Refine spatial plots in R.

Learning Objectives

After completing this tutorial, you will be able to:

- Remove borders and refine the size of plots in an output rmarkdown report.
- Adjust the aspect ratio of a plot rendered to a pdf using knitr.
- Customize the location of legends in a plot in R.

What you need

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

Download Week 6 Data (~500 MB){:data-proofer-ignore=" .btn }

In the previous lessons, we opened landsat and MODIS data in R. In this lesson, we will learn how to refine our plots in R to make our report look nicer and in turn more professional. First, let's import some data.

```
# import landsat data
all_landsat_bands <- list.files("data/week6/Landsat/LC80340322016189-SC20170128091153/crop",
           pattern=glob2rx("*band*.tif$"),
           full.names = T) # use the dollar sign at the end to get all files that END WITH
# create spatial stack
all_landsat_bands_st <- stack(all_landsat_bands)
```

Titles using plotRGB()

You can try to add a title but it won't plot

```
plotRGB(all_landsat_bands_st,
        r=4,b=3,g=2,
        stretch="hist",
        main="title here")
```

If we add the `axes=T` argument to our plot, our title plots but we also get the x and y axis in black which doesn't look nice. We don't need those. As a work around we can set the x and y axis labels to plot using "white" (`col.axis="white"`). Also we turn off the tick marks using `tck=0`.

```
# adjust the parameters so the axes colors are white. Also turn off tick marks.
par(col.axis="white", col.lab="white", tck=0)
# plot
plotRGB(all_landsat_bands_st,
        r=4,b=3,g=2,
        stretch="hist",
        main="title here",
        axes=T)
```

The final step is to turn off box which leaves that annoying line on the left hand side and bottom of the plot.

```
# adjust the parameters so the axes colors are white. Also turn off tick marks.
par(col.axis="white", col.lab="white", tck=0)
# plot
plotRGB(all_landsat_bands_st,
        r=4,b=3,g=2,
```

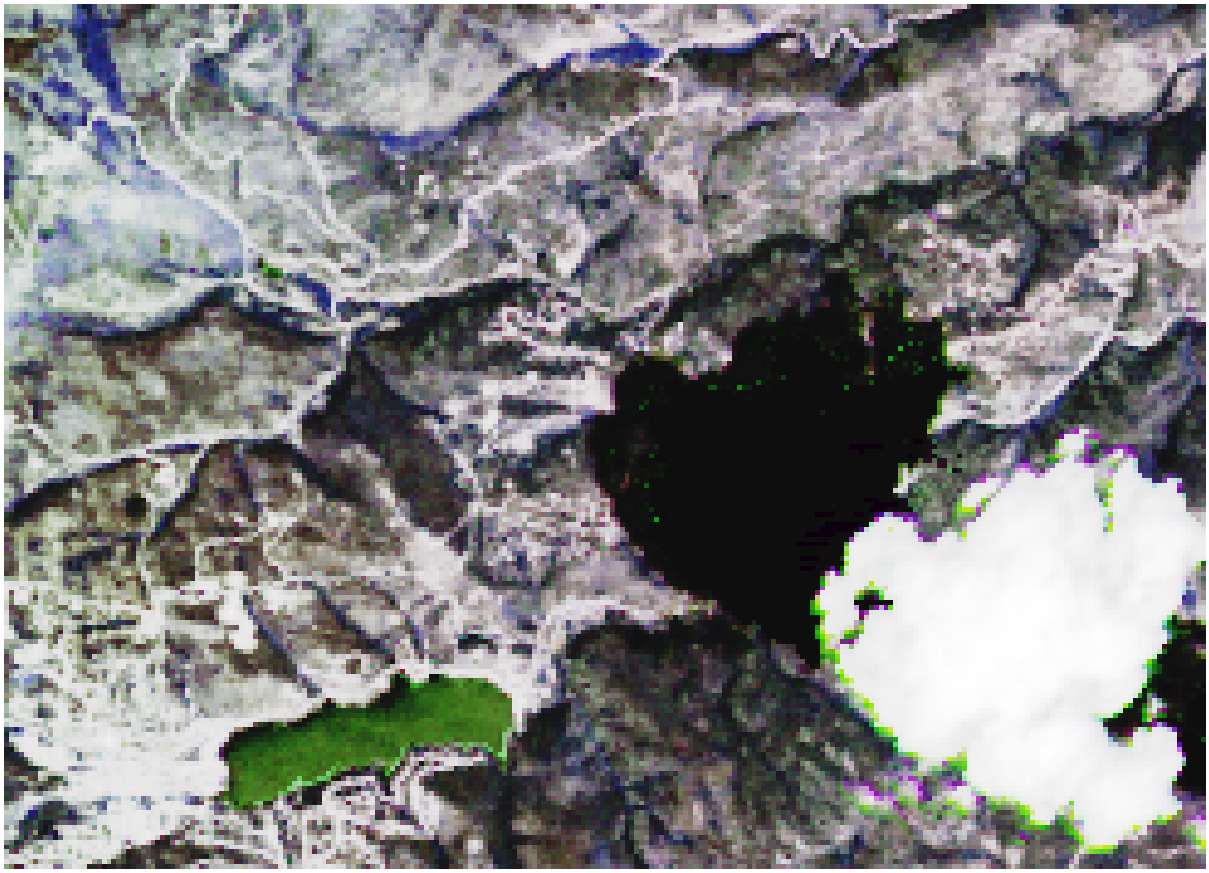

Figure 1: Remove axes labels.

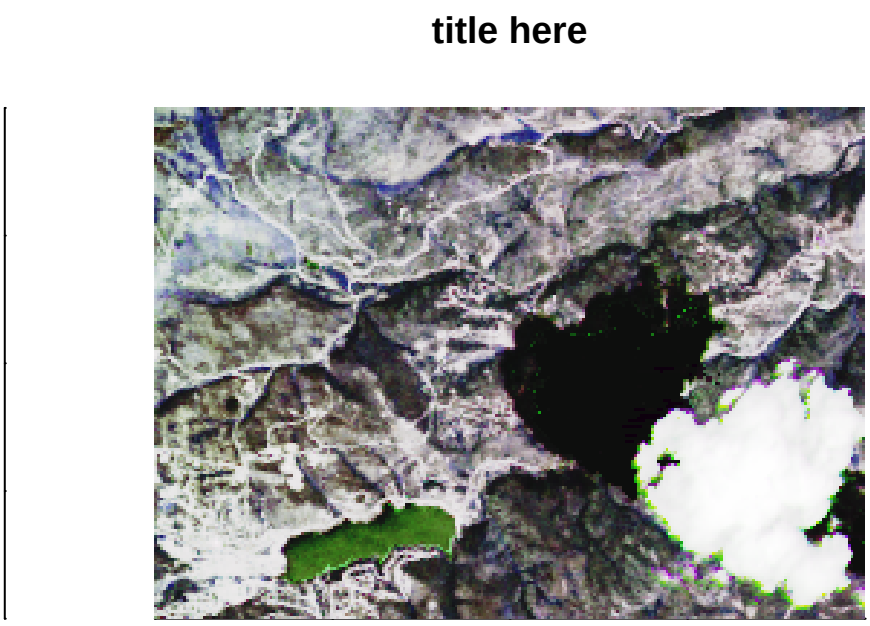

Figure 2: Remove axes labels.

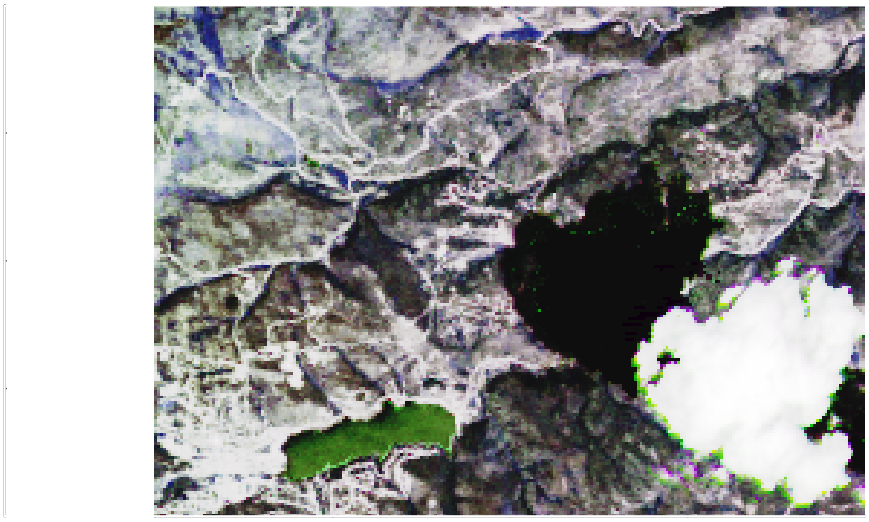

Figure 3: Remove axes labels.

```
        stretch="hist",
        main="title here",
        axes=T)
# set bounding box to white as well
box(col="white") # turn all of the lines to white
```

This looks nice, but now the plot itself is too tall. There is extra white space above and below the plot that we don't need. This is because the `dev` space where R studio renders plots is set to a particular size based upon how you've adjusted it and your monitor resolution. I can account for this too - using the code chunk arguments: `fig.width` and `fig.height`.

In my plot below, I used `fig.width=7, fig.height=6` in my code chunk arguments. The units are in inches.

My code chunk looks like this: `{r plot-rgb4, fig.cap="Adjust figure width and height.", fig.width=7, fig.height=6}`

```
# adjust the parameters so the axes colors are white. Also turn off tick marks.
par(col.axis="white", col.lab="white", tck=0)
# plot
plotRGB(all_landsat_bands_st,
        r=4,b=3,g=2,
        stretch="hist",
        main="title here",
        axes=T)
# set bounding box to white as well
box(col="white") # turn all of the lines to white
```

Notice that now my plot has less white space above and below the image. This is because it's no longer plotting using a square aspect ratio - we've adjusted that! You can experiment with different fig width and height values depending upon the aspect ratio of the plot that you are trying to print to your report.

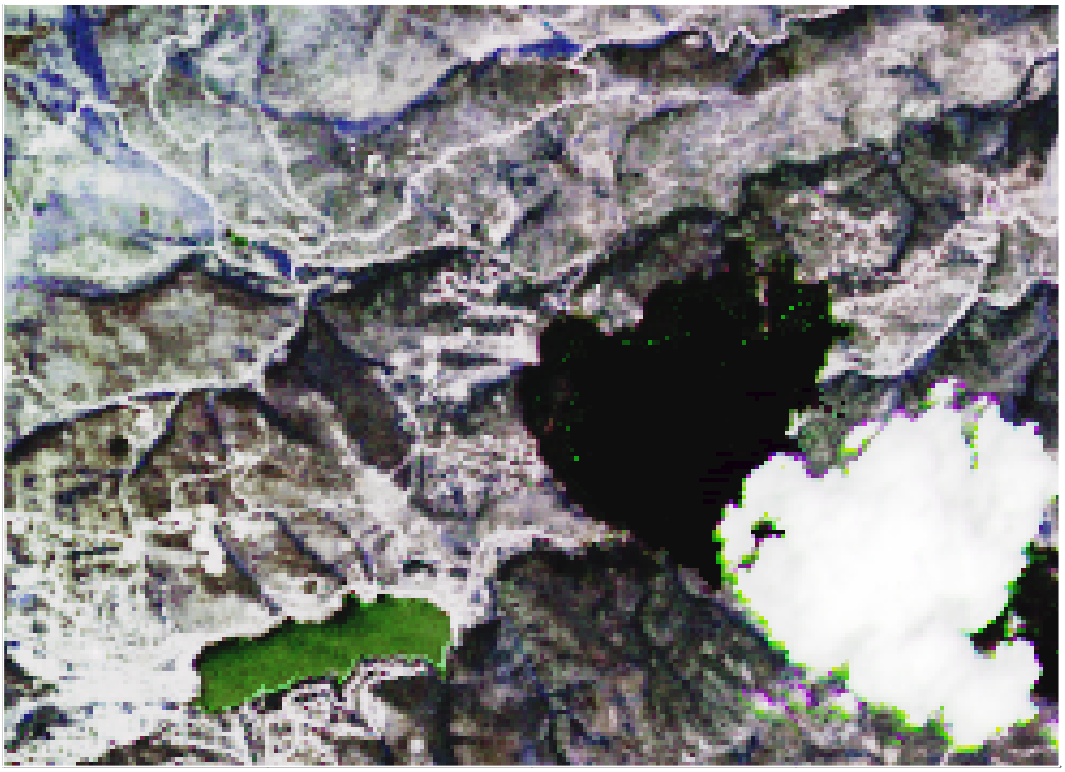

Figure 4: Adjust figure width and height.

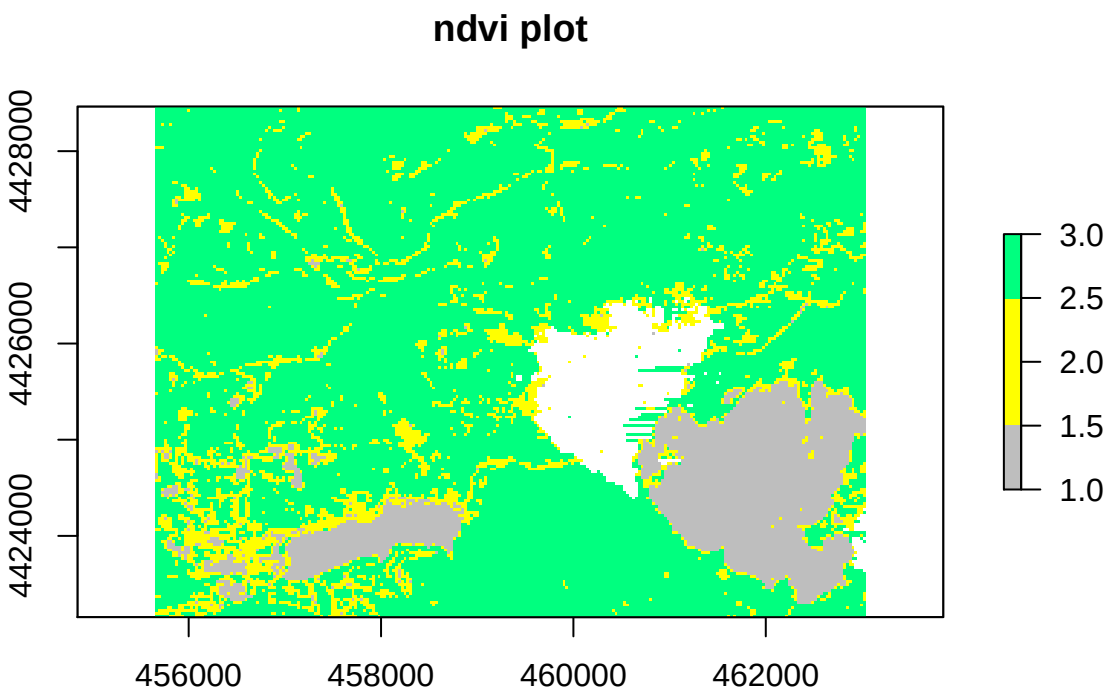

Figure 5: ndvi plot - no legend

Reset dev

Any time you modify the parameters, you should consider resetting the `dev()` space. Why? Because if you plot another plot in R, it will use the parameters that you set previously! In the case above we set the axes to white. This setting will become a global setting until you clear your plot space! To clear the plot dev space programmatically, use `dev.off()`.

```
# reset dev (space where plots are rendered in RStudio)
dev.off()
```

Adjusting legends

Legends can also be trick to figure out in R. Take a look at the plot below. It's not pretty. For one, we don't need the x and y axes on this plot.

```
# plot ndvi with legend
plot(ndvi_classified,
     main="ndvi plot",
     col=the_colors)
```

First let's get rid of the unnecessary axes and turn off the legend. We can remote the axes & box that surrounds our image using: `axes=F` and `box=F`.

```
# plot ndvi with legend
plot(ndvi_classified,
     main="ndvi plot",
     col=the_colors,
     axes=F, box=F)
```

Next, we turn off the legend and add our own legend. However, the legend isn't exactly where we want it to

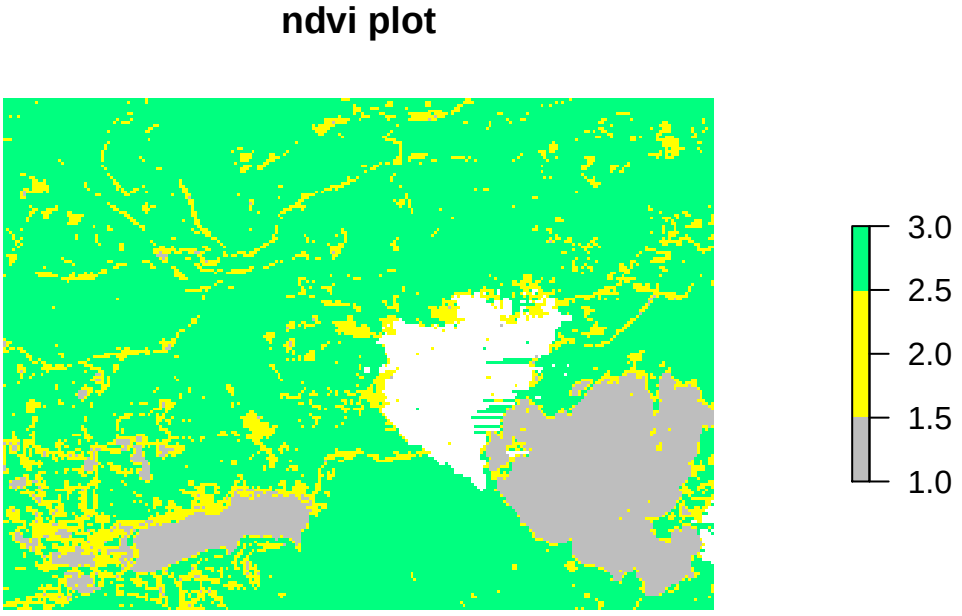

Figure 6: ndvi plot - no legend

be here. It's above the plot and we'd like it to be to the right of the plot.

```
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="ndvi plot",
     col=the_colors,
     axes=F, box=F)
legend("topright",
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       fill= the_colors)
```

We can force the legend to plot outside of our axes using the parameter `xpd=T`. We can locate the legend in the upper right hand corner OUTSIDE of our plot by specifying the max x and y values derived from the `extent` of the spatial object that we are plotting

Here I set the x max value to me the **furthest east** hand corner of my object extent. `x = ndvi_classified@extent@xmax`

Here I set the y max value to me the **furthest north** of my object extent. `y = ndvi_classified@extent@ymax`

```
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="ndvi plot",
     col=the_colors,
     axes=F, box=F)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       fill= rev(the_colors)) # use rev to reverse the order of colors for the legend
```

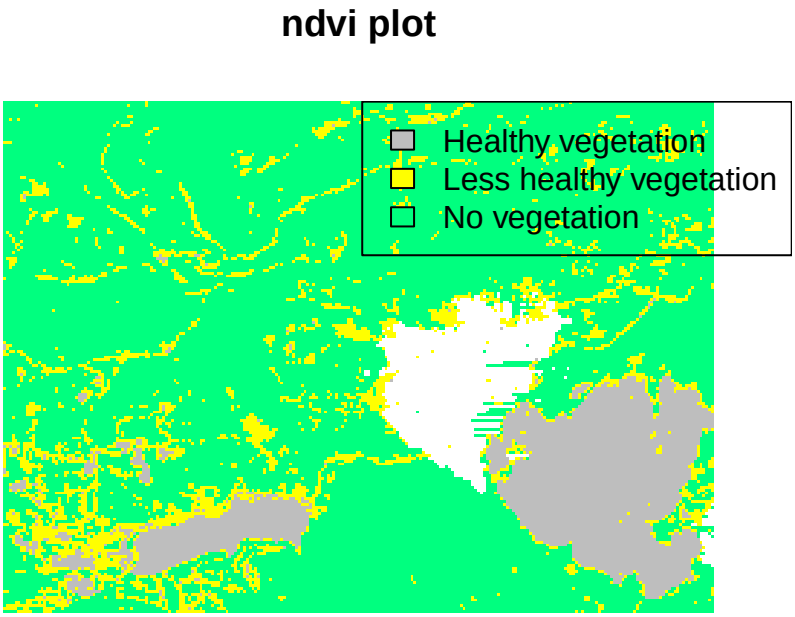

Figure 7: ndvi plot - no legend

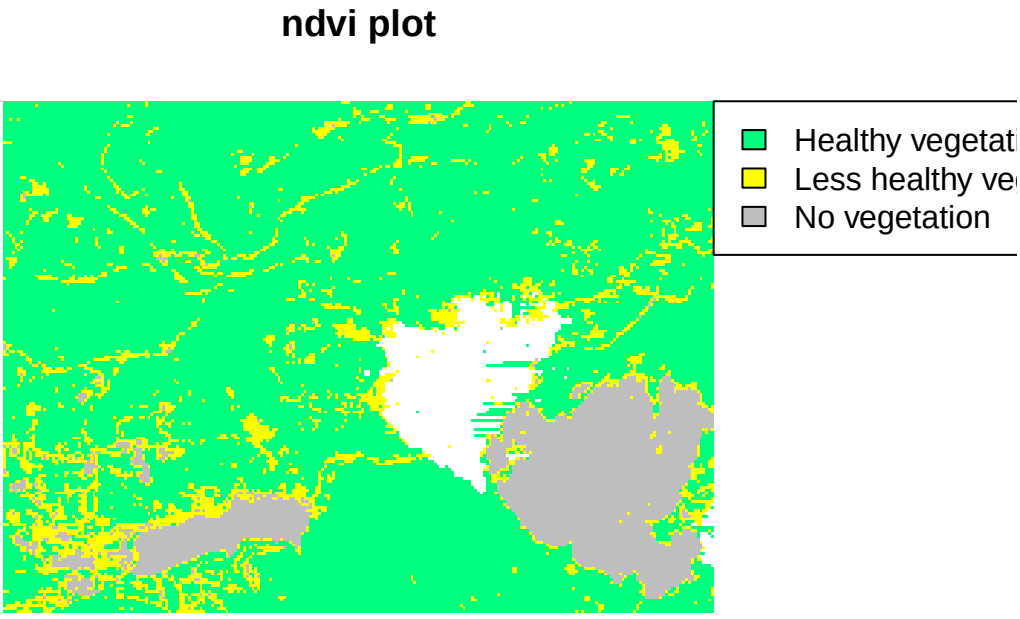

Figure 8: plot with legend in the upper right.

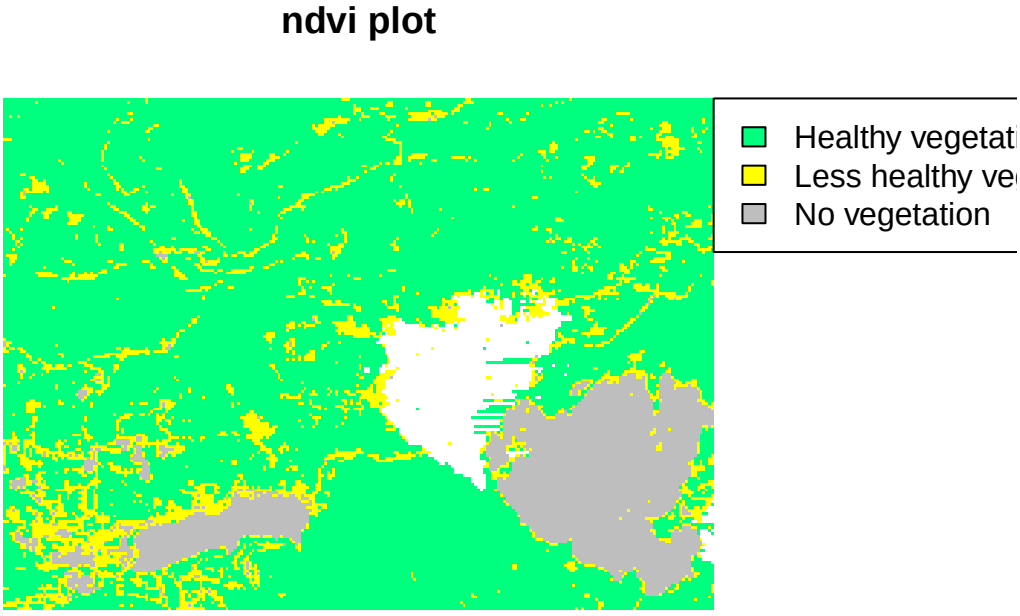

Figure 9: plot with legend in the upper right.

Now, another problem with our legend. *The order of our colors is all wrong:* Grey should represent "no vegetation" and green should represent health vegetation. We can use `rev()` on our list of colors to reverse the order of colors drawn on the legend.

```
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="ndvi plot",
     col=the_colors,
     axes=F, box=F)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       fill= rev(the_colors)) # use rev to reverse the order of colors for the legend
```

On to the pesky white space on either side of the plot. There are several ways to handle this. One is by specifying margins for out plot.

Notice in the plot above that there is too much white space above and below the plot. In the case of my plot, the aspect ratio of width:height is not right. We want the height to be SMALLER to remove some of the white space. The white space is there because R is trying to plot our figure using a 1:1 aspect ratio. We can use the `mar` argument to adjust the white space on the bottom, left, top, or right sides of the plot. This makes room for our legend.

Below, we adjusted the top margin to `2` and the right margin to `5`. This makes space for our legend but also makes a bit of space for our plot title.

```
# set a margin for our figure
par(xpd=F, mar=c(0,0,2,5))
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
```

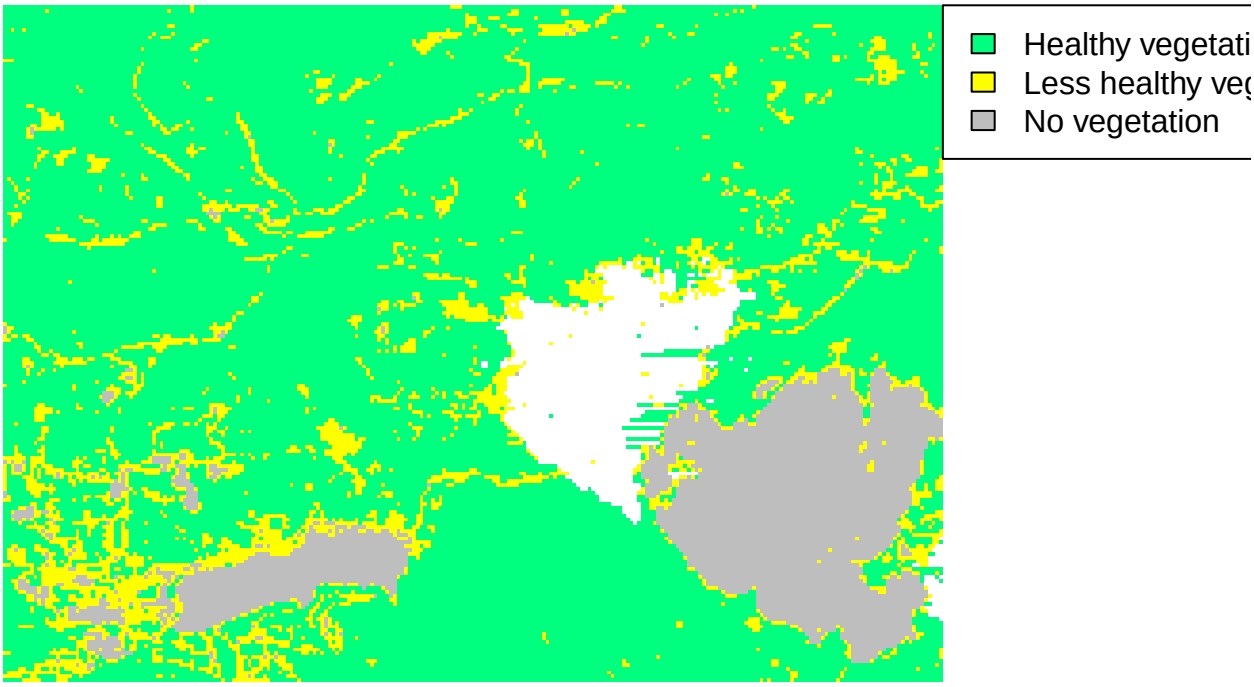

Figure 10: plot with legend in the upper right.

```
     main="ndvi plot with axes & box turned off",
     col=the_colors,
     axes=F,
     box=F)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       bty=F, # turn off legend border
       fill= rev(the_colors)) # use rev to reverse the order of colors for the legend
```

```
dev.off()
## null device
##           1
```

I can do better than that however. That box around the legend is annoying. Let's remove it using the legend argument: `bty="n"`. Let's also make the legend fonts a bit smaller using the argument `cex=.9`.

```
# set a margin for our figure
par(xpd=F, mar=c(0,0,2,5))
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="ndvi plot with axes & box turned off",
     col=the_colors,
     axes=F,
```

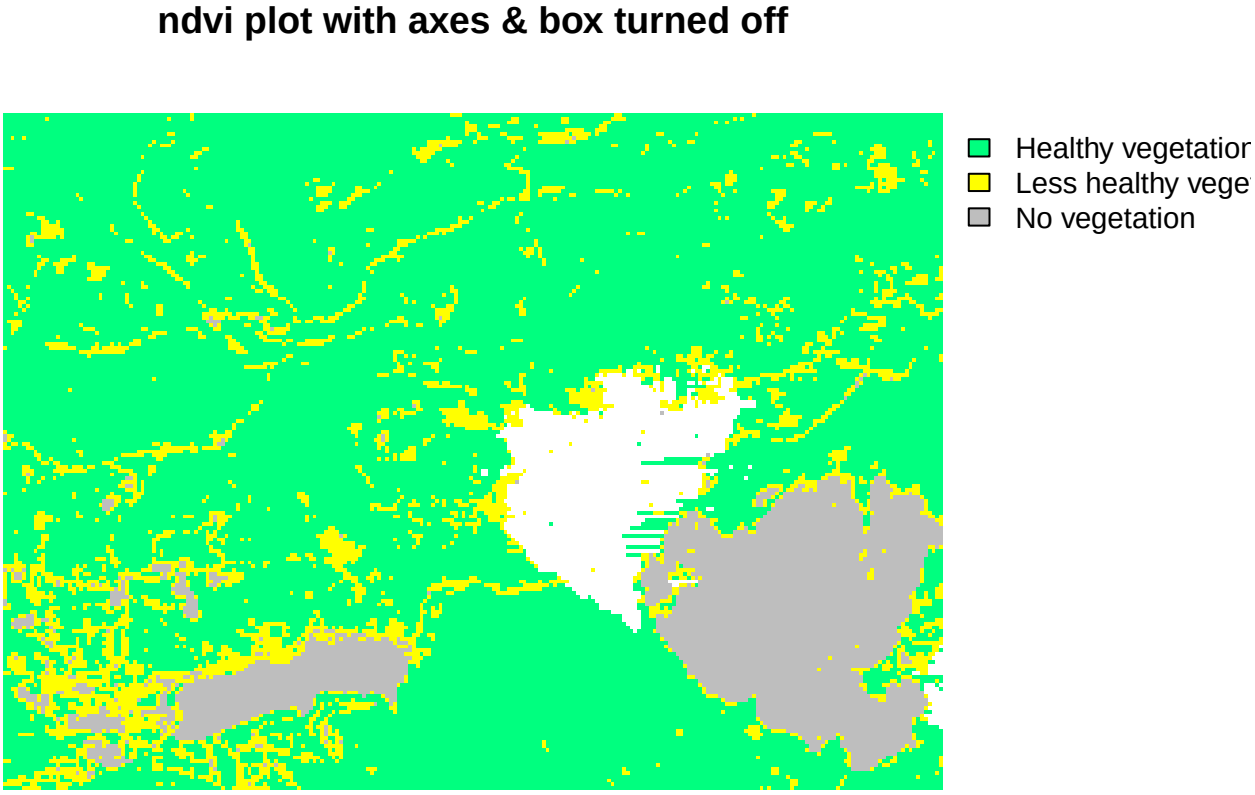

Figure 11: plot with legend in the upper right.

```
     box=F)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       fill= rev(the_colors),# use rev to reverse the order of colors for the legend
       bty="n", # turn off legend border
       cex=.9)  # adjust legend font size
```

```
dev.off()
## null device
##           1
```

If things are still not looking right, we can adjust the size of our output figure. Try adding the arguments fig.width and fig.height to your code chunk. `{r fix-plot-legend3, fig.cap="plot with legend in the upper right.", fig.width=6, fig.height=4}`

I use these values because i want the HEIGHT of the figure to be smaller. I want the WIDTH of the figure to be "landscape" or wider so i'll leave that value alone.

Below, I set `fig.width=6`, `fig.height=4` in my code chunk. setting a smaller height pulled the title down closer to my plot. I've also decreased the font size of my legend using the `cex` legend argument. Note that you may have to play with the figure size a bit to get it just right.

HINT: use 'dev.size()" to figure out the size of your plot dev space in RStudio.

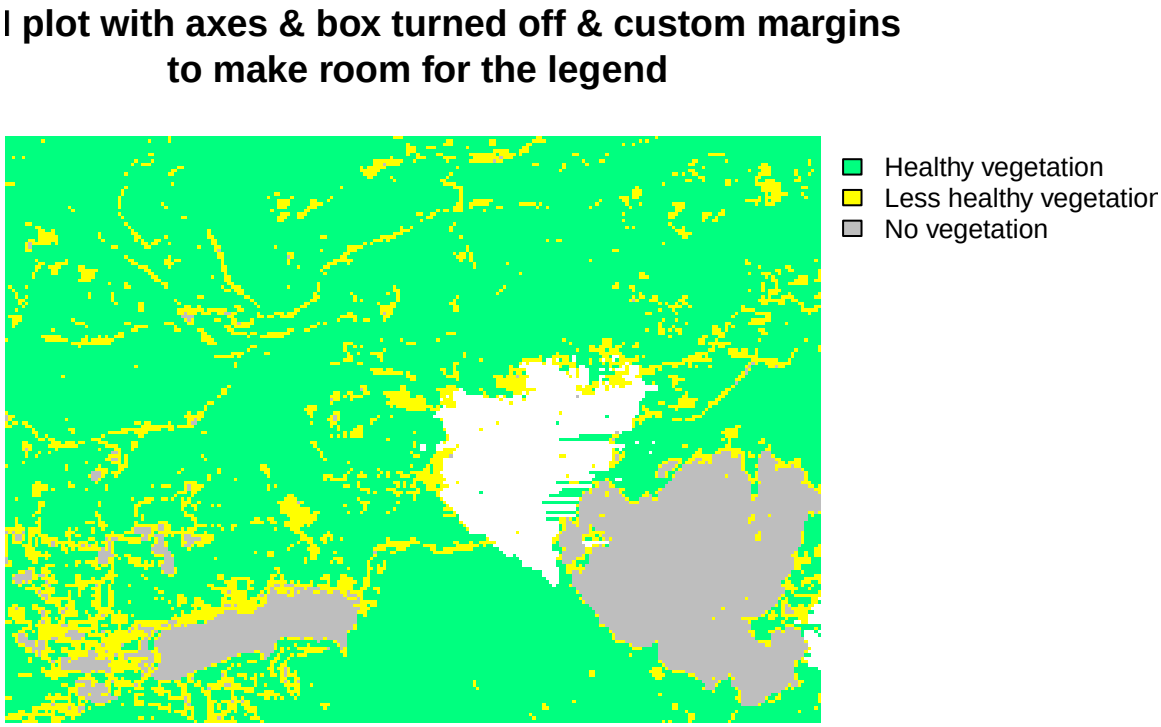

Figure 12: plot with legend in the upper right.

```
# set a margin for our figure
par(xpd=F, mar=c(0,0,2,6))
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="NDVI plot with axes & box turned off & custom margins\n to make room for the legend",
     col=the_colors,
     axes=F,
     box=F)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       bty="n", # turn off legend border
       cex=.8, # make the legend font a bit smaller
       fill= rev(the_colors)) # use rev to reverse the order of colors for the legend
```

While we are at it, let's add a border using a crop extent shapefile that was used to clip these data

```
# import crop extent
crop_ext <- readOGR("data/week6/vector_layers/fire_crop_box_2000m.shp")
# set a margin for our figure
par(xpd=F, mar=c(0,0,2,6))
# plot ndvi with legend
plot(ndvi_classified,
     legend=F,
     main="NDVI plot with axes & box turned off & custom margins\n to make room for the legend",
     col=the_colors,
```

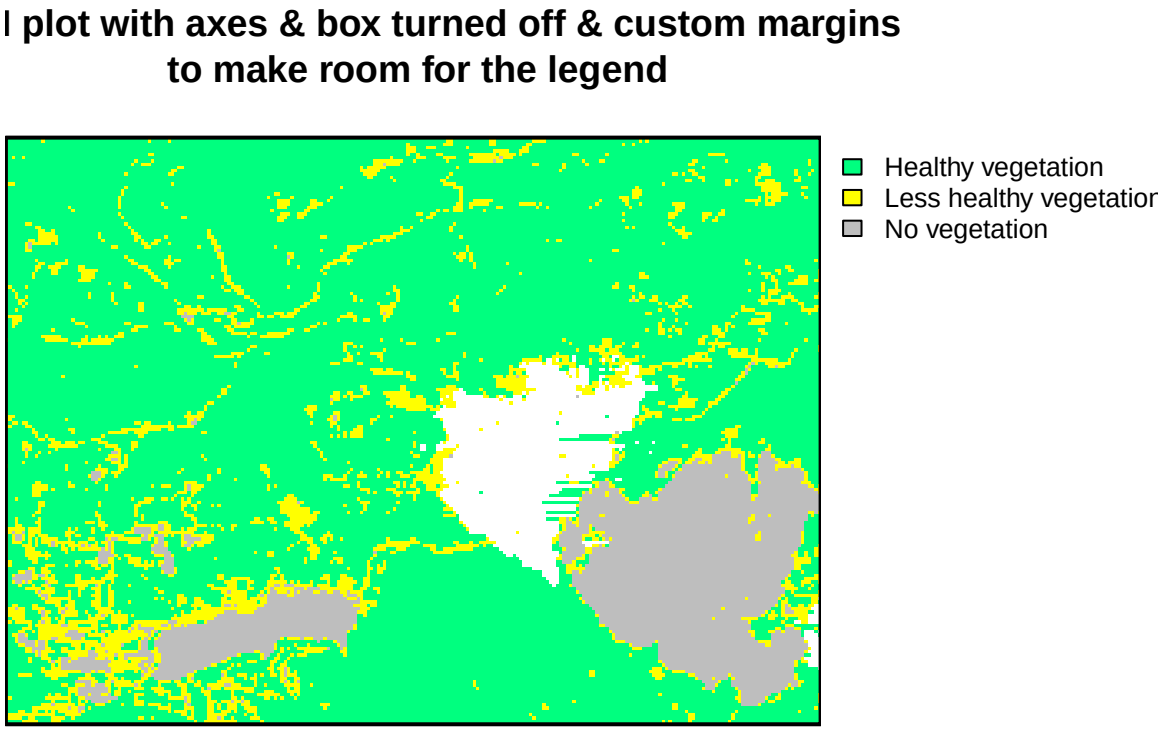

Figure 13: add crop box

```
     axes=F,
     box=F)

plot(crop_ext,
     lwd=2,
     add=T)
# set xpd to T to allow the legend to plot OUTSIDE of the plot area
par(xpd=T)
legend(x = ndvi_classified@extent@xmax, y=ndvi_classified@extent@ymax,
       legend=c("Healthy vegetation", "Less healthy vegetation", "No vegetation"),
       bty="n", # turn off legend border
       cex=.8, # make the legend font a bit smaller
       fill= rev(the_colors)) # use rev to reverse the order of colors for the legend
```

Again - always reset dev when you've been adjusting `par()` values for plots!

```
dev.off()
```

That looks better, doesn't it? Leave comments below if you find other tricks to make your plot look better!