Landsat tif files 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 5 of the course.

In the previous lesson, we learned how to import a multi-band image into R using the `stack()` function. We then plotted the data as a composite, RGB (and CIR) image using `plotRGB()`. However, sometimes data are downloaded in individual bands rather than a composite raster stack.

In this lesson we will learn how to work with Landsat data in R. In this case, our data are downloaded in .tif format with each file representing a single band rather than a stack of bands.

About Landsat data

> At over 40 years, the Landsat series of satellites provides the longest temporal record of moderate resolution multispectral data of the Earth's surface on a global basis. The Landsat record has remained remarkably unbroken, proving a unique resource to assist a broad range of specialists in managing the world's food, water, forests, and other natural resources for a growing world population. It is a record unmatched in quality, detail, coverage, and value. Source: USGS

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/TimelineOr
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/TimelineO
</a>
<figcaption>The 40 year history of landsat missions. Source: USGS - https://landsat.usgs.gov/landsat-mi
</figcaption>
```

Landsat data is a spectral dataset, collected from space. As we discussed in the first lesson, the spectral bands and associated spatial resolution of each band is listed below.

Landsat 8 Bands

Band	Wavelength range (nanometers)	Spatial Resolution (m)	Spectral Width (nm)
Band 1 - Coastal aerosol	430 - 450	30	2.0
Band 2 - Blue	450 - 510	30	6.0
Band 3 - Green	530 - 590	30	6.0
Band 4 - Red	640 - 670	30	0.03
Band 5 - Near Infrared (NIR)	850 - 880	30	3.0
Band 6 - SWIR 1	1570 - 1650	30	8.0
Band 7 - SWIR 2	2110 - 2290	30	18
Band 8 - Panchromatic	500 - 680	15	18
Band 9 - Cirrus	1360 - 1380	30	2.0

Understanding landsat

When working with landsat, it is important to understand both the metadata and the file naming convention. The metadata tell us about how the data were processed, where the data are from and how they are structured.

The file names, tell us what sensor collected the data, the date the data were collected, and more.

Landsat file naming convention

```
<a href="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/Collection
<img src="~/Documents/Github/earthlab.github.io/images/course-materials/earth-analytics/week-6/Collecti
</a>
<figcaption>Landsat file names Source: USGS Landsat - https://landsat.usgs.gov/what-are-naming-conventi
</figcaption>
```

Landsat tif files in R

Now that we understand how our file is named.

```
# load spatial packages
library(raster)
library(rgdal)
library(rgeos)
# turn off factors
options(stringsAsFactors = F)
```

If we look at the directory that contains our landsat data, we will see that each of the individual bands is stored individually as a geotiff rather than being stored as a stacked or layered raster.

Why would they store the data this way?

Conventionally landsat was stored in a file format called HDF - hierarchical data format. However that format, while extremely efficient, is a bit more challenging to work with. In recent years USGS has started to make each band of a landsat scene available as a .tif file. This makes it a bit easier to use across many different programs and platforms.

We have already been working with the geotiff file format in this class! We will thus use many of the same functions we used previously, to work with Landsat.

Get list of files

To begin, let's explore our file directory in R, We can use `list.files()` to grab a list of all files within any directory on our computer.

```
# get list of all tifs
list.files("data/week6/landsat/LC80340322016205-SC20170127160728/crop")
##  [1] "LC80340322016205LGN00_bqa_crop.tif"
##  [2] "LC80340322016205LGN00_cfmask_conf_crop.tif"
##  [3] "LC80340322016205LGN00_cfmask_crop.tif"
##  [4] "LC80340322016205LGN00_sr_band1_crop.tif"
##  [5] "LC80340322016205LGN00_sr_band2_crop.tif"
##  [6] "LC80340322016205LGN00_sr_band3_crop.tif"
##  [7] "LC80340322016205LGN00_sr_band4_crop.tif"
##  [8] "LC80340322016205LGN00_sr_band5_crop.tif"
##  [9] "LC80340322016205LGN00_sr_band6_crop.tif"
## [10] "LC80340322016205LGN00_sr_band7_crop.tif"
```

```
## [11] "LC80340322016205LGN00_sr_cloud_crop.tif"
## [12] "LC80340322016205LGN00_sr_ipflag_crop.tif"
```

We can also use list.files with the pattern argument. This allows us to specify a particular pattern that further subsets our data. In this case, we just want to look at a list of files with the extention: `.tif`. Note that it is important that the file **ends with** .tif. So we use the dollar sign at the end of our pattern to tell R to only grab files that end with .tif.

```
pattern=".tif$"
```

```
# but really we just want the tif files
all_landsat_bands <- list.files("data/week6/Landsat/LC80340322016205-SC20170127160728/crop",
                      pattern=".tif$",
                      full.names = T) # make sure we have the full path to the file
all_landsat_bands
##  [1] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_bqa_crop.tif"
##  [2] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_cfmask_conf_cr
##  [3] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_cfmask_crop.ti
##  [4] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band1_crop.
##  [5] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band2_crop.
##  [6] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band3_crop.
##  [7] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band4_crop.
##  [8] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band5_crop.
##  [9] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band6_crop.
## [10] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band7_crop.
## [11] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_cloud_crop.
## [12] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_ipflag_crop
```

Above, we use the $ after .tif to tell R to look for files that end with .tif. This is a good start but there is one more condition that we'd like to meet. We only want the .tif files that are spectral bands. Notice that some of our files have text that includes "mask", flags, etc. Those are all additional layers that we don't need right now. We just need the spectral data saved in bands 1_7.

Thus, we want to grab all bands that both end with `.tif` AND contain the text "band" in them. To do this we use the function `glob2rx()` which allows us to specify both conditions. Here we tell R to select all files that have the word **band** in the filename. We use a * sign before and after band because we don't know exactly what text will occur before or after band. We use `.tif$` to tell R that each file needs to end with .tif.

```
all_landsat_bands <- list.files("data/week6/Landsat/LC80340322016205-SC20170127160728/crop",
           pattern=glob2rx("*band*.tif$"),
           full.names = T) # use the dollar sign at the end to get all files that END WITH
all_landsat_bands
## [1] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band1_crop.t
## [2] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band2_crop.t
## [3] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band3_crop.t
## [4] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band4_crop.t
## [5] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band5_crop.t
## [6] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band6_crop.t
## [7] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band7_crop.t
```

Now we have a list of all of the landsat bands in our folder. We could chose to open each file individually using the `raster()` function.

```
# get first file
all_landsat_bands[2]
## [1] "data/week6/Landsat/LC80340322016205-SC20170127160728/crop/LC80340322016205LGN00_sr_band2_crop.t
landsat_band1 <- raster(all_landsat_bands[2])
```

Landsat cropped band 2
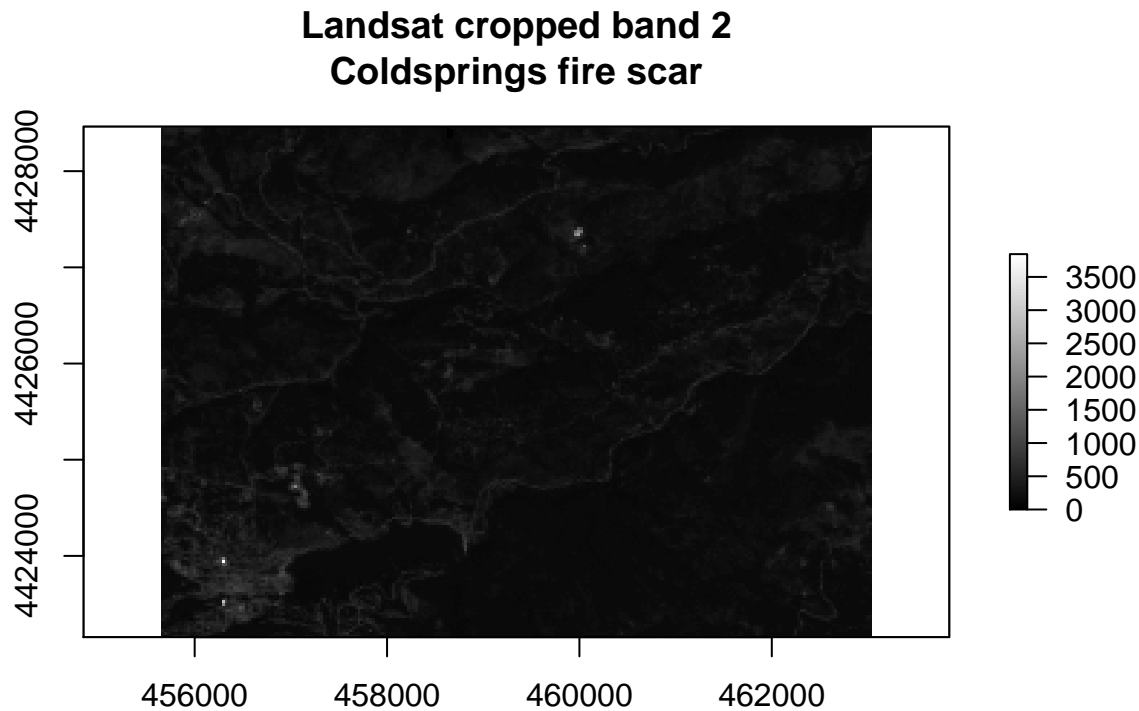**Coldsprings fire scar**

Figure 1:

```
plot(landsat_band1,
     main="Landsat cropped band 2\nColdsprings fire scar",
     col=gray(0:100 / 100))
```

However, that is not a very efficient approach. It's more efficiently to open all of the layers together as a stack. Then we can access each of the bands and plot / use them as we want. We can do that using the stack() function.

```
# stack the data
landsat_stack_csf <- stack(all_landsat_bands)
# view stack attributes
landsat_stack_csf
## class       : RasterStack
## dimensions  : 177, 246, 43542, 7  (nrow, ncol, ncell, nlayers)
## resolution  : 30, 30  (x, y)
## extent      : 455655, 463035, 4423155, 4428465  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=13 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
## names       : LC8034032//band1_crop, LC8034032//band2_crop, LC8034032//band3_crop, LC8034032//band4_
## min values  :                     0,                     0,                     0,
## max values  :                  3488,                  3843,                  4746,
```

Let's plot each individual band in our stack.

```
plot(landsat_stack_csf,
     col=gray(20:100 / 100))
```

```
# get list of each layer name
names(landsat_stack_csf)
## [1] "LC80340322016205LGN00_sr_band1_crop"
## [2] "LC80340322016205LGN00_sr_band2_crop"
```
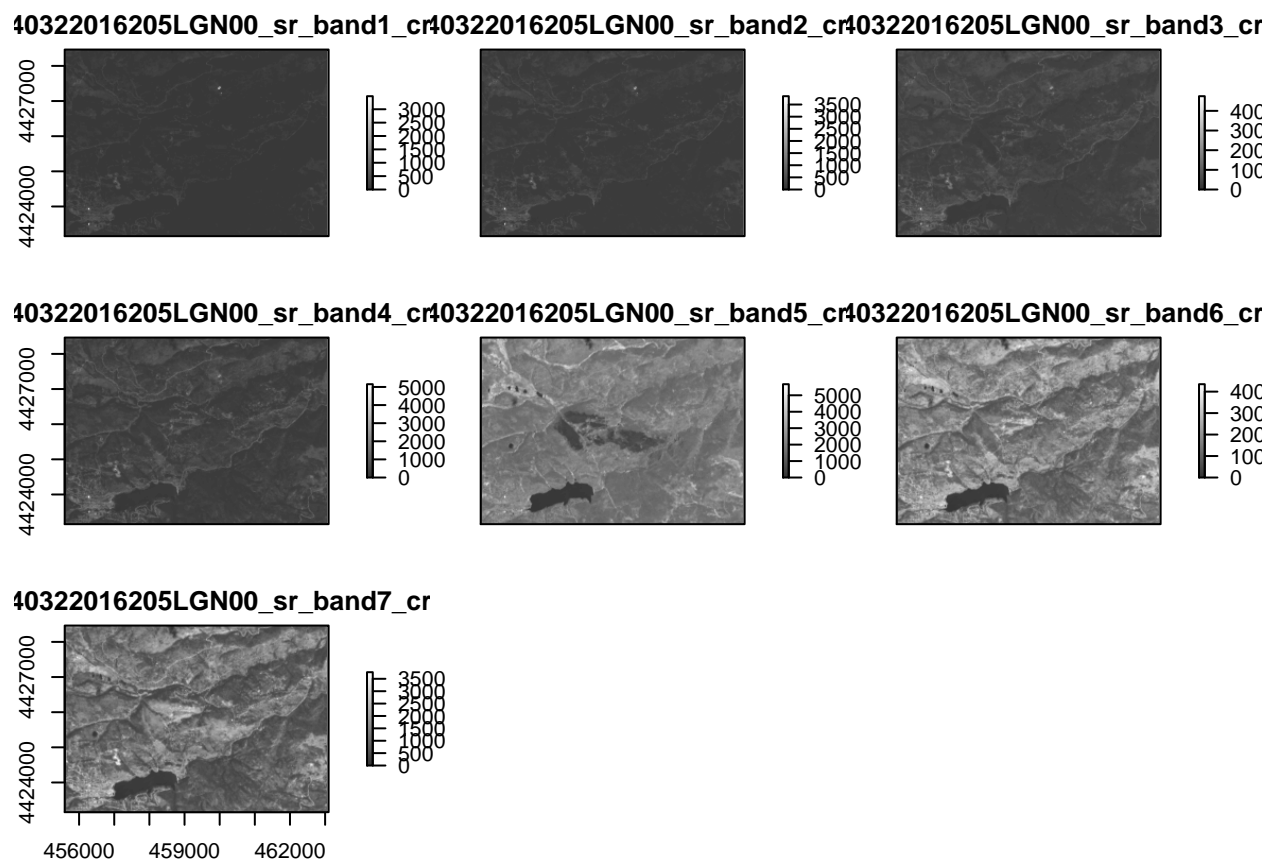
4

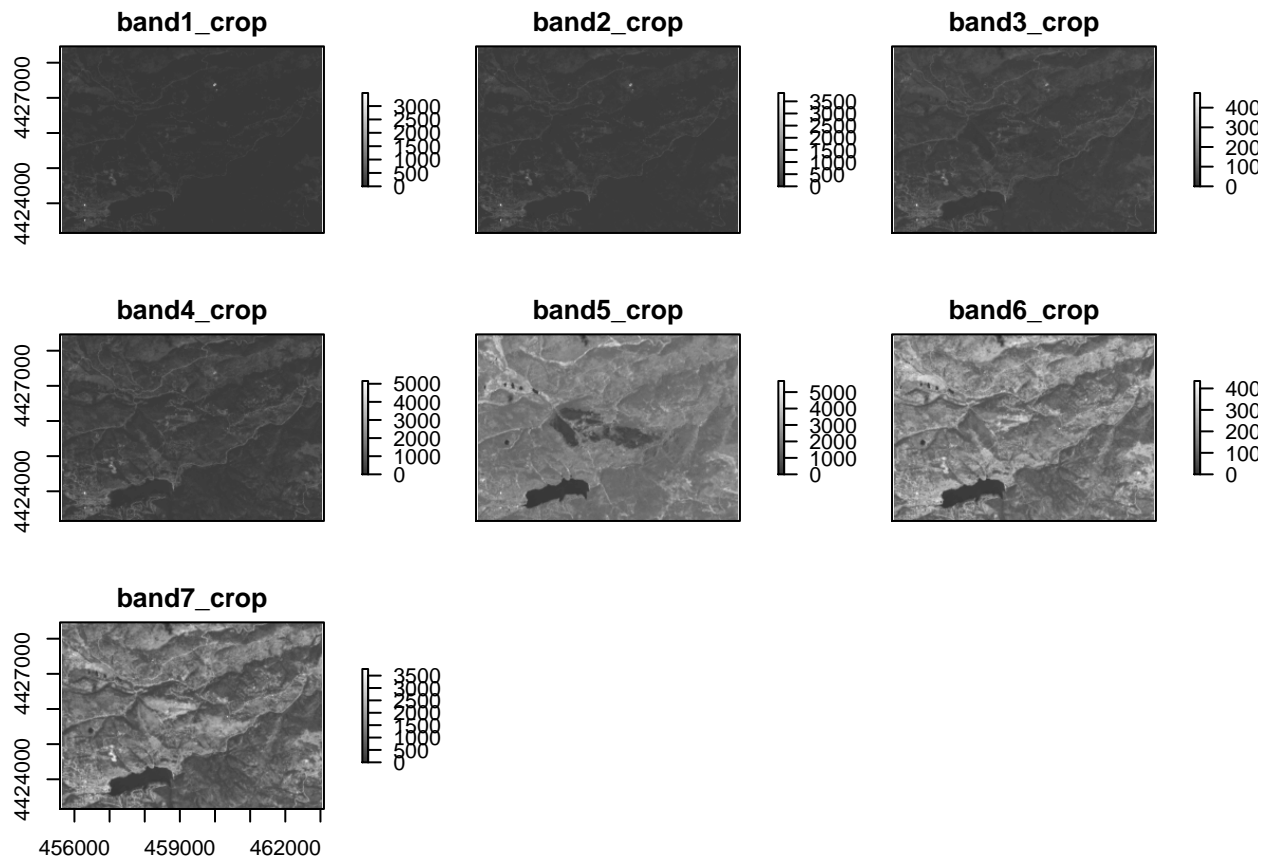Figure 2: plot individual landsat bands

Figure 3: plot individual landsat bands good names

```
## [3] "LC80340322016205LGN00_sr_band3_crop"
## [4] "LC80340322016205LGN00_sr_band4_crop"
## [5] "LC80340322016205LGN00_sr_band5_crop"
## [6] "LC80340322016205LGN00_sr_band6_crop"
## [7] "LC80340322016205LGN00_sr_band7_crop"
# remove the filename from each band name for pretty plotting
names(landsat_stack_csf) <- gsub(pattern = "LC80340322016205LGN00_sr_", replacement = "", names(landsat_
plot(landsat_stack_csf,
     col=gray(20:100 / 100))
```

Plot RGB image

Next, let's plot an RGB image using landsat. Refer to the landsat bands in the table at the top of this page to figure out the red, green and blue bands. Or read the ESRI landsat 8 band combinations post.

```
par(col.axis="white", col.lab="white", tck=0)
plotRGB(landsat_stack_csf,
     r=4, g=3, b=2,
     stretch="lin",
     axes=T,
     main="RGB composite image\n Landsat Bands 4, 3, 2")
box(col="white")
```

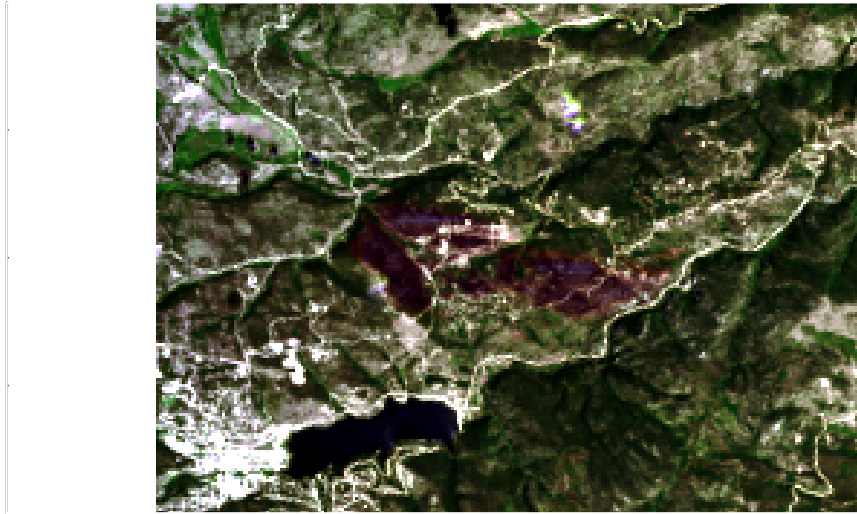**RGB composite image**
Landsat Bands 4, 3, 2

Figure 4: plot rgb composite

Now we've created a red, green blue color composite image. Remember this is what our eye would see. What happens if we plot the near infrared band instead of red? Try the following combination:

```r
par(col.axis="white", col.lab="white", tck=0)
plotRGB(landsat_stack_csf,
    r=5, g=4, b=3,
    stretch="lin",
    axes=T,
    main="Color infrared composite image\n Landsat Bands 5, 4, 3")
box(col="white")
```

optional challenge

Using the ESRI landsat 8 band combinations post as a guide. Plot the following landsat band combinations :
* false color * color infrared * agriculture * healthy vegetation

Be sure to add a title to each of your plots.

optional challenge 2

Using the ESRI landsat 8 band combinations post as a guide. Plot the following landsat band combinations :
* false color * color infrared * agriculture * healthy vegetation

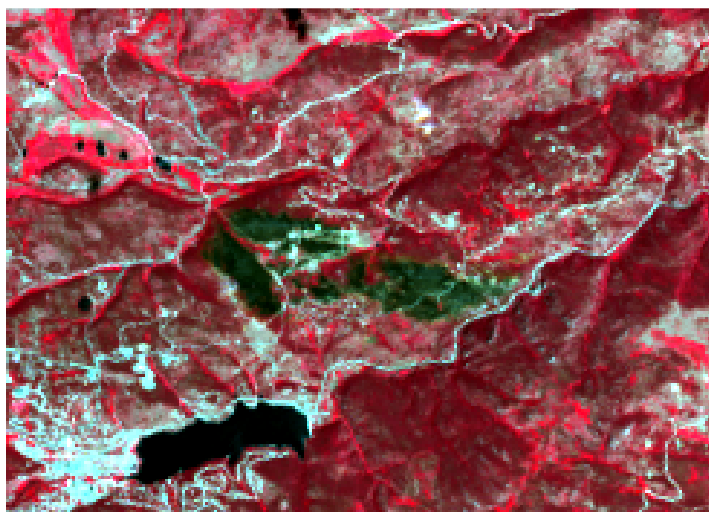Be sure to add a title to each of your plots.

**Color infrared composite image
Landsat Bands 5, 4, 3**

Figure 5: plot rgb composite