Working with multiple bands in R.

Learning Objectives

After completing this tutorial, you will be able to:

What you need

You will need a computer with internet access to complete this lesson and the data for week 6 of the course.

About Raster Bands in R

In the previous weeks, we've worked with rasters derived from lidar remote sensing instruments. These rasters consisted of one layer or band. In this lesson, we'll learn how to work with rasters with multiple bands in `R`.

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/single_mul
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/single_mu
raster function to import one single band from a single OR multi-band
raster. ">
</a>
<figcaption>A raster can contain one or more bands. We can use the
raster function to import one single band from a single OR multi-band
raster.  Source: Colin Williams, NEON.</figcaption>
```

Previously, we used the `raster()` function to open raster data in `R`. To work with multi-band rasters in `R`, we need to change how we import and plot our data in several ways.

- To import multi band raster data we will use the `stack()` function.
- If our multi-band data are imagery that we wish to composite, we can use `plotRGB()`, instead of `plot()`, to plot a 3 band raster image.

About Multi-Band Imagery

One type of multi-band raster dataset that is familiar to many of us is a color image. A basic color image consists of three bands: red, green, and blue. Each band represents light reflected from the red, green or blue portions of the electromagnetic spectrum. The pixel brightness for each band, when composited creates the colors that we see in an image.

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/RGBSTack_
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/RGBSTack
rendered together in a GIS, or even a tool like Photoshop or any other
image software, the 3 bands create a color image."></a>
<figcaption>A color image consists of 3 bands - red, green and blue. When
rendered together in a GIS, or even a tool like Photoshop or any other
image software, the 3 bands create a color image.
Source: Colin Williams, NEON.
</figcaption>
```

We can plot each band of a multi-band image individually using a grayscale color gradient. Remember that the LIGHTER colors represent a stronger reflection in that band. DARKER colors represent a weaker reflection.

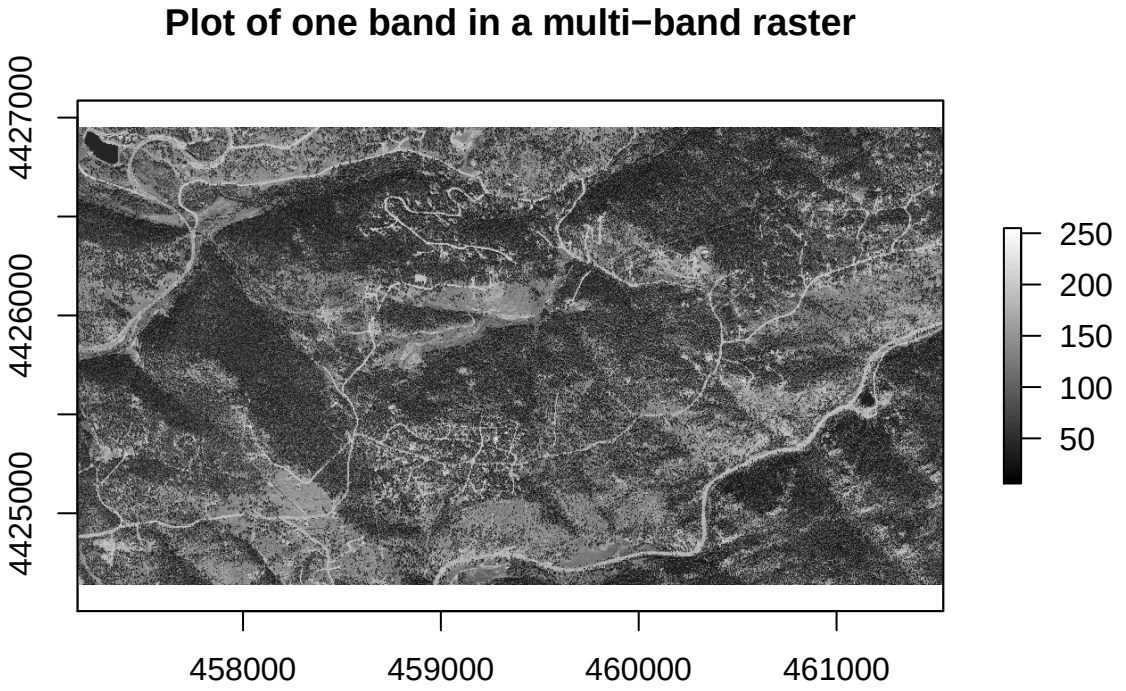

Figure 1: single band image

Each band plotted separately

Note there are four bands below. You are looking at the blue, green, red and Near infrared bands of a NAIP image.

We can plot the red, green and blue bands together to create an RGB image. This is what we would see with our eyes if we were in the airplane looking down at the earth.

CIR image

If the image has a 4th NIR band, you can create a CIR (sometimes called false color) image. here, the NIR band is plotted using the "red" band. Thus vegetation, which reflects stronly in the NIR part of the spectrum, is colored "red".

Other Types of Multi-band Raster Data

Multi-band raster data might also contain:

1. **Time series:** the same variable, over the same area, over time.
2. **Multi or hyperspectral imagery:** image rasters that have 4 or more (multi-spectral) or more than 10-15 (hyperspectral) bands.

Work with Landsat data in R

Now, we have learned that basic concepts associated with a raster stack. We want to work with spectral imagery to better understand our study site - which is the cold springs fire scare in Colorado.

To work with multi-band raster data we will use the `raster` and `rgdal` packages.

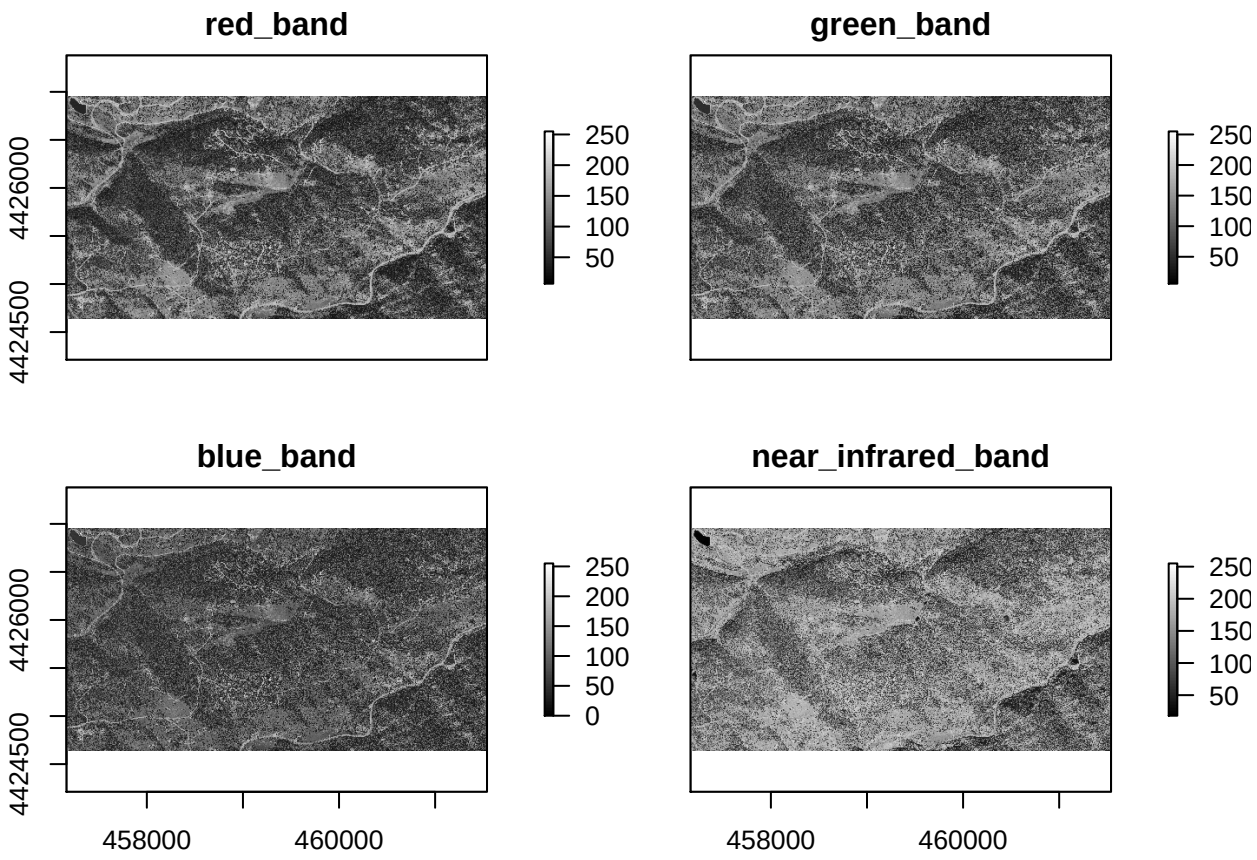

Figure 2: All bands plotted separately

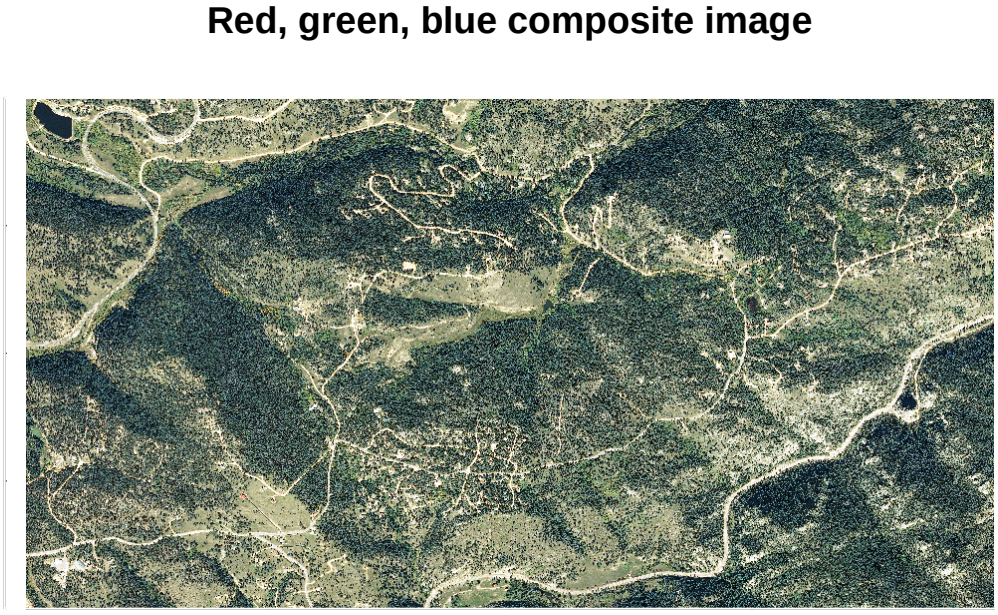

Figure 3: 3 band image plot rgb

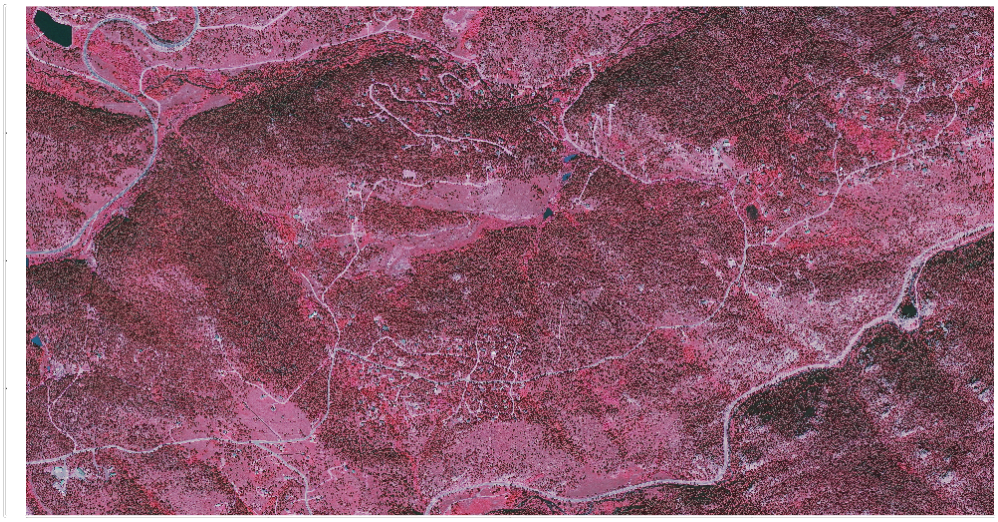

Figure 4: 3 band cir image

```
# load spatial packages
library(raster)
library(rgdal)
library(rgeos)
```

In this lesson we will use imagery from the National Agricultural Imagery Program (NAIP).

About NAIP:

> The National Agriculture Imagery Program (NAIP) acquires aerial imagery during the agricultural growing seasons in the continental U.S. A primary goal of the NAIP program is to make digital ortho photography available to governmental agencies and the public within a year of acquisition.
>
> NAIP is administered by the USDA's Farm Service Agency (FSA) through the Aerial Photography Field Office in Salt Lake City. This "leaf-on" imagery is used as a base layer for GIS programs in FSA's County Service Centers, and is used to maintain the Common Land Unit (CLU) boundaries.

Read more about NAIP

NAIP is A great source of high resolution imagery across the United States. NAIP imagery is often flown with just a Red, green and Blue band. However, some flights include a near infrared band which is very useful for quantifying vegetation cover and health.

NAIP data access: For this lesson we used the USGS Earth explorer site to download NAIP imagery.

Next, let's open up our NAIP imagery for the Coldsprings fire study area in Colorado.

```
# Read in multi-band raster with raster function.
# Default is the first band only.
# csf = cold springs fire!
```

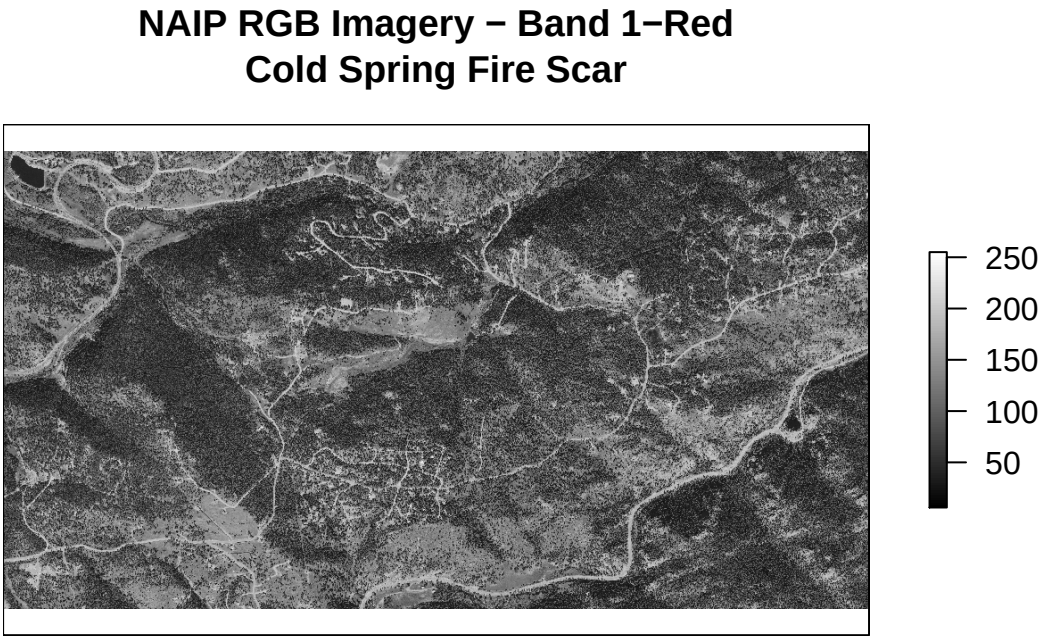

Figure 5: naip imagery single band plot.

```
naip_csf <- raster("data/week6/naip/m_3910505_nw_13_1_20130926/crop/m_3910505_nw_13_1_20130926_crop.tif

# Plot band 1
plot(naip_csf,
     col=gray(0:100 / 100),
     axes=FALSE,
     main="NAIP RGB Imagery - Band 1-Red\nCold Spring Fire Scar")
```

```
# view data dimensions, CRS, resolution, attributes, and band info
naip_csf
## class       : RasterLayer
## band        : 1  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop
## values      : 0, 255  (min, max)
```

Notice that when we look at the attributes of RGB_Band1, we see:

`band: 1  (of  4  bands)`

This is `R` telling us that this particular raster object has more bands (4) associated with it than we imported using the `raster()` function.

Data Tip: The number of bands associated with a raster object can also be determined using the `nbands` slot. Syntax is `ObjectName@file@nbands`, or specifically for our file: `RGB_band1@file@nbands`. {: .notice}

Image Raster Data Values

Let's next examine the raster's min and max values. What is the value range?

```
# view min value
minValue(naip_csf)
## [1] 0

# view max value
maxValue(naip_csf)
## [1] 255
```

This raster contains values between 0 and 255. These values represent degrees of brightness associated with the image band. In the case of a RGB image (red, green and blue), band 1 is the red band. When we plot the red band, larger numbers (towards 255) represent pixels with more red in them (a strong red reflection). Smaller numbers (towards 0) represent pixels with less red in them (less red was reflected). To plot an RGB image, we mix red + green + blue values into one single color to create a full color image - similar to the color image a digital camera creates.

Import A Specific Band

We can use the `raster()` function to import specific bands in our raster object by specifying which band we want with `band=N` (N represents the band number we want to work with). To import the green band, we would use `band=2`.

```
# Can specify which band we want to read in
rgb_band2 <- raster("data/week6/naip/m_3910505_nw_13_1_20130926/crop/m_3910505_nw_13_1_20130926_crop.ti
             band = 2)

# plot band 2
plot(rgb_band2,
     col=gray(0:100 / 100),
     axes=FALSE,
     main="RGB Imagery - Band 2 - Green\nCold Springs Fire Scar")
```

```
# view attributes of band 2
rgb_band2
## class       : RasterLayer
## band        : 2  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop
## values      : 0, 255  (min, max)
```

Notice that band 2 is the second of 3 bands `band: 2  (of  4  bands)`.

Raster Stacks in R

Next, we will work with all four image bands (red, green and blue) as an `R RasterStack` object. We will then plot a 3-band composite, or full color, image.

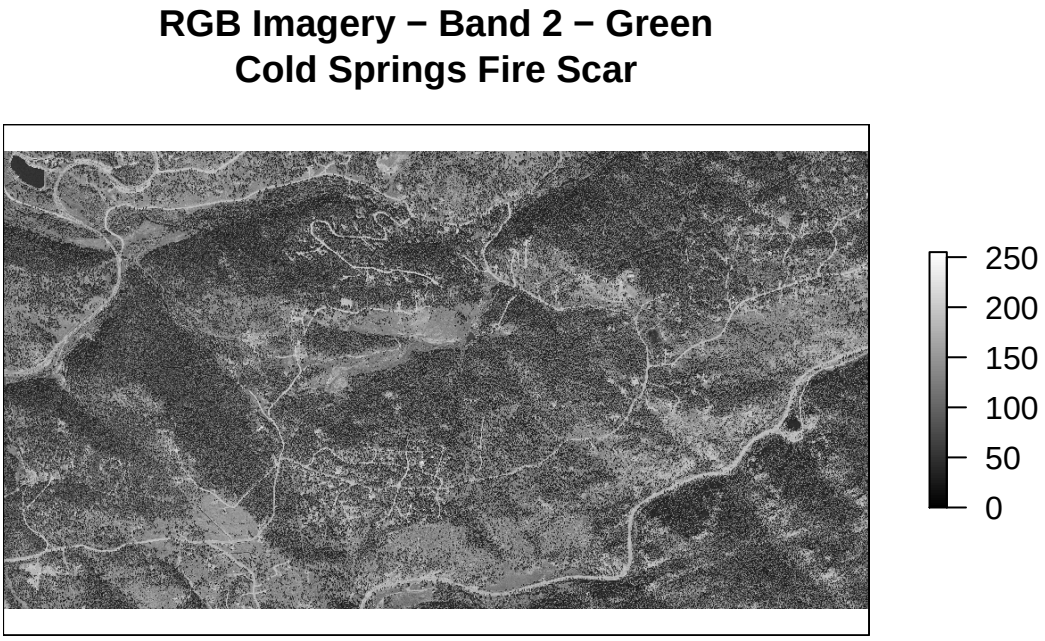

Figure 6: naip imagery band 2 plot.

To bring in all bands of a multi-band raster, we use the`stack()` function. IMPORTANT: All rasters in a raster stack must have the same *extent*, *CRS* and *resolution*.

```
# Use stack function to read in all bands
naip_stack_csf <-
  stack("data/week6/naip/m_3910505_nw_13_1_20130926/crop/m_3910505_nw_13_1_20130926_crop.tif")

# view attributes of stack object
naip_stack_csf
## class       : RasterStack
## dimensions  : 2312, 4377, 10119624, 4  (nrow, ncol, ncell, nlayers)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## names       : m_3910505_nw_13_1_20130926_crop.1, m_3910505_nw_13_1_20130926_crop.2, m_3910505_nw_13_
## min values  :                                 0,                                 0,
## max values  :                               255,                               255,
```

We can view the attributes of each band the stack using `naip_stack_csf@layers`. Or we if we have hundreds of bands, we can specify which band we'd like to view attributes for using an index value: `naip_stack_csf[[1]]`. We can also use the `plot()` and `hist()` functions on the `RasterStack` to plot and view the distribution of raster band values.

```
# view raster attributes
naip_stack_csf@layers
## [[1]]
## class       : RasterLayer
## band        : 1  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
```

```
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop.1
## values      : 0, 255  (min, max)
##
##
## [[2]]
## class       : RasterLayer
## band        : 2  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop.2
## values      : 0, 255  (min, max)
##
##
## [[3]]
## class       : RasterLayer
## band        : 3  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop.3
## values      : 0, 255  (min, max)
##
##
## [[4]]
## class       : RasterLayer
## band        : 4  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop.4
## values      : 0, 255  (min, max)

# view attributes for one band
naip_stack_csf[[1]]
## class       : RasterLayer
## band        : 1  (of  4  bands)
## dimensions  : 2312, 4377, 10119624  (nrow, ncol, ncell)
## resolution  : 1, 1  (x, y)
## extent      : 457163, 461540, 4424640, 4426952  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs
## data source : /Users/lewa8222/Documents/earth-analytics/data/week6/naip/m_3910505_nw_13_1_20130926/c
## names       : m_3910505_nw_13_1_20130926_crop.1
## values      : 0, 255  (min, max)

# view histogram for each band
```

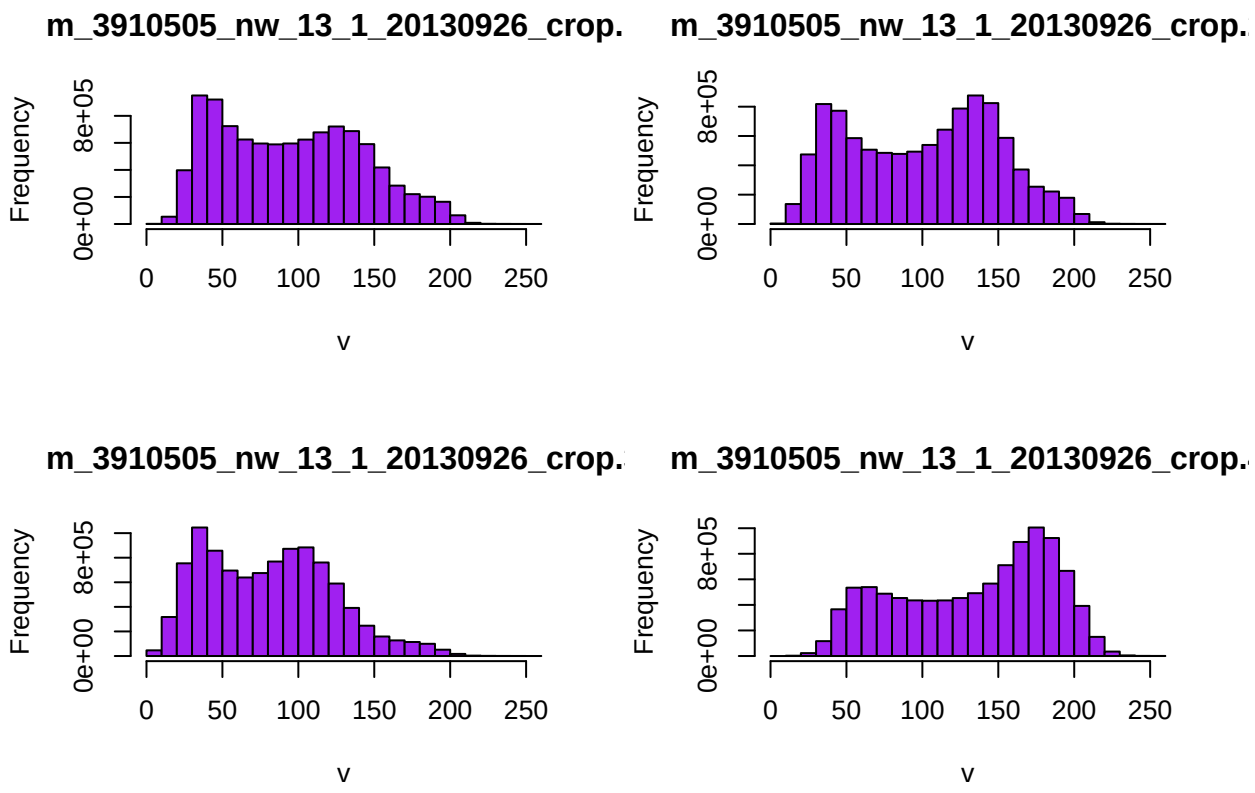

Figure 7: histogram of each band for a total of 4 bands

```
hist(naip_stack_csf,
     maxpixels=ncell(naip_stack_csf),
     col="purple")
```

We can view a histogram of each band in our stack. This is useful to better understand the distribution of reflectance values for each band.

```
# plot 4 bands separately
plot(naip_stack_csf,
     col=gray(0:100 / 100))
```

We can plot just one band too if we want.

```
# plot band 2
plot(naip_stack_csf[[2]],
     main="NAIP Band 2\n Coldsprings Fire Site",
     col=gray(0:100 / 100))
```

Optional challenge: Making Sense of Single Band Images

Use the plot() command to compare grayscale plots of band 1 (red), band 2 (green) and band 4 (near infrared). Is the forested area darker or lighter in band 2 (the green band) compared to band 1 (the red band)?

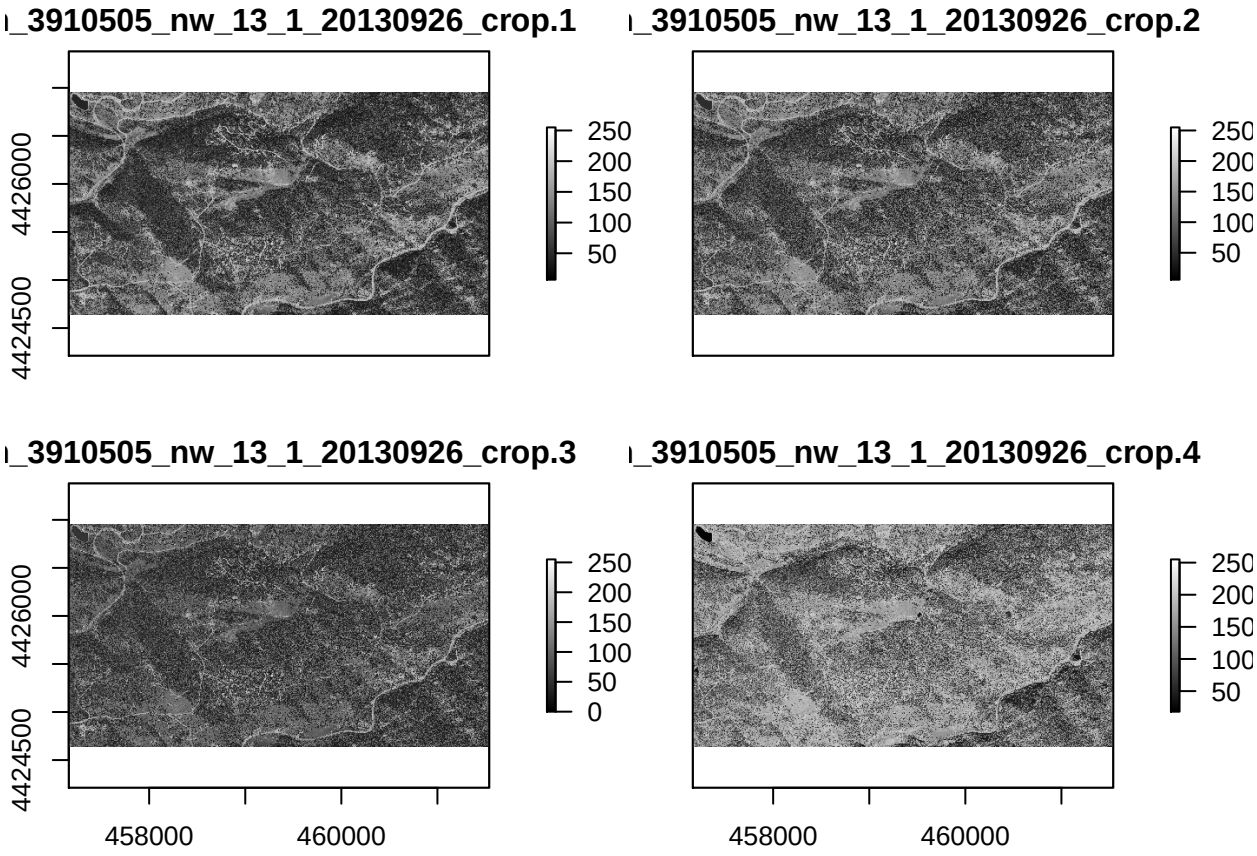

Figure 8: plot each band for a total of 4 bands

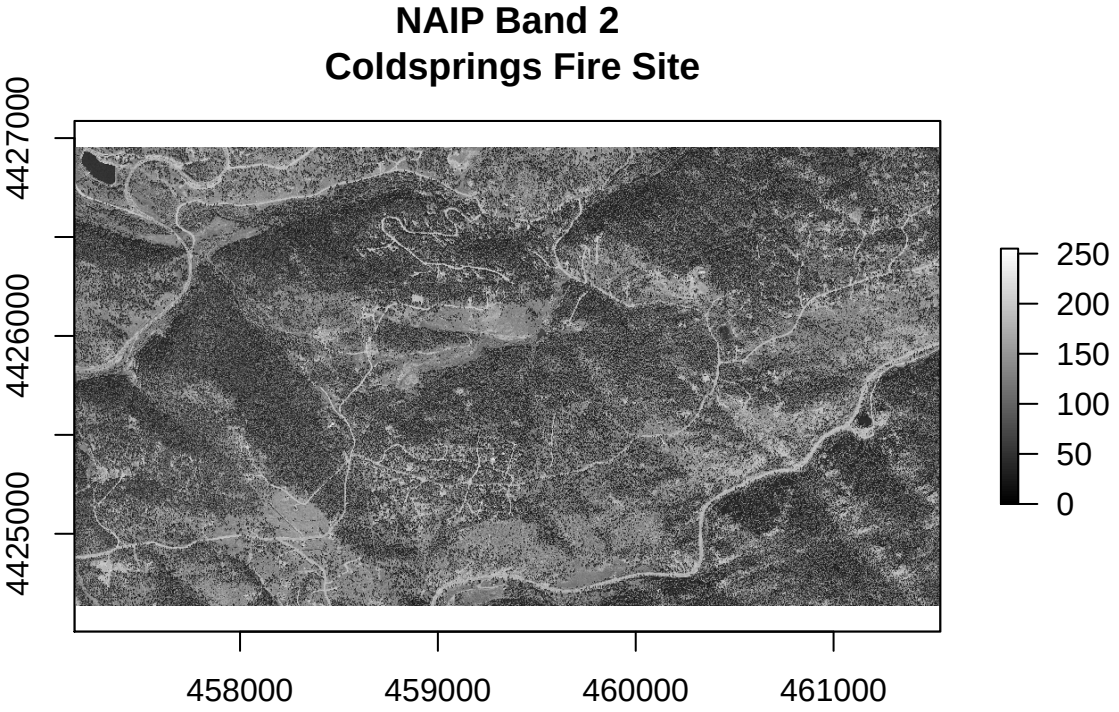

Figure 9: plot individual band - band 2

Figure 10: RGB image of NAIP imagery.

RGB Data

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week6/RGBSTack_1
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week6/RGBSTack_
<figcaption>A "true" color image consists of 3 bands - red, green and blue.
When composited or rendered together in a GIS, or even a image-editor like
Photoshop the bands create a color image.
Source: NEON.
</figcaption>
```

Use plotRGB() to create a composite 3 band image

To render a 3 band, color image in `R`, we use `plotRGB()`.

This function allows us to:

1. Identify what bands we want to render in the red, green and blue regions. The `plotRGB()` function defaults to a 1=red, 2=green, and 3=blue band order. However, you can define what bands you'd like to plot manually. Manual definition of bands is useful if you have, for example a near-infrared band and want to create a color infrared image.
2. Adjust the `stretch` of the image to increase or decrease contrast.

Let's plot our 3-band image.

```
# Create an RGB image from the raster stack
plotRGB(naip_stack_csf,
        r = 1, g = 2, b = 3)
```

Here's how we add a title to our plot. To do this, we adjust the **par**ameters of the plot as follows:

- `col.axis="white"`: set the axes to render in white. if you turn off the axes then the plot title will also be turned off.

Figure 11: RGB image of NAIP imagery.

- `col.lab="white"`: turn plot tick mark labels to white
- `tck=0`: turn off plot "ticks"

Finally after the plot code if you set `box(col="white")` it removes the line that is drawn alongside of your plot.

```
# adjust the plot parameters to render the axes using white
# this is a way to "trick" R
par(col.axis="white", col.lab="white", tck=0)
plotRGB(naip_stack_csf,
        r = 1, g = 2, b = 3)
box(col="white") # turn all of the lines to white
```

The image above looks pretty good. We can explore whether applying a stretch to the image might improve clarity and contrast using `stretch="lin"` or `stretch="hist"`.

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/imageStre
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/imageStr
darker image is rendered by default. We can stretch the values to extend to
the full 0-255 range of potential values to increase the visual contrast of
the image.">
</a>
<figcaption>When the range of pixel brightness values is closer to 0, a
darker image is rendered by default. We can stretch the values to extend to
the full 0-255 range of potential values to increase the visual contrast of
the image.
</figcaption>
```

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/imageStre
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/imageStr
lighter image is rendered by default. We can stretch the values to extend to
the full 0-255 range of potential values to increase the visual contrast of
```

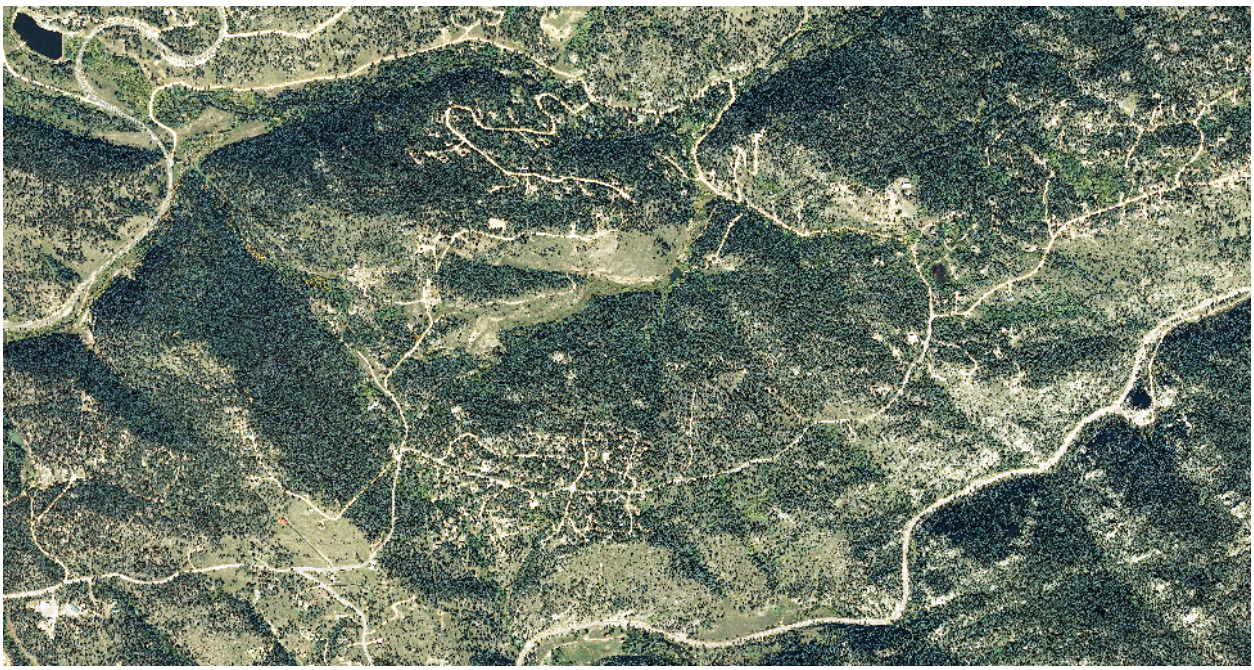

Figure 12: lin stretch rgb image

```
the image.">
</a>
<figcaption>When the range of pixel brightness values is closer to 255, a
lighter image is rendered by default. We can stretch the values to extend to
the full 0-255 range of potential values to increase the visual contrast of
the image.
</figcaption>
```

```
# what does stretch do?
plotRGB(naip_stack_csf,
        r = 1, g = 2, b = 3,
        stretch = "lin")
```

What does the image look like using a different stretch? Any better? worse?

```
plotRGB(naip_stack_csf,
        r = 1, g = 2, b = 3,
        scale=800,
        stretch = "hist")
```

In this case, the stretch doesn't enhance the contrast our image significantly given the distribution of reflectance (or brightness) values is distributed well between 0 and 255. We are lucky! Our NAIP imagery has been processed well and thus we don't need to worry about image stretch.

RasterStack vs RasterBrick in R

The `R` `RasterStack` and `RasterBrick` object types can both store multiple bands. However, how they store each band is different. The bands in a `RasterStack` are stored as links to raster data that is located somewhere on our computer. A `RasterBrick` contains all of the objects stored within the actual `R` object. In most cases, we can work with a `RasterBrick` in the same way we might work with a `RasterStack`. However

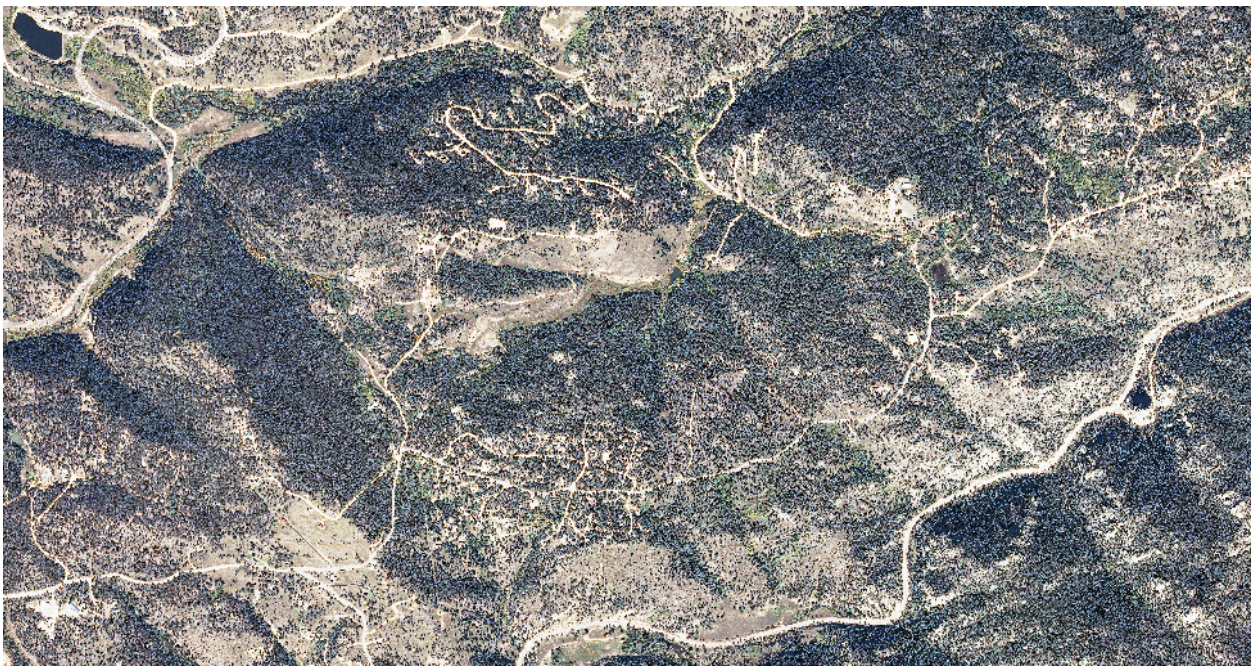

Figure 13: plot RGB with his stretch

a `RasterBrick` is often more efficient and faster to process - which is important when working with larger files.

- More on Raster Bricks

We can turn a `RasterStack` into a `RasterBrick` in `R` by using `brick(StackName)`. Let's use the `object.size()` function to compare `stack` and `brick` `R` objects.

```
# view size of the RGB_stack object that contains our 3 band image
object.size(naip_stack_csf)
## 52424 bytes

# convert stack to a brick
naip_brick_brick <- brick(naip_stack_csf)

# view size of the brick
object.size(naip_brick_csf)
## 12912 bytes
```

Notice that in the `RasterBrick`, all of the bands are stored within the actual object. Thus, the `RasterBrick` object size is much larger than the `RasterStack` object.

You use `plotRGB` to block a `RasterBrick` too.

```
# plot brick
plotRGB(naip_brick_csf)
```

Optional challenge

The NAIP image that we've been working with so far is pre-fire. Import the `naip/m_3910505_nw_13_1_20150919/crop/m_3910` into R and plot a

Figure 14: challenge rgb plot 2015 data

1. RGB image
2. CIR image

Then anwer the following questions:

- How many bands does the raster have?
- What CRS is the raster in?
- What is the resolution of the data?

Optional challenge: What Methods Can Be Used on an R Object?

We can view various methods available to call on an `R` object with `methods(class=class(objectNameHere))`. Use this to figure out:

1. What methods can be used to call on the `naip_stack_csf` object?
2. What methods are available for a single band within `naip_stack_csf`?
3. Why do you think there is a difference?

Additional resources

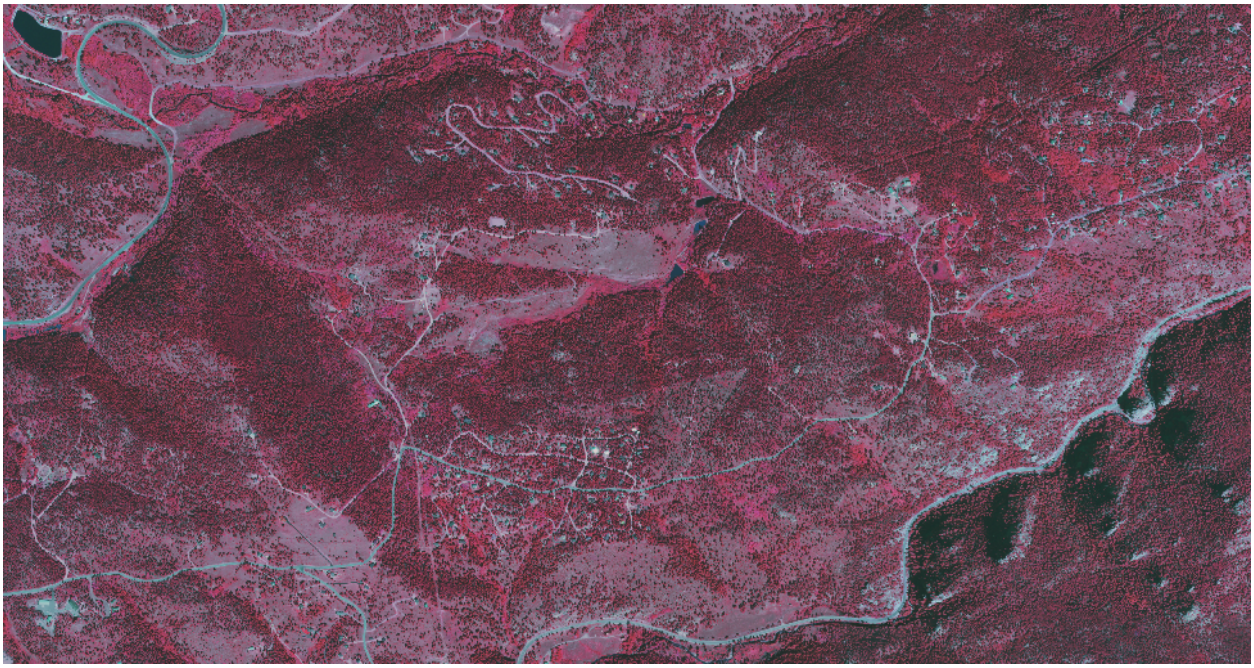

Figure 15: challenge cir plot 2015 data