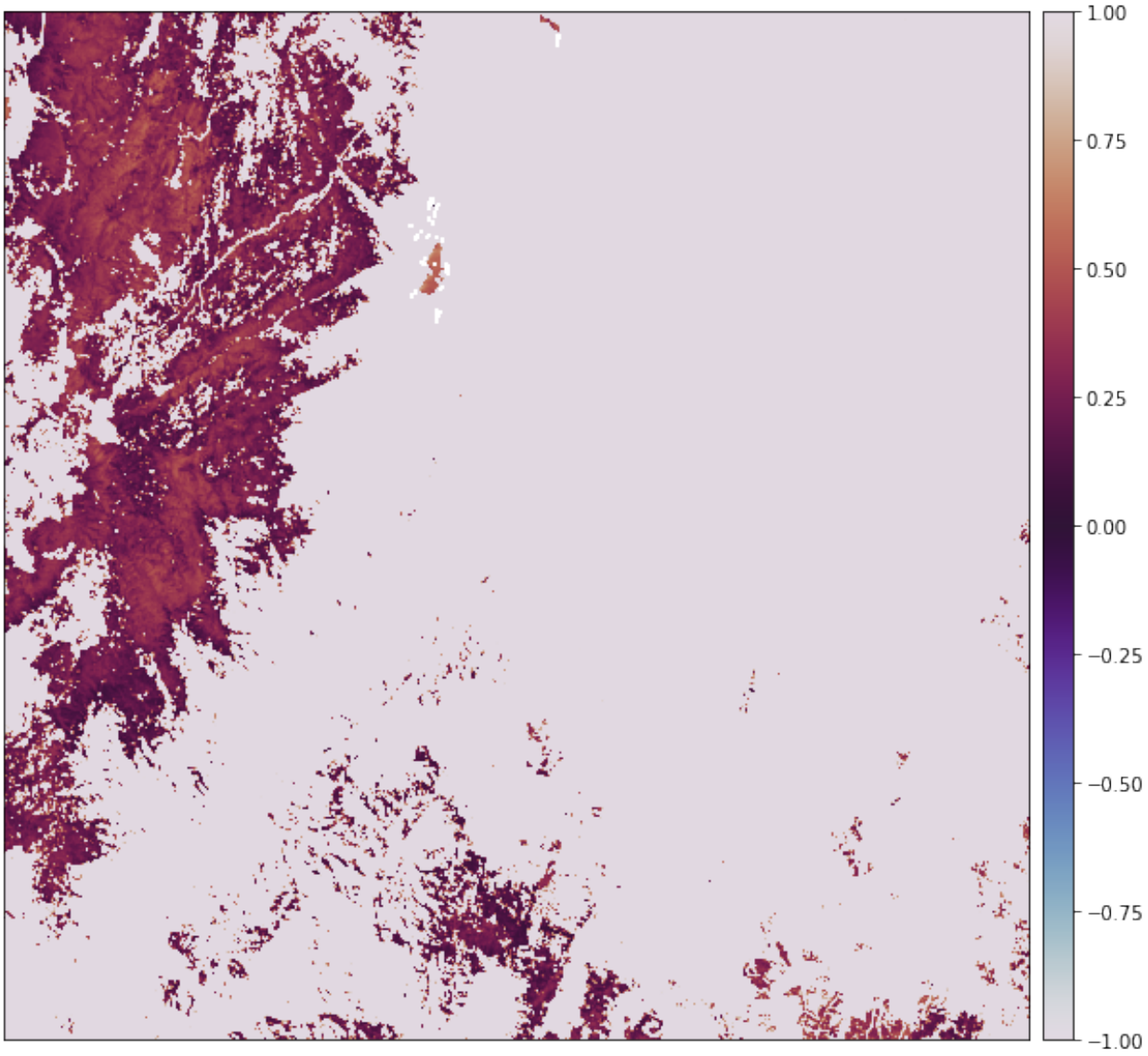
NDMI = (NIR - SWIR1)/(NIR + SWIR1)

NIR - Near Infrared (Band-8)

SWIR1 - Shortwave Infrared (Band-11)

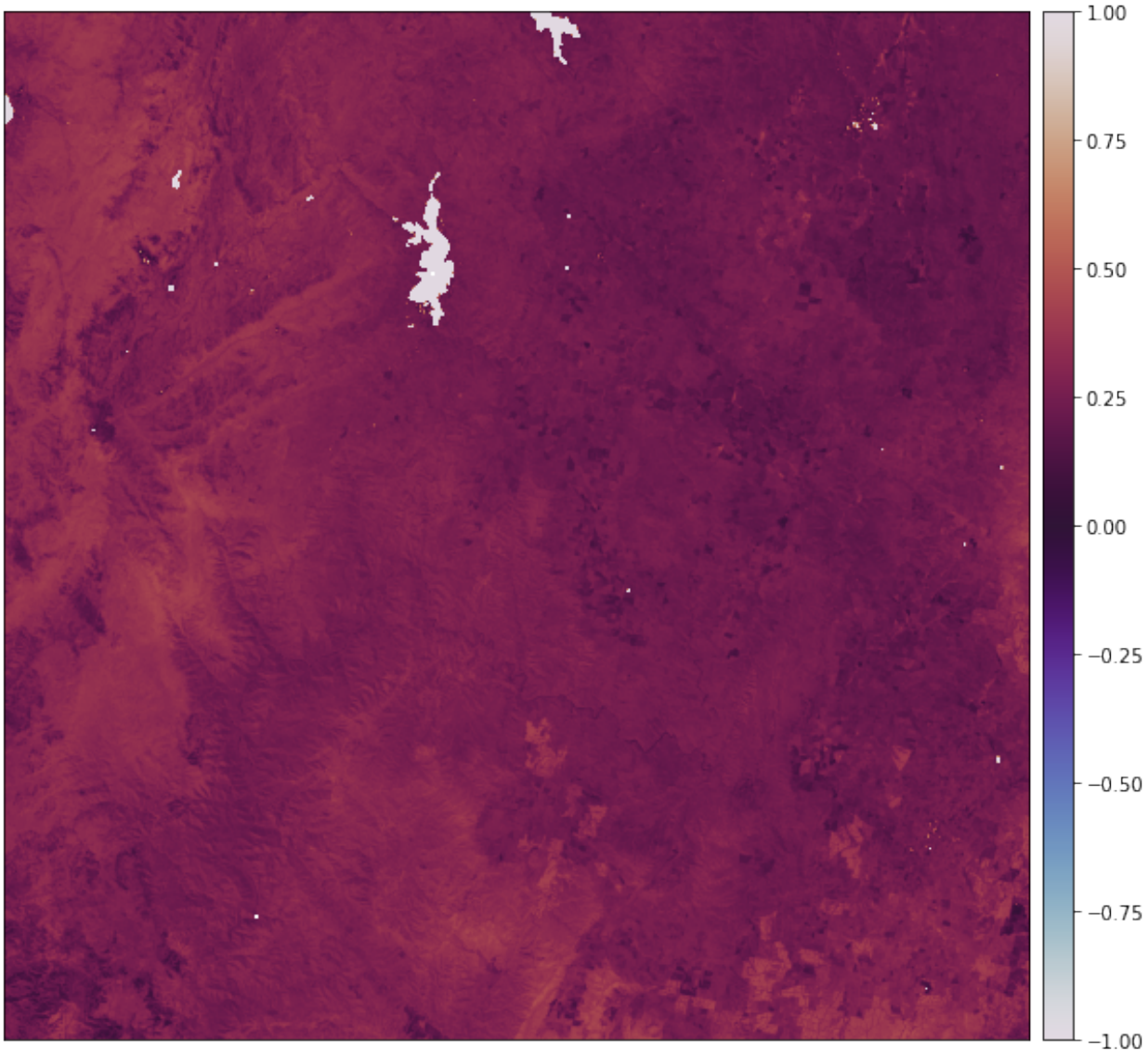
Let us see the NDWI calculation for the Pre-fire images.

In [61]:
```
# Calculate the MNDWI using earthypy.spatial.normalised_diff() for pre-fir
# The band used here is band 8 and band 11

ndmi = es.normalized_diff(prefire_stack[6], prefire_stack[7])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(ndmi,
              cmap="twilight",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



In below code, we will calculate the NDWI for the post-fire data.

In [62]:
```
# Calculate the MNDWI using earthypy.spatial.normalised_diff() for post-fi
# The band used here is band 8 and band 11

ndmi = es.normalized_diff(postfire_stack[6], postfire_stack[7])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(ndmi,
              cmap="twilight",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

In above two images, clearly shows that it has lost moisture in some areas, in some areas they have gained moisture too. Now, we move on to geological indices.

Geology Indices

Aerial photography and satellite imagery have also proved to be valuable methods in the support of mineral exploration projects. They can be used in a number of different ways.

Clay Minerals Ratio (CMR)

The clay ratio is calculated by multiplying the SWIR1 and SWIR2 bands together. This ratio takes advantage of the fact that hydrous minerals like clays and alunite absorb radiation in the 2.0–2.3 micron range. Since it is a ratio, this index mitigates illumination changes caused by terrain.

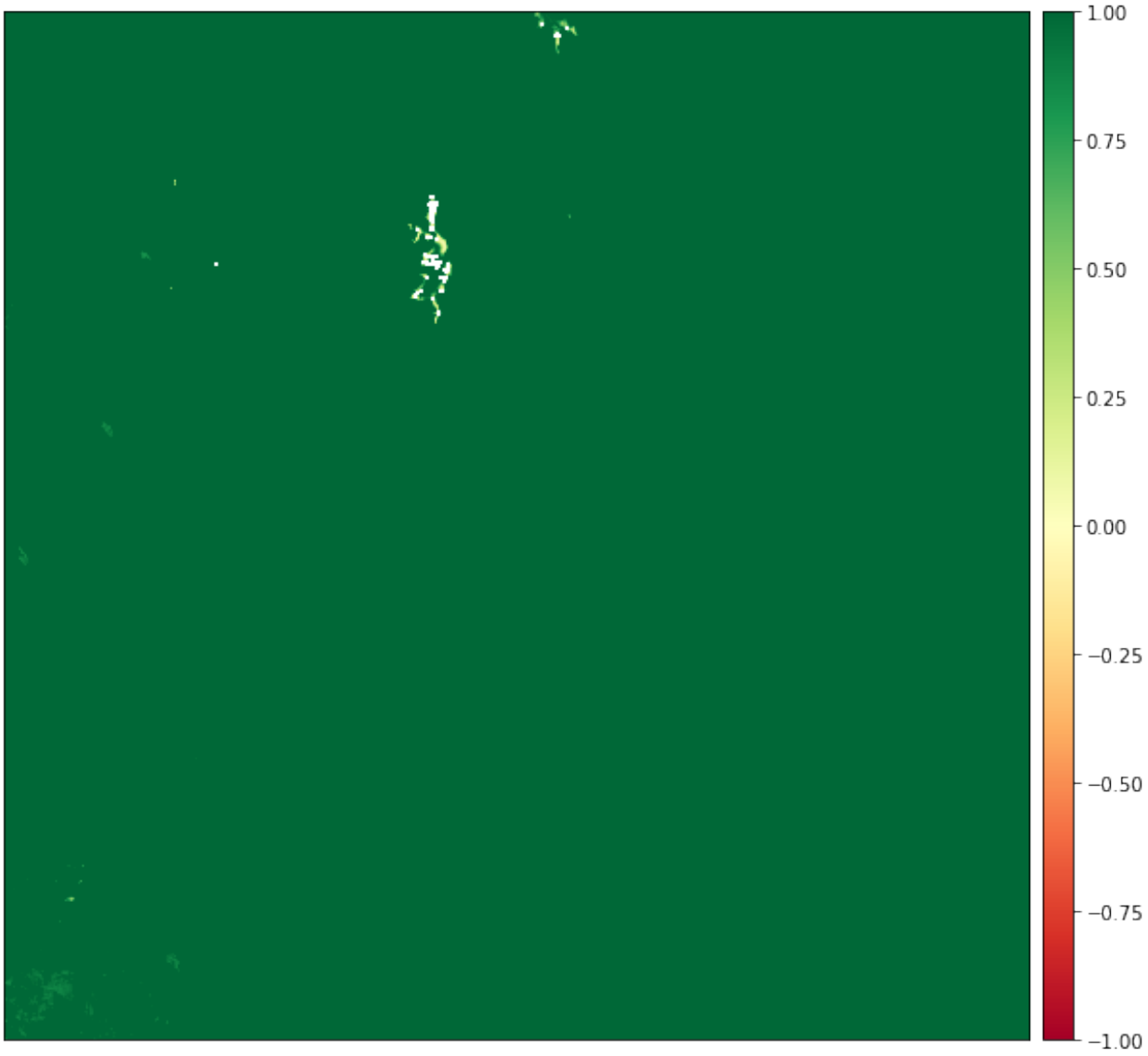
It will be calculated as follows,

CMR = SWIR1 / SWIR2

SWIR1 - Shortwave Infrared (Band - 11) SWIR2 - Shortwave Infraeed (Band - 12)

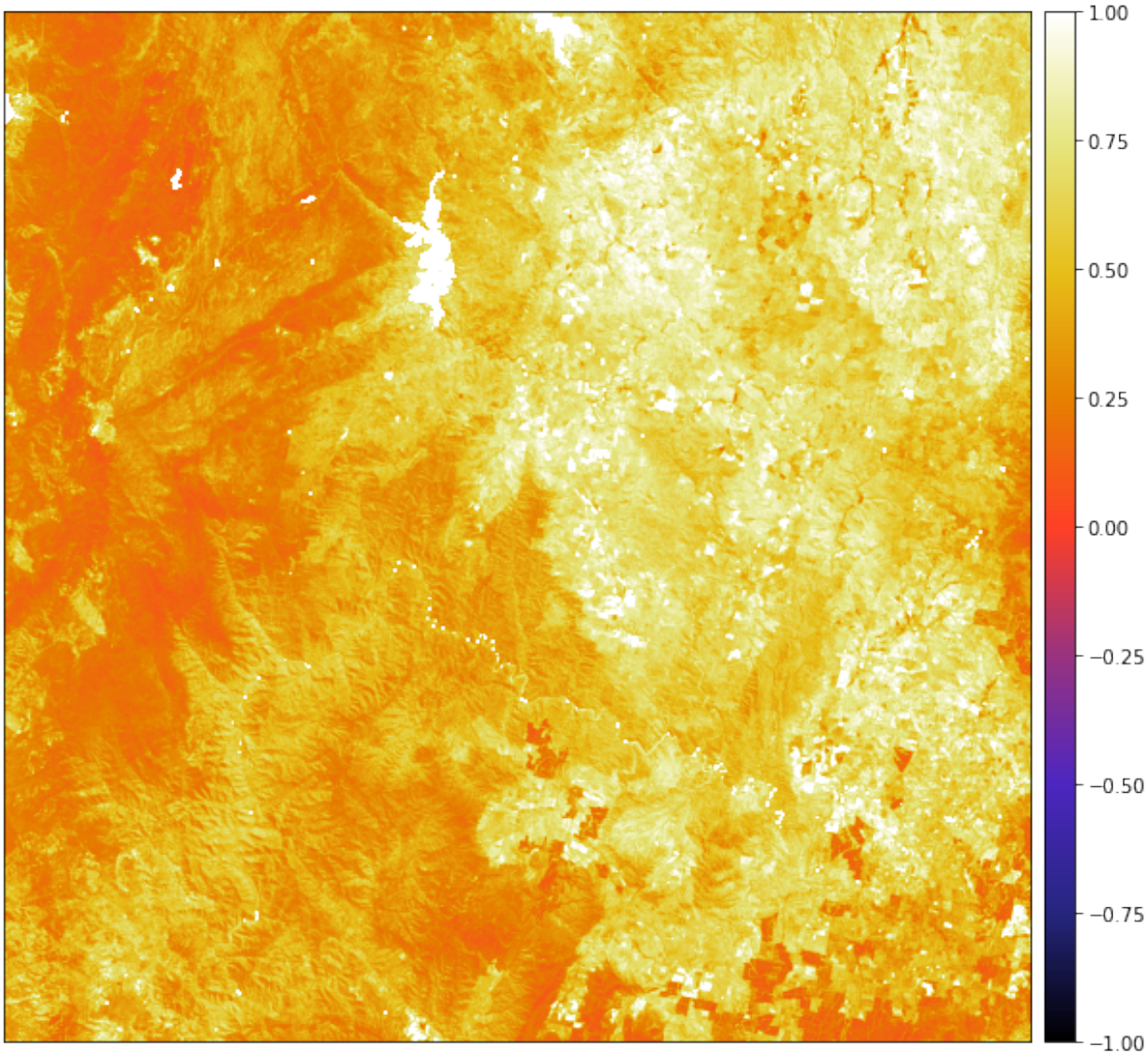
Let's see the CMR for pre-fire image.

In [81]:
```
# Calculate the CMR for pre-fire data using numpy.divide()
cmr = np.divide(prefire_stack[7], prefire_stack[8])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(cmr,
              cmap="RdYlGn",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let us see for the CMR for the post-fire data.

In [73]:
```
# Calculate the CMR for post-fire data using numpy.divide()
cmr = np.divide(postfire_stack[7], postfire_stack[8])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(cmr,
              cmap="CMRmap",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



In above comparison both images, It lost the minerals in the soil post fire.

Ferrous Minerals Ratio (FMR)

Iron-bearing materials are highlighted by the ferrous minerals ratio. It makes use of the ratio between the SWIR and NIR bands.

The calculation of the FMR as follows,

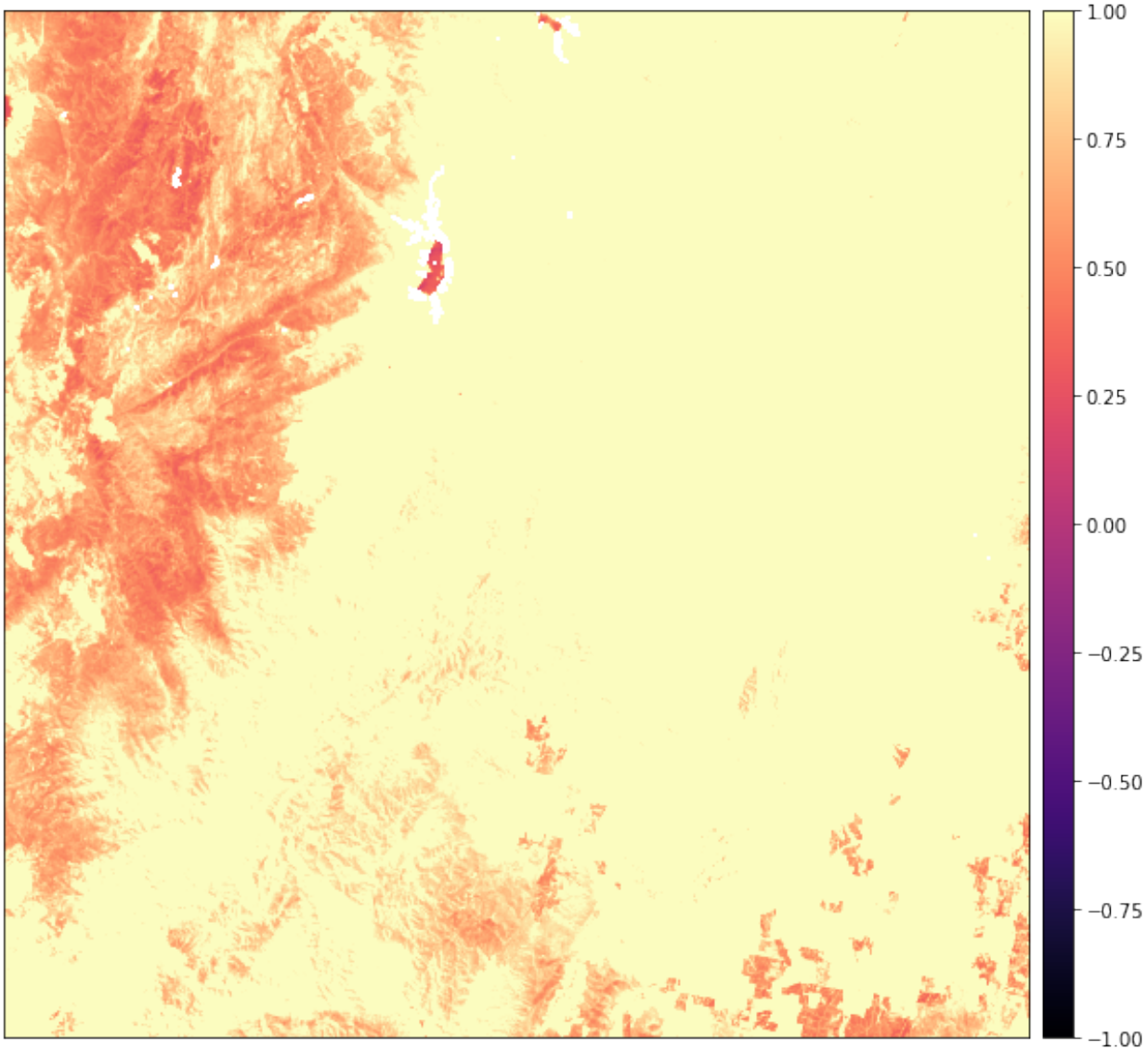
FMR = SWIR1 / NIR

SWIR1 - Shortwave Infrared1 (Band - 11)

NIR - Near Infrared (Band - 8)

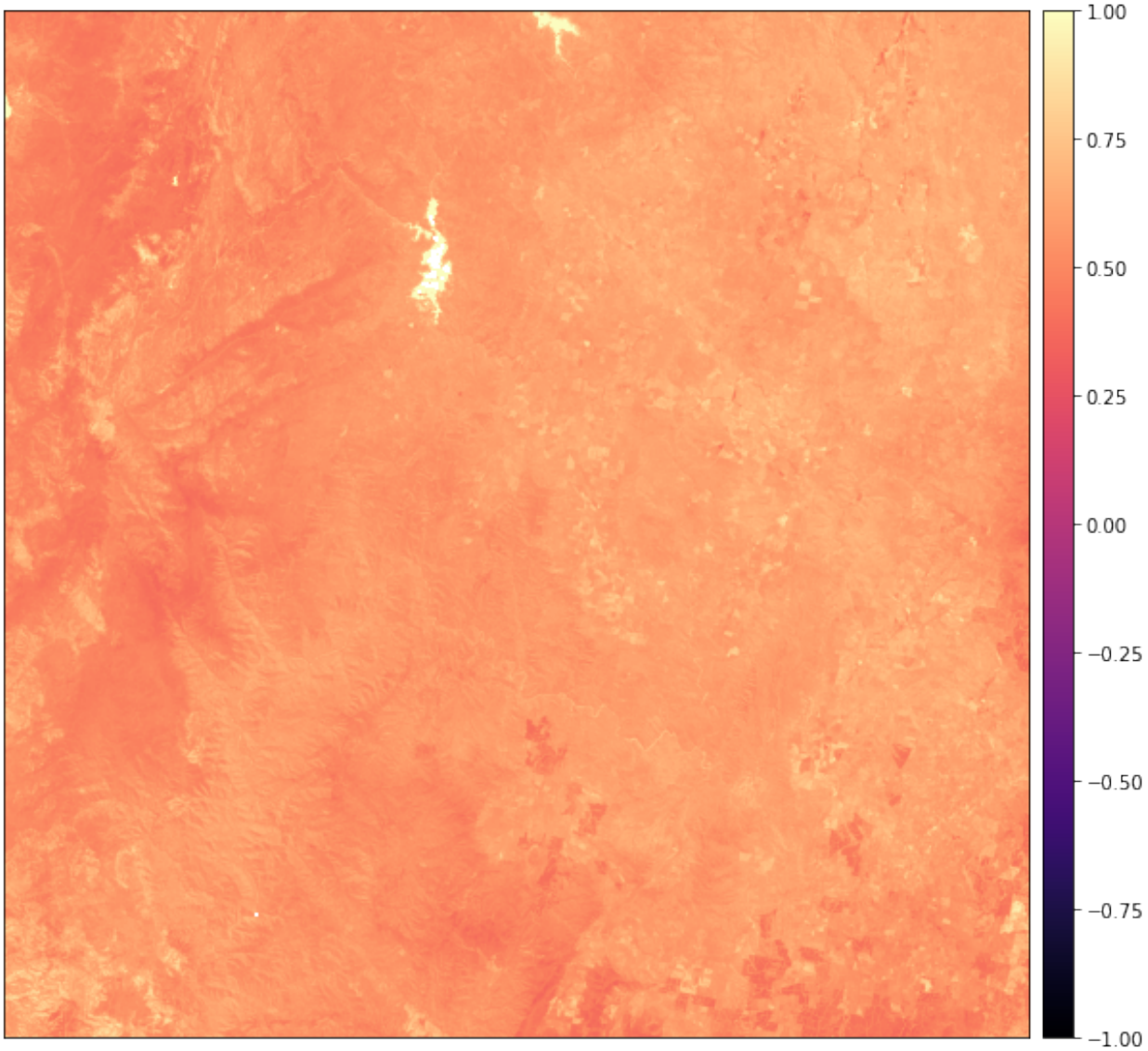
In below code we will calculate the FMR for pre-fire data.

In [78]:
```
# Calculate the FMR for pre-fire data using numpy.divide()
fmr = np.divide(prefire_stack[7], prefire_stack[6])

# Plot the data using earthpy.plot.plot_bands()
ep.plot_bands(fmr,
              cmap="magma",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let's plot FMR for the post-fire data.

In [79]:
```
# Calculate the FMR for post-fire data using numpy.divide()
fmr = np.divide(postfire_stack[7], postfire_stack[6])

# Plot the bands using earthpy.plot.plot_bands()
ep.plot_bands(fmr,
              cmap="magma",
              vmin=-1,
              vmax=1,
              figsize=(10, 14))

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



In above comparison both images, It lost the iron content in the soil post fire.

Fire affected areas detection

Normalized Burn Ratio (NBR)

As previously stated, spectral reflectance can be used to distinguish various properties of the earth's surface, including vegetation, moisture, burnt areas, and water, among other things. Different indices, such as the Normalized Difference Vegetation Index (NDVI) and the Normalized Difference Water Index (NDWI), use different bands of light to distinguish vegetation and water-based bodies. Using the satellite image's Near Infra-Red (NIR) and Short Wave Infra-Red (SWIR) bands, the Normalized Burn Ratio (NBR) is used to identify burnt areas in fire zones.

A high NBR value indicates healthy vegetation, while a low NBR value indicates bare land or areas that have been recently burned. Non-burned areas had values similar to zero.

The formula for calculating NBR as follows,

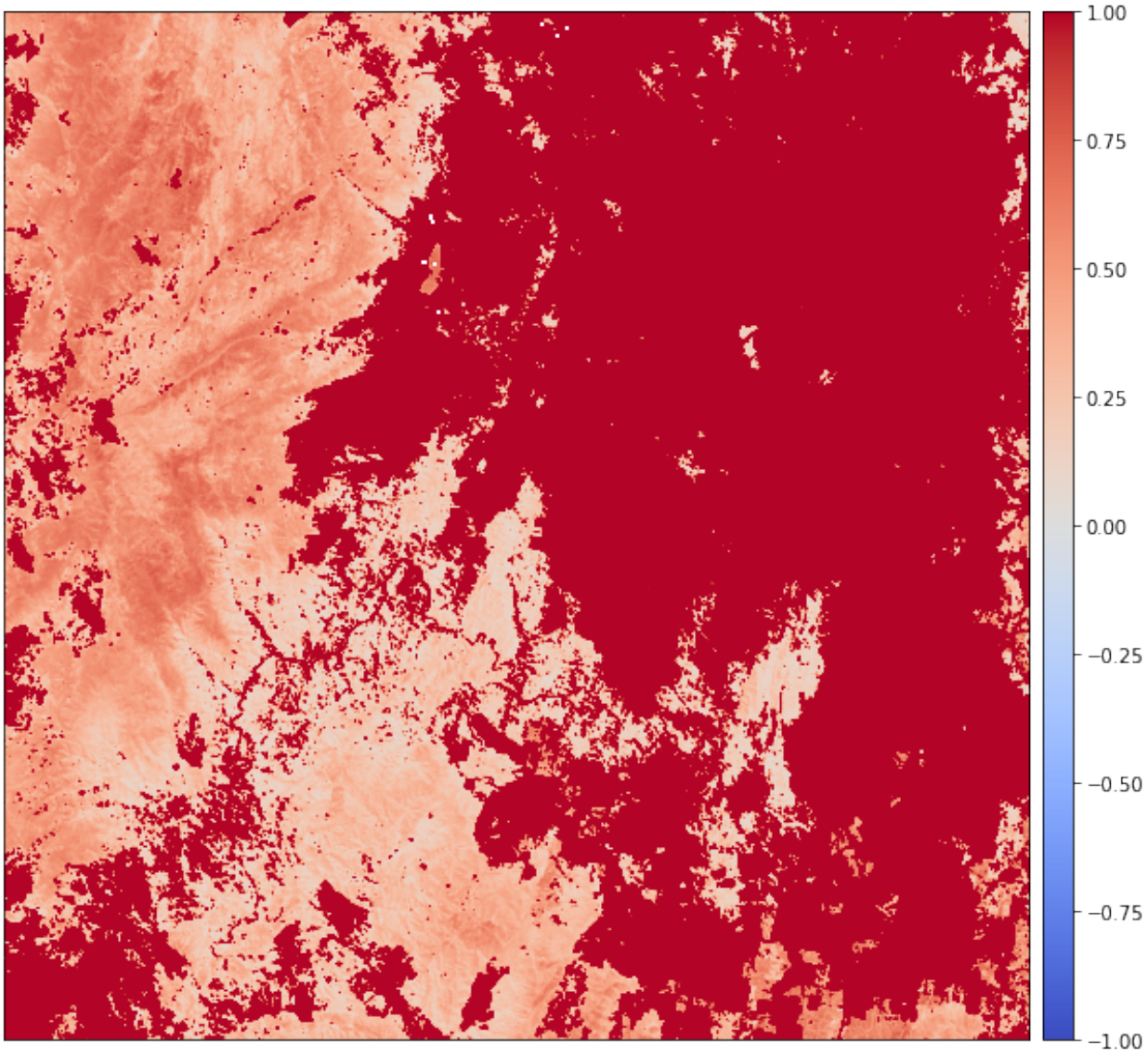
NBR = (NIR - SWIR)/(NIR + SWIR)

NIR - Near Infrared (Band-8)

SWIR - Shortwave Infrared (Band-12)

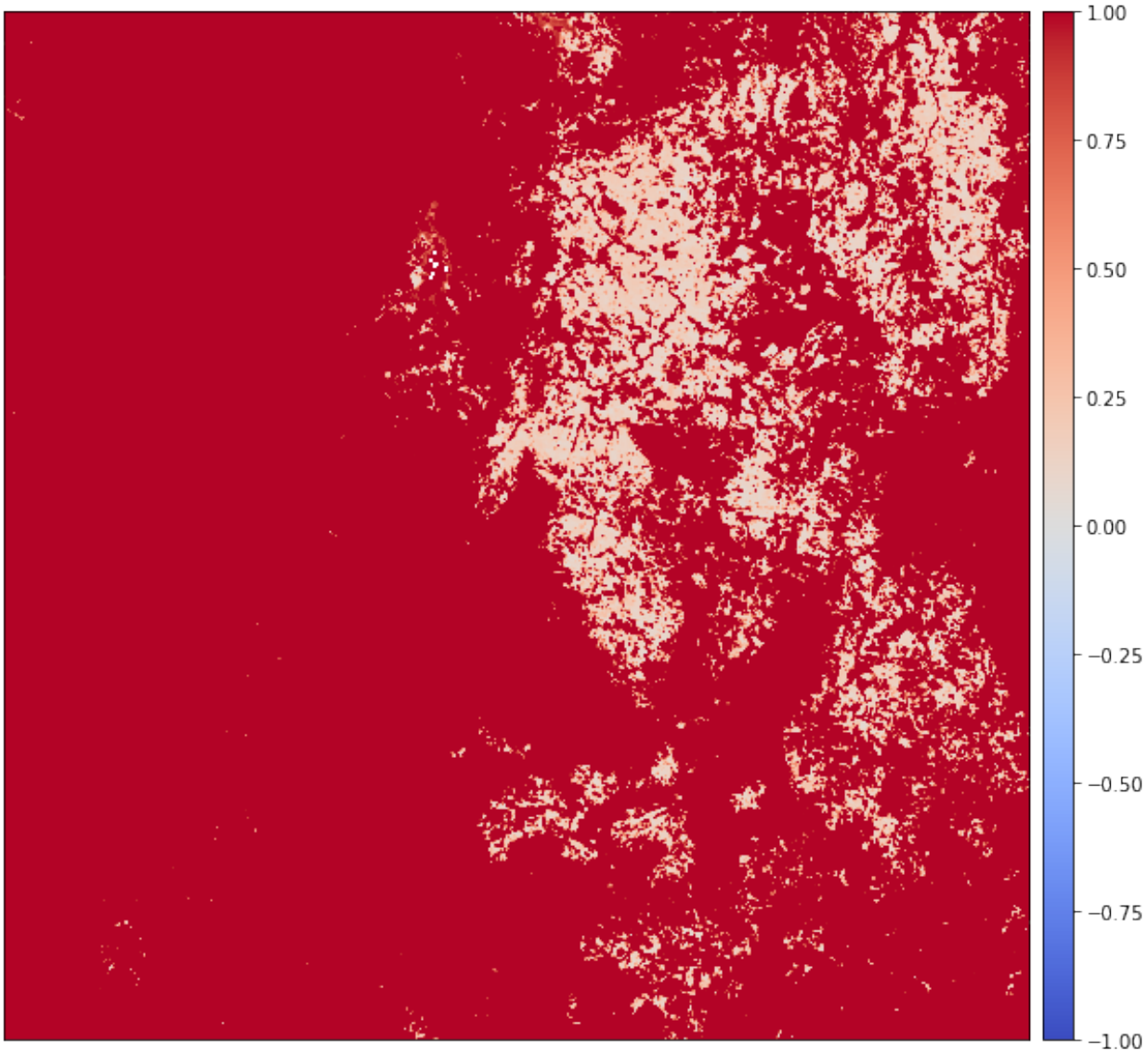
Let us calculate the NBR for Pre-fire data, in below caode.

In [83]:
```
# Normalized Burn Ratio(NBR) using earthpy.spatial for pre-fire data
prefire_nbr1 = es.normalized_diff(prefire_stack[6, :, :], prefire_stack[8,

# Plot NBR bands using earthpy.plot.plot_bands()
ep.plot_bands(prefire_nbr1,
              cmap='coolwarm',
              vmin=-1,
              vmax=1,
              figsize=(10, 14),
              scale=False)

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Let us calculate NBR for the post-fire data in below code.

In [84]:
```
# Normalized Burn Ratio(NBR) using earthpy.spatial for post-fire data
postfire_nbr1 = es.normalized_diff(postfire_stack[6, :, :], postfire_stack

# Plot NBR bands using earthpy.plot.plot_bands()
ep.plot_bands(postfire_nbr1,
              cmap='coolwarm',
              vmin=-1,
              vmax=1,
              figsize=(10, 14),
              scale=False)

# Plot the image using matplotlib.pyplot.show()
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Burn Severity Classification

Delta Normalized Burn Ration (dNBR)

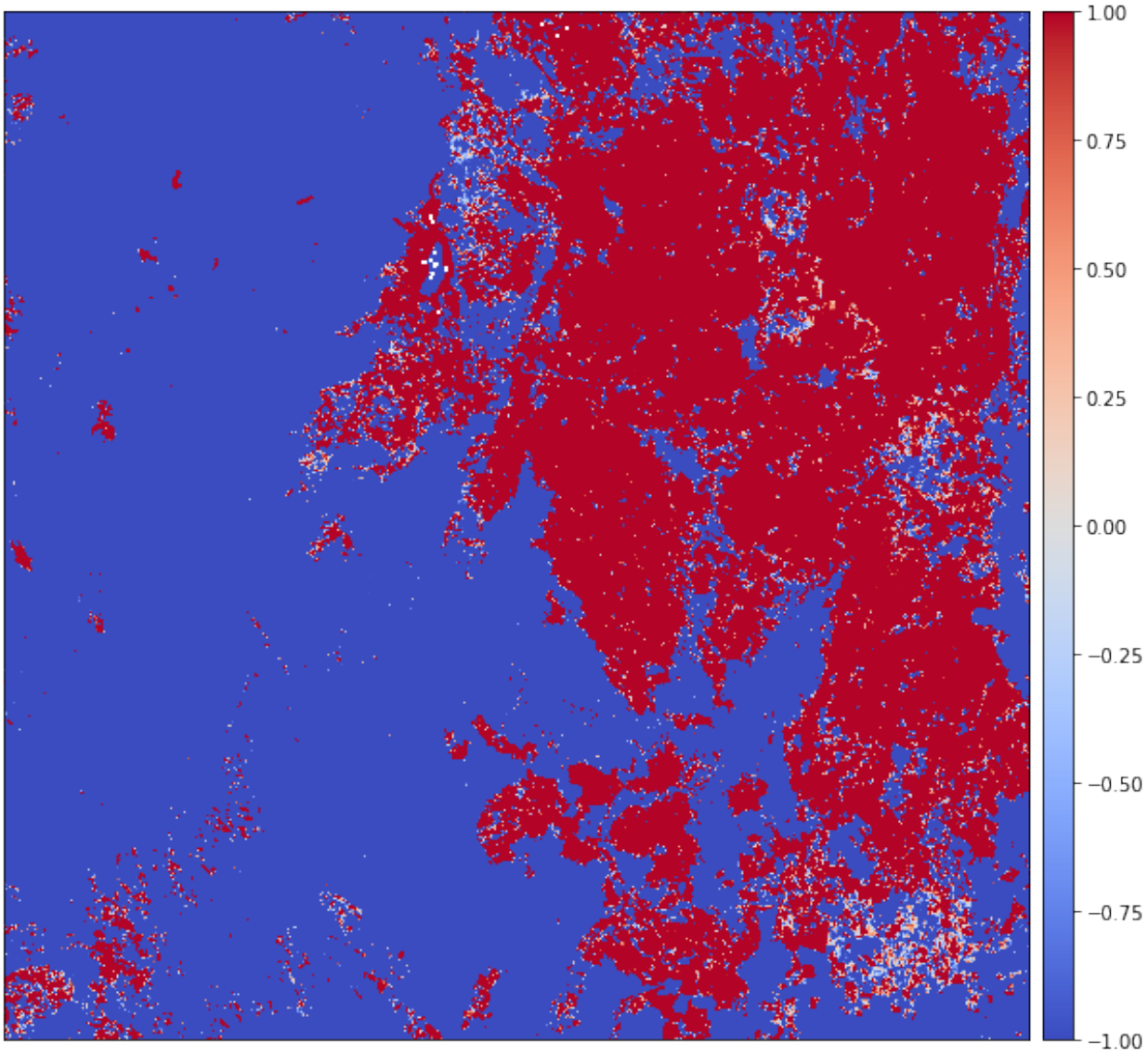
The Delta Normalized Burn Ratio (dNBR), which is the difference between the pre-fire and post-fire NBR, is used to determine the burn intensity. The higher the dBNR value, the more serious the damage, while the lower the dNBR value, the more regrowth after the burn. The dNBR values will differ from one case to the next. As a result, the United States Geological Survey (USGS) has suggested various burn intensity standards for dNBR values. The table below illustrates how dNBR values are used to classify burn severity.

Severity Level	dNBR Range (scaled by 10^3)	dNBR Range (not scaled)
Enhanced Regrowth, high (post-fire)	-500 to -251	-0.500 to -0.251
Enhanced Regrowth, low (post-fire)	-250 to -101	-0.250 to -0.101
Unburned	-100 to +99	-0.100 to +0.99
Low Severity	+100 to +269	+0.100 to +0.269
Moderate-low Severity	+270 to +439	+0.270 to +0.439
Miderate-high Severity	+440 to +659	+0.440 to +0.659
High Severity	+660 to +1300	+0.660 to +1.300

Image source : USGS

Let us calculate the dNBR for the pre-fire and post fire data.

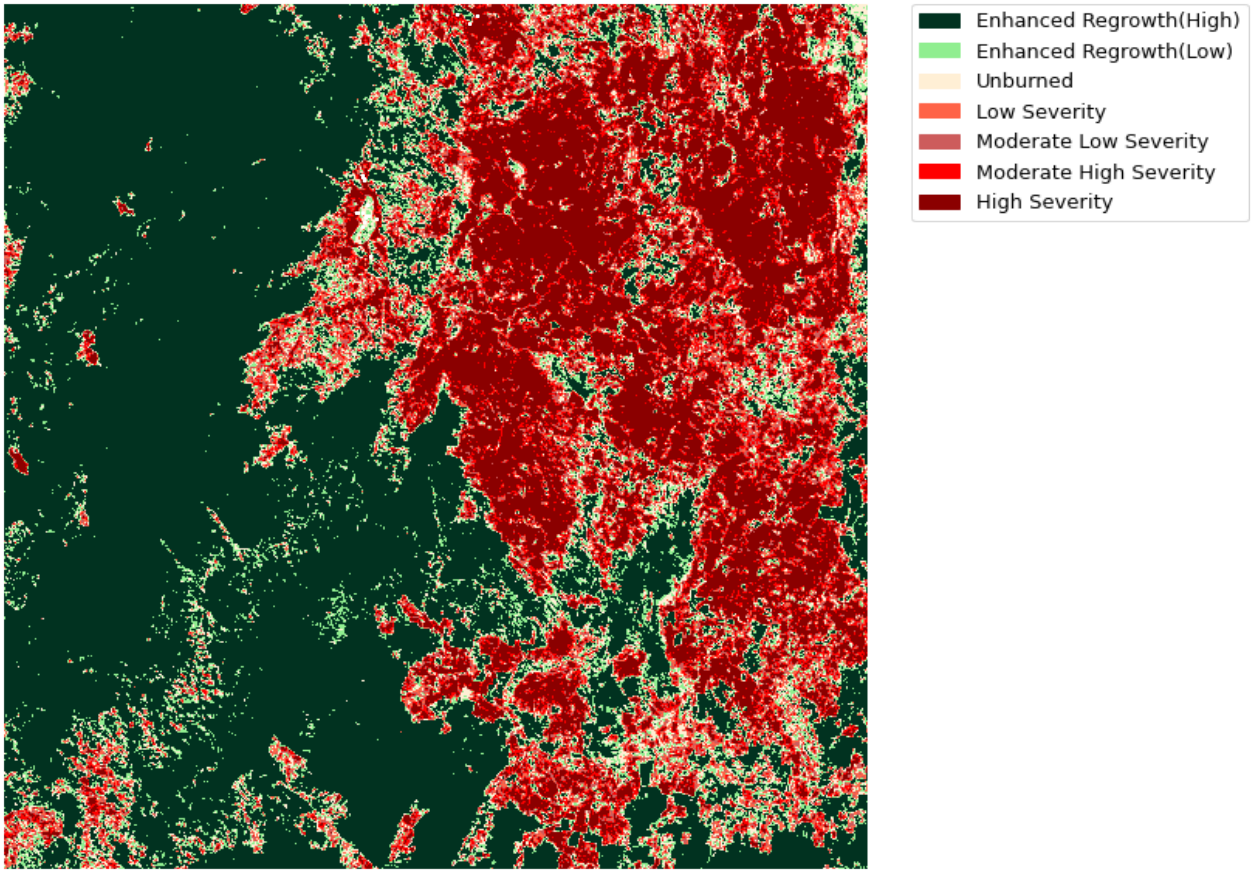
In [85]:
```
# dNBR calculation for pre-fire and post-fire data
dnbr = prefire_nbr1 - postfire_nbr1

# Plot dBNR data using earthpy.plot.plot_bands()
ep.plot_bands(dnbr,
              cmap='coolwarm',
              vmin=-1,
              vmax=1,
              figsize=(10, 14),
              scale=False)

# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Based on the dBNR values, the following code divides them into seven categories: Enhanced Regrowth(High), Enhanced Regrowth(Medium), Unburned, Low Severity, Moderate Low Severity, Moderate High Severity, and High Severity. The ploting the classification of burn severity as follows.

Burn Severity Classification

In [92]:
```
# Assigning the class bins as per the USGS burn severity values
class_bins = [-.5, -.251, -.101, .099, .269, .439, 0.659, 1.3]
# Use numpy.digitize() to return the indices of the bins
dnbr_class = np.digitize(dnbr, class_bins)
# Apply the nodata mask to the newly classified NDVI data
dnbr_class = np.ma.masked_where(np.ma.getmask(dnbr), dnbr_class)
# Assign the classification categories
dnbr_cat_names = ["Enhanced Regrowth(High)",
                  "Enhanced Regrowth(Low)",
                  "Unburned",
                  "Low Severity",
                  "Moderate Low Severity",
                  "Moderate High Severity",
                  "High Severity"]
# Assign the colors to the classification categories
nbr_colors = ["#013220","#90ee90","#FFEFD5",
                "#FF6347","#CD5C5C","#FF0000","#8B0000"]
# Use the listedcolormap() function to assign the data
nbr_cmap = ListedColormap(nbr_colors)
# Get list of classes using numpy.unique()
classes = np.unique(class_bins)
classes = classes.tolist()
# The mask returns a value of none in the classes and removes that
classes = classes[0:7]
# Plot the data
# Use matplotlib.pyplot.subplots()
fig, ax = plt.subplots(figsize=(10, 14))
im = ax.imshow(dnbr_class,
               cmap=nbr_cmap)
# Add legends in the map using earthpy.plot.draw_legend()
ep.draw_legend(im_ax=im,
               classes=classes,
               titles=dnbr_cat_names)
# Subpolt to fit figure size
ax.set_axis_off()
# Plot the image using matplotlib.pyplot.show()
plt.show()
```



Conclusion

To put it in a nutshell, the changes in the intensity of a high-severity fire, as well as its duration and spatial configuration, can affect a number of ecosystem processes that interact to assess post-fire recovery, including the conversion to non-forest alternative states. Between January 2020 and March 2021, our study revealed a rise in some areas and vegetation regrowth in both the total and proportion of high-severity burned area in Canberra. High-severity patches have become more aggregated and irregular in appearance over time. Changes in the spatial patterns of high-severity fire across period can have unforeseen consequences on natural vegetation, demonstrating the increased damage caused to forest habitats by evolving fire regimes.