An example of creating modular code in R - Efficient scientific programming

Learning Objectives

After completing this tutorial, you will be able to:

- Describe how functions can make your code easier to read / follow

What you need

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

{% include/data_subsets/course_earth_analytics/_data-week6-7.md %}

```r
# set working dir
setwd("~/Documents/earth-analytics")

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

```r
# get list of tif files
all_landsat_bands <- list.files("data/week6/Landsat/LC80340322016189-SC20170128091153/crop",
                                pattern=glob2rx("*band*.tif$"),
                                full.names = T)

# stack the data (create spatial object)
landsat_stack_csf <- stack(all_landsat_bands)

par(col.axis="white", col.lab="white", tck=0)
# plot brick
plotRGB(landsat_stack_csf,
  r=4,g=3, b=2,
  main="RGB Landsat Stack \n pre-fire",
  axes=T,
  stretch="hist")
box(col="white") # turn all of the lines to white
```

```r
# we can do the same things with functions
get_stack_bands <- function(the_dir_path, the_pattern){
  # get list of tif files
  all_landsat_bands <- list.files(the_dir_path,
                                pattern=glob2rx(the_pattern),
                                full.names = T)

  # stack the data (create spatial object)
  landsat_stack_csf <- stack(all_landsat_bands)
```

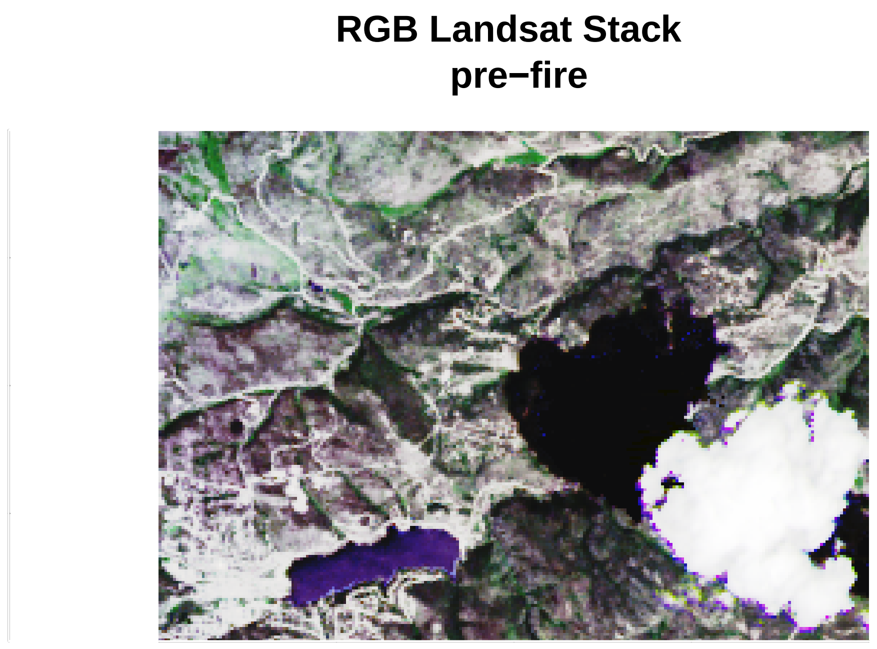

Figure 1: landsat pre fire raster stack plot

```
  return(landsat_stack_csf)

}
```

Example using functions

Here's we've reduced the code by a few lines using a get bands function. Then we can plot like we did before.

```
# code to go here
landsat_pre_fire <- get_stack_bands(the_dir_path = "data/week6/Landsat/LC80340322016189-SC2017012809115
                the_pattern = "*band*.tif$")


par(col.axis="white", col.lab="white", tck=0)
# plot brick
plotRGB(landsat_pre_fire,
  r=4,g=3, b=2,
  main="RGB Landsat Stack \n pre-fire",
  axes=T,
  stretch="lin")
box(col="white") # turn all of the lines to white
```

Now, what if we created a function that adjusted all of the parameters that we wanted to set to plot an RGB image? Here we will require the user to send the function a stack with the bands in the order that they want to plot the data.

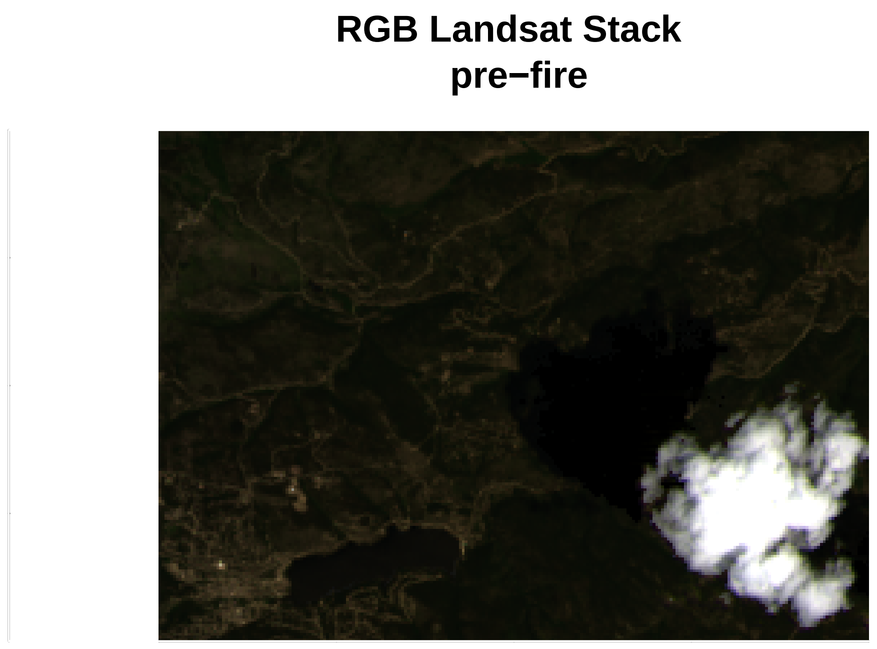

Figure 2: landsat pre fire raster stack plot

```r
# notice here i set a default stretch to blank
create_rgb_plot <-function(a_raster_stack, the_plot_title, r=3, g=2, b=1, the_stretch=NULL){
  # this function plots an RGB image with a title
  # it sets the plot border and box to white
  # Inputs a_raster_stack - a given raster stack with multiple spectral bands
  # the_plot_title - teh title of the plot - text string format in quotes
  # red, green, blue - the numeric index location of the bands that you want
  #  to plot on the red, green and blue channels respectively
  # the_stretch -- defaults to NULL - can take "hist" or "lin" as an option
  par(col.axis="white", col.lab="white", tck=0)
  # plot brick
  plotRGB(a_raster_stack,
    main=the_plot_title,
    r=r, g=g, b=b,
    axes=T,
    stretch=the_stretch)
  box(col="white") # turn all of the lines to white

}
```

Let’s use the code to plot pre-fire RGB image.

```r
# code to go here
landsat_pre_fire <- get_stack_bands(the_dir_path = "data/week6/Landsat/LC80340322016189-SC20170128091153
                the_pattern = "*band*.tif$")

# plot the data
create_rgb_plot(a_raster_stack = landsat_pre_fire,
                r=4, g = 3, b=2,
```

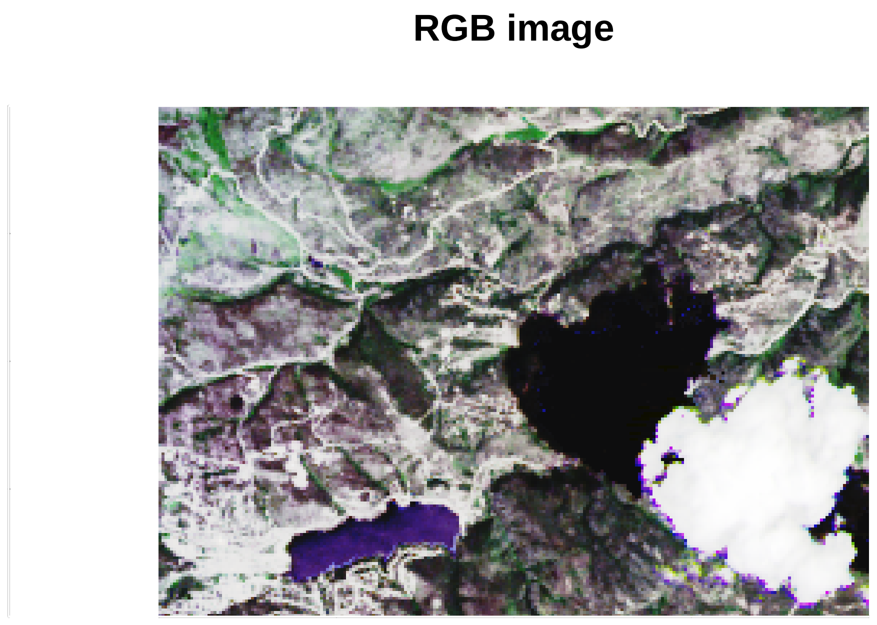

Figure 3: pre-fire rgb image

```
                 the_plot_title = "RGB image",
                 the_stretch="hist")
```

Once our plot parameters are setup, we can use the same code to plot our data over and over without having to set parameters each time!

Now we can plot a CIR fire image with one function!

```
# plot the data
create_rgb_plot(a_raster_stack = landsat_pre_fire,
                 r=5, g = 4, b = 3,
                 the_plot_title = "RGB image",
                 the_stretch="hist")
```

Let's run the same functions on another landsat dataset - post fire.

```
# create stack
landsat_post_fire <- get_stack_bands(the_dir_path = "data/week6/Landsat/LC80340322016205-SC2017012716072
                 the_pattern = "*band*.tif$")

# plot the 3 band image of the data
create_rgb_plot(a_raster_stack = landsat_post_fire,
                 r=4, g = 3, b=2,
                 the_plot_title = "RGB image",
                 the_stretch="hist")
```

What if we want to plot a CIR image post fire?

```
# plot the 3 band image of the data
create_rgb_plot(a_raster_stack = landsat_post_fire,
                 r=5, g = 4, b = 3,
                 the_plot_title = "Landsat post fire CIR image",
```

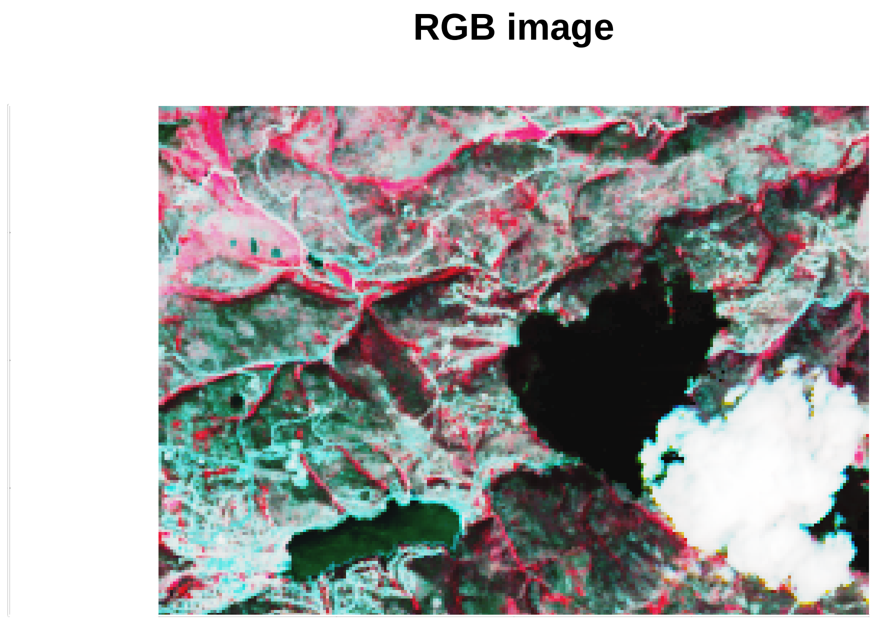

Figure 4: pre-fire cir image

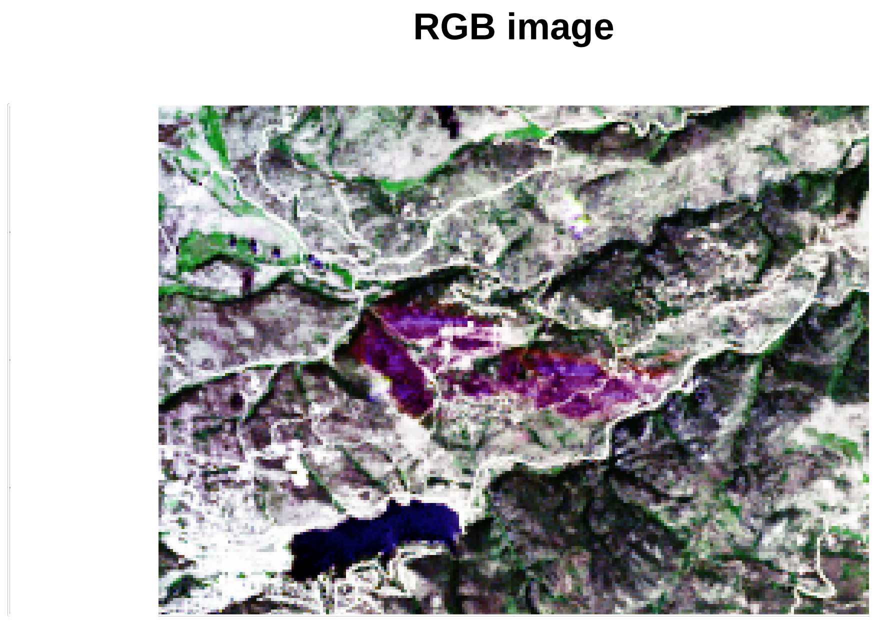

Figure 5: landsat post fire raster stack plot

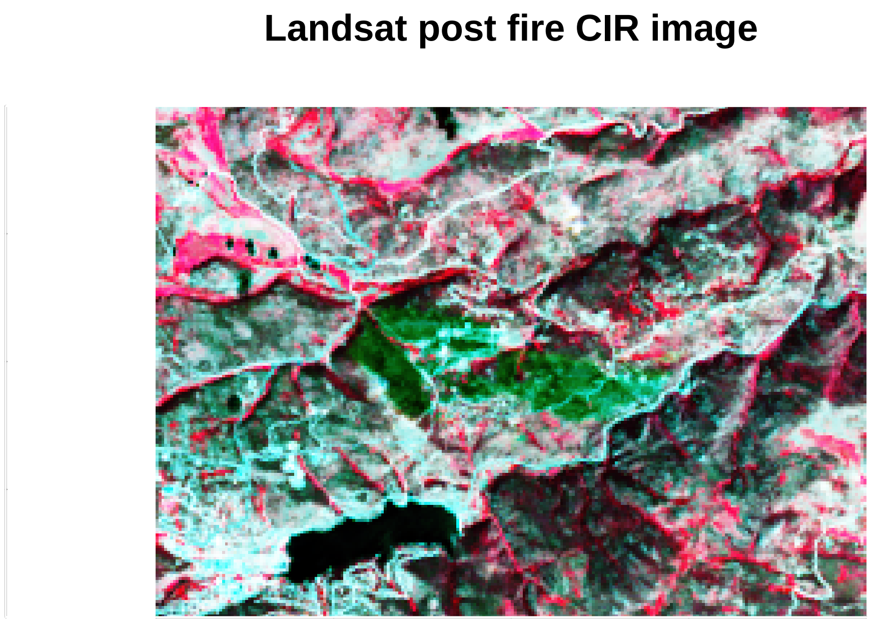

Figure 6: landsat CIR post fire raster stack plot

```
                the_stretch="hist")
```

Are our functions general enough to work with MODIS?

```
# import MODIS
modis_pre_fire <- get_stack_bands(the_dir_path = "data/week6/modis/reflectance/07_july_2016/crop",
                the_pattern = "*sur_refl*.tif$")

# plot the data
create_rgb_plot(a_raster_stack = modis_pre_fire,
                r=1, g = 4, b=3,
                the_plot_title = "MODIS RGB image",
                the_stretch="hist")
```

Looks like it works!

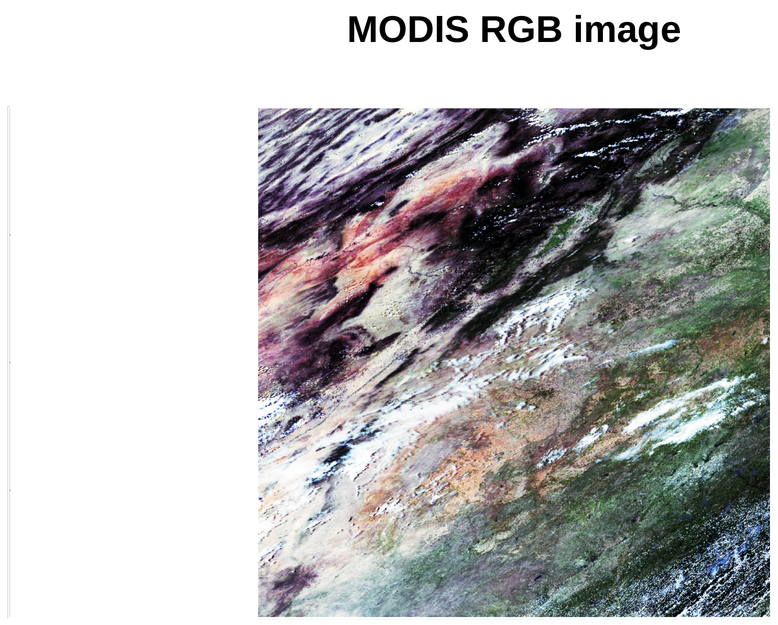

Figure 7: pre-fire rgb image MODIS