Visualization in R

We will visualize the Landsat 8 Level-2 scene (LC08196025202009190 1RT-SC20200925100850.tif) in R in this section, which you have already downloaded in our download exercise.

When you are ready, open R-Studio.

Prerequisites and import

We need the raster package, so import it into your current working session first. Afterwards, we set the working directory that contains the LS-8 scene and import it as raster-brick. The raster package offers various structures for importing data, depending on how the image data precedes:

- layer() – one file, one band
- stack() – multiple files, multiple bands
- brick() – one file, multiple bands

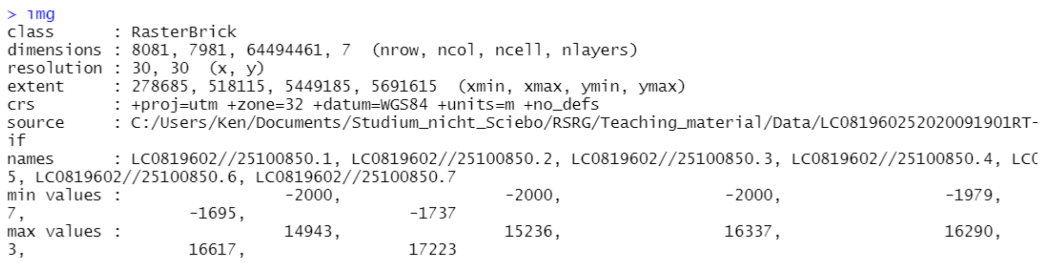
```
library(raster)

#Working directory
setwd("~/Studium_nicht_Sciebo/RSRG/Teaching_material/Data")

img <- brick("LC08196025202009190 1RT-SC20200925100850.tif")

img
```

By calling the object `img` in line 5, the most important metadata can be viewed, e.g., raster dimensions, geometric resolution, geographical extent, coordinate system, file path, as well as minimum and maximum digital numbers per band. This completes the import of the L8 image as a RasterBrick object into R.

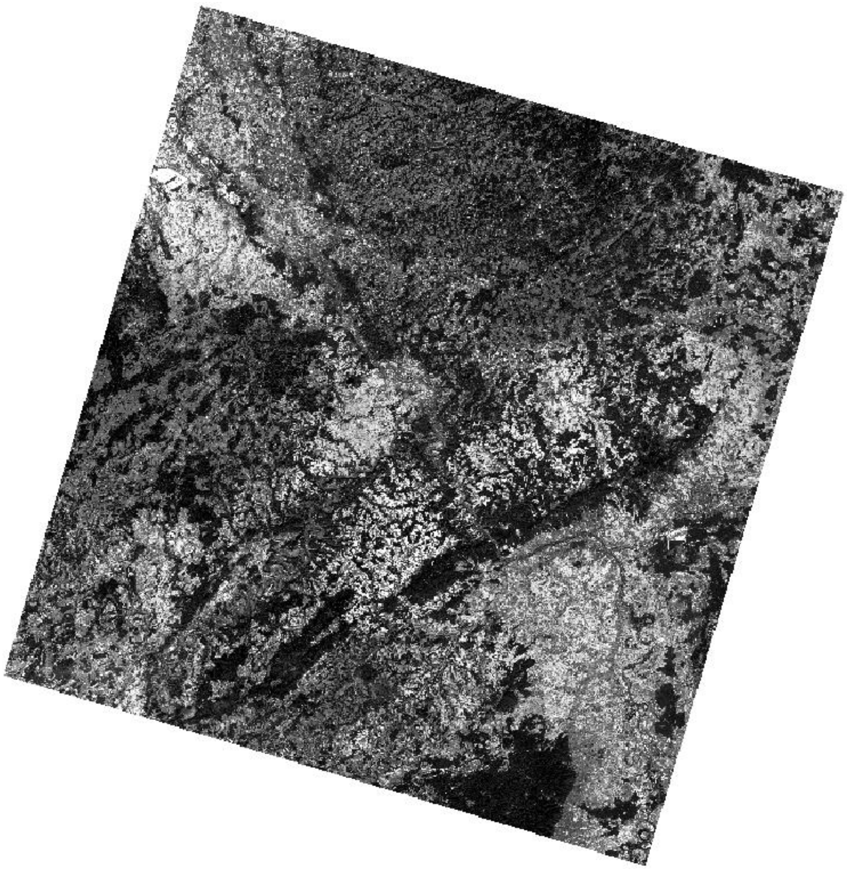


```
> img
class       : RasterBrick
dimensions  : 8081, 7981, 64494461, 7  (nrow, ncol, ncell, nlayers)
resolution  : 30, 30  (x, y)
extent      : 278685, 518115, 5449185, 5691615  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=32 +datum=WGS84 +units=m +no_defs
source      : C:/Users/Ken/Documents/Studium_nicht_Sciebo/RSRG/Teaching_material/Data/LC08196025202009190 1RT-
if
names       : LC0819602//25100850.1, LC0819602//25100850.2, LC0819602//25100850.3, LC0819602//25100850.4, LCC
5, LC0819602//25100850.6, LC0819602//25100850.7
min values  :                 -2000,                 -2000,                 -2000,                 -1979,
7,                -1695,                 -1737
max values  :                 14943,                 15236,                 16337,                 16290,
3,                16617,                 17223
```

Plot

We can now plot the image data. The `plotRGB()` function is a convenient way to visualize different RGB composites. Check out the help for this feature in R-Studio by running `?plotRGB`. The function has three arguments called `r =`, `g =`, and `b =`, which define the three layers to be used for the RGB slots in a RasterBrick or Rasterstack.
If we just want to visualize one band (singleband visualization), we will fill all three slots with the same band, e.g. the green band:

```
plotRGB(img,
        r = 3, g = 3, b = 3,
        stretch = "lin"
        )
```



The stretch argument in line 3 was set to “lin” in order to perform a minimum-maximum contrast stretching to increase the contrast of the image, as described previously.
We can also do a spatial subsetting here by specifying an “Extent”-object using the ext = argument. Objects of class “Extent” are used to define the spatial extent. The four specified arguments are UTM coordinates for (min x, max x, min y, max y). You can set those UTM coordinates arbitrarily and get them, for example, by reading them from QGIS or Google Maps.

By defining different bands for the RGB slots, we can, just as in QGIS, create all imaginable RGB composites:

```
plotRGB(img,
         r = 4, g = 3, b = 2,
         stretch = "lin",
         ext = extent(362699.321, 369890.701, 5619755.014, 5624836.560)
)
```

Make your spatial subset permanent by writing the trimmed image as a new file to your hard drive with `writeRaster`. This is especially useful if you want to test a classification workflow: small data sets will cheer up your work in QGIS, in R, as well as in all other software solutions.

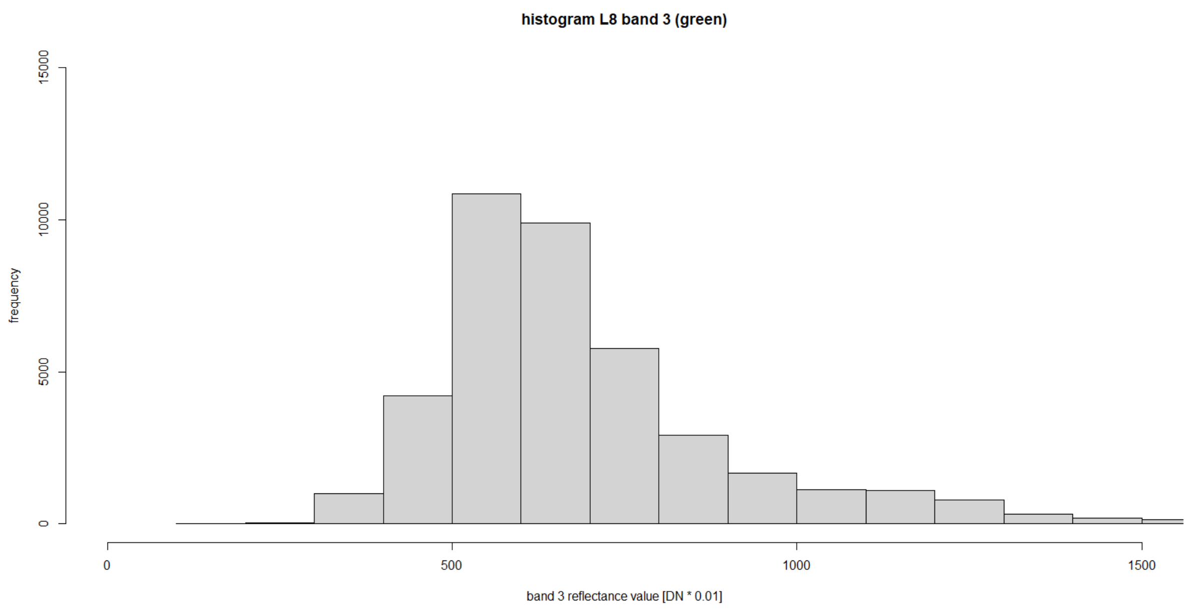
```
img.subset_vis <- crop(img, extent(362699.321, 369890.701, 5619755.014,
5624836.560))

writeRaster(img.subset_vis,
            filename = "C08196025202009190IRT-
SC20200925100850_subset_vis.tif",
            format = "GTiff",
            overwrite = TRUE
)
```

Of course, it is also possible to examine the underlying data distribution in more detail. Therefore, we can look at a histogram of a specific band with the function `hist()`:

```
hist(green,
     breaks = 200,
     xlim = c(0, 1500),
     ylim = c(0, 15000),
     xlab = "band 3 reflectance value [DN * 0.01]",
     ylab = "frequency",
     main = "histogram L8 band 3 (green)"
)
```

The `breaks()`-argument in line 2 is assigned with a single number giving the number of cells for the histogram. So with more breaks, the bars in the histogram become narrower. With `xlim =` and `ylim =` you can narrow the x axis and y axis to a certain range.


histogram L8 band 3 (green)
frequency
0
5000
10000
15000
0
500
1000
1500
band 3 reflectance value [DN * 0.01]