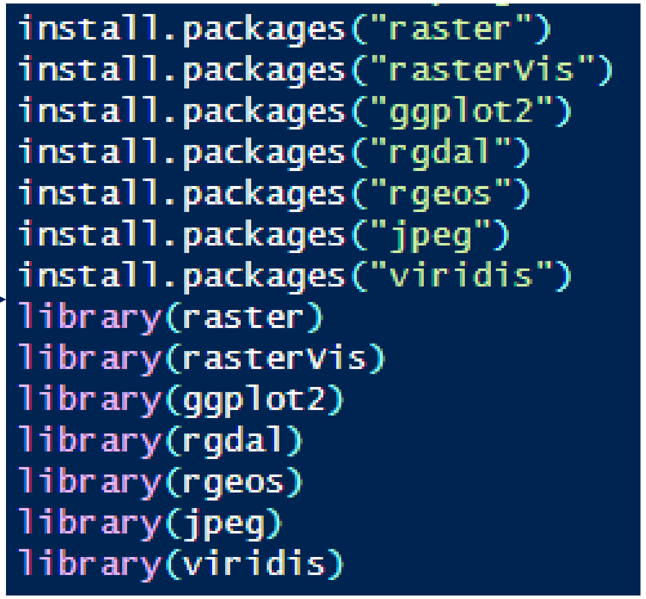
Multi spectral analysis

For a Sentinel-2A dataset

Set up section

The chosen packages were necessary to develop the script and analyze all the given data.



```
install.packages("raster")
install.packages("rasterVis")
install.packages("ggplot2")
install.packages("rgdal")
install.packages("rgeos")
install.packages("jpeg")
install.packages("viridis")
library(raster)
library(rasterVis)
library(ggplot2)
library(rgdal)
library(rgeos)
library(jpeg)
library(viridis)
```

Directory was set as a folder containing the dataset shown in the following slide.

Dataset

S2A_MSIL2A_20230710T110621_N0509_R137_T29SQB_20230710T172204

Dataset Sentinel 2

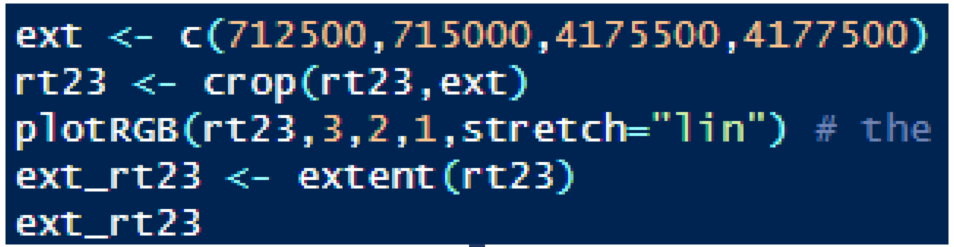
Our available dataset is composed of 4 bands (R,G,B,NIR) corresponding to 490 nm, 560 nm, 665 nm, 842 nm respectively.

So our data is multi-spectral, with only 4 triggerable wavelengths .

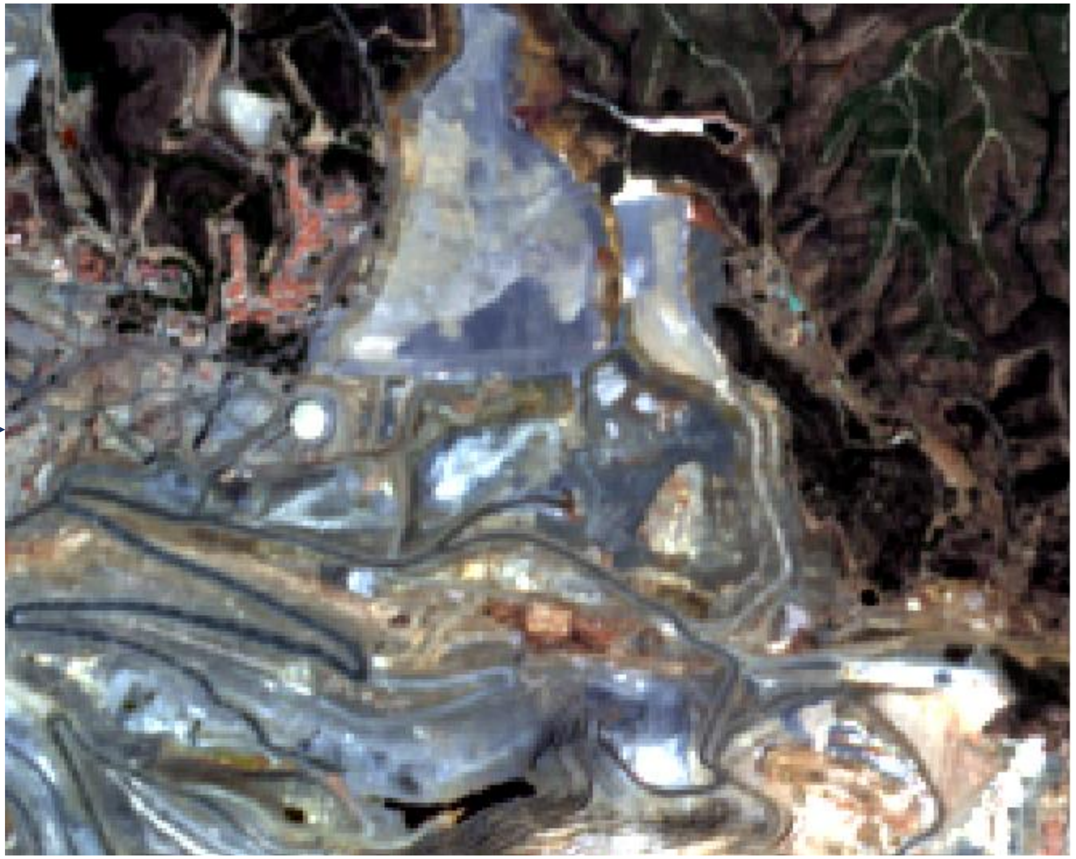
The Level-2A processing includes a Scene Classification and an Atmospheric Correction applied to Top-Of-Atmosphere (TOA) Level-1C orthoimage products. Level-2A main output is an orthoimage atmospherically corrected, Surface Reflectance product.

Dataset

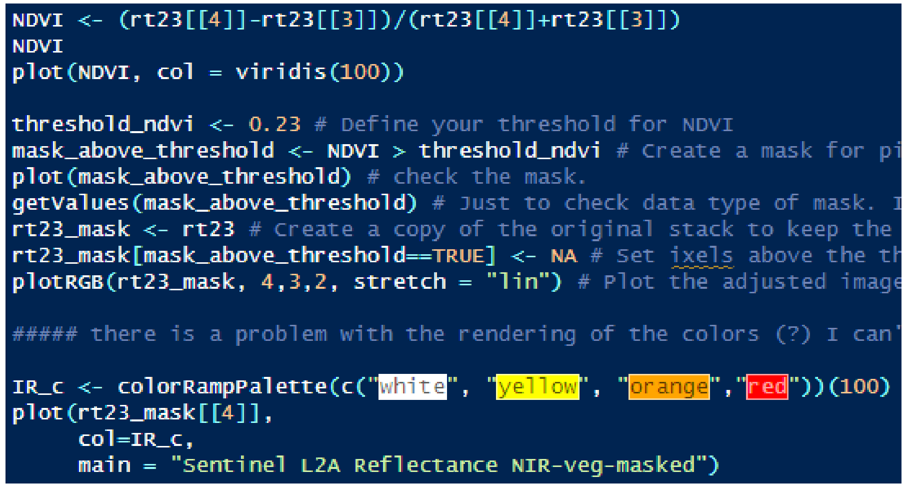
The dataset was cropped to a subset, to make calculations easier for any computer.



```
ext <- c(712500,715000,4175500,4177500)
rt23 <- crop(rt23,ext)
plotRGB(rt23,3,2,1,stretch="lin") # the
ext_rt23 <- extent(rt23)
ext_rt23
```



Data-clean, NDVI



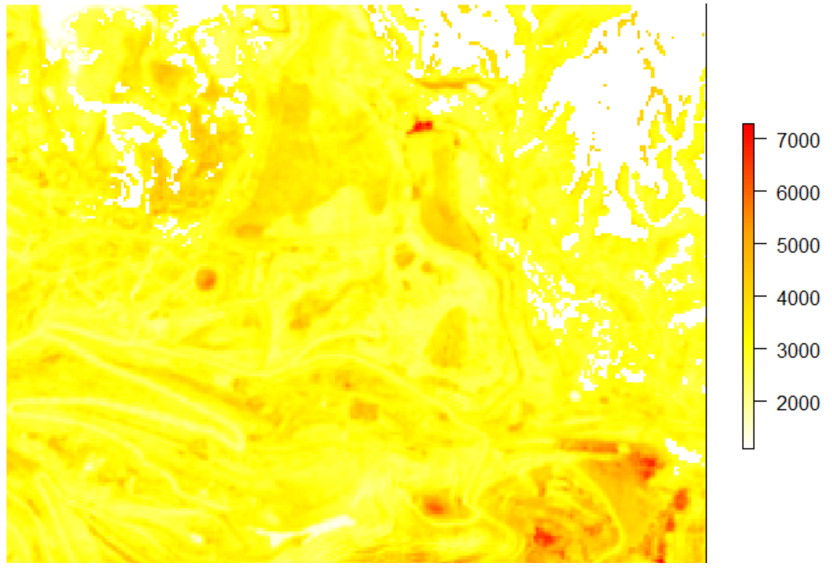
```
NDVI <- (rt23[[4]]-rt23[[3]])/(rt23[[4]]+rt23[[3]])
NDVI
plot(NDVI, col = viridis(100))

threshold_ndvi <- 0.23 # Define your threshold for NDVI
mask_above_threshold <- NDVI > threshold_ndvi # Create a mask for pi
plot(mask_above_threshold) # check the mask.
getValues(mask_above_threshold) # Just to check data type of mask. I
rt23_mask <- rt23 # Create a copy of the original stack to keep the
rt23_mask[mask_above_threshold==TRUE] <- NA # Set ixels above the th
plotRGB(rt23_mask, 4,3,2, stretch = "lin") # Plot the adjusted image

##### there is a problem with the rendering of the colors (?) I can'

IR_c <- colorRampPalette(c("white", "yellow", "orange","red"))(100)
plot(rt23_mask[[4]],
     col=IR_c,
     main = "Sentinel L2A Reflectance NIR-veg-masked")
```

A simple NDVI was calculated:
NDVI = (Band NIR – Band RED) / (Band NIR + Band RED)

A threshold was imposed, so that whatever goes above 0.23 was considered vegetation cover.

These pixels were then located on the original crop and swapped to NA's.

The product was plotted by a new color palette to make things clear, as shown below.



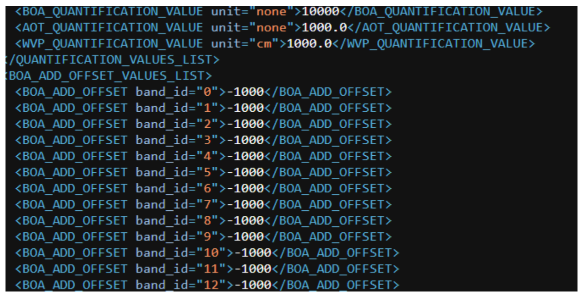
Data clean, reflectance values

Now we need to **convert digital numbers (DN) to the original reflectance values**. Indications can be found on https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a-algorithms-products

Trying to retrive surface reflectance values given the formula:

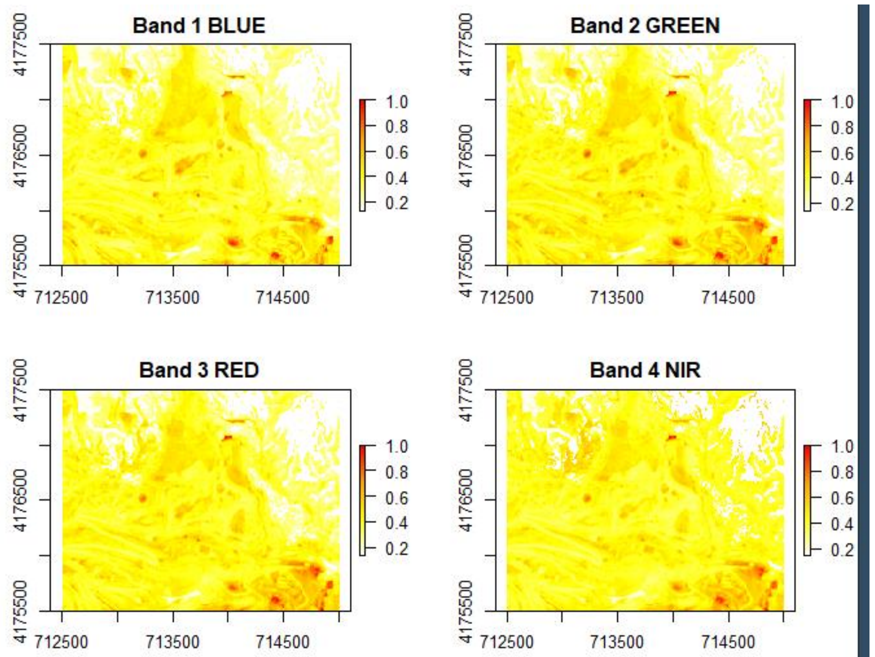
L2A_SR = (L2A_DN + BOA_ADD_OFFSET) / QUANTIFICATION_VALUE

Values are found in the MTD_MSIL2A.xml file.



```
 <BOA_QUANTIFICATION_VALUE unit="none">10000</BOA_QUANTIFICATION_VALUE>
 <AOT_QUANTIFICATION_VALUE unit="none">1000.0</AOT_QUANTIFICATION_VALUE>
 <WVP_QUANTIFICATION_VALUE unit="cm">1000.0</WVP_QUANTIFICATION_VALUE>
/QUANTIFICATION_VALUES_LIST>
BOA_ADD_OFFSET_VALUES_LIST>
 <BOA_ADD_OFFSET band_id="0">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="1">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="2">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="3">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="4">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="5">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="6">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="7">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="8">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="9">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="10">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="11">-1000</BOA_ADD_OFFSET>
 <BOA_ADD_OFFSET band_id="12">-1000</BOA_ADD_OFFSET>
```

Incoherent **negative numbers** were replaced by **NA's**.



```r
par(mfrow = c(2, 2), mar = c(4, 4, 2, 1))
bandcolor <- c("BLUE", "GREEN", "RED", "NIR")
for (i in 1:4) { # OMG this is the first "for"
  plot(rt23_mask_SR[[i]],
       main = paste("Band", i, bandcolor[[i]]),
       col = IR_c)
}

dev.off()
```

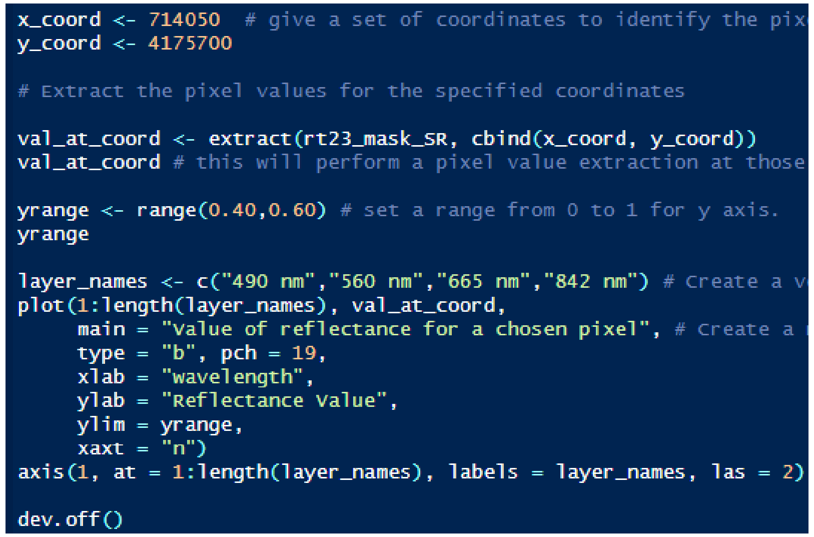
The plot above was successfully drawn.

Single pixel spectrum extraction

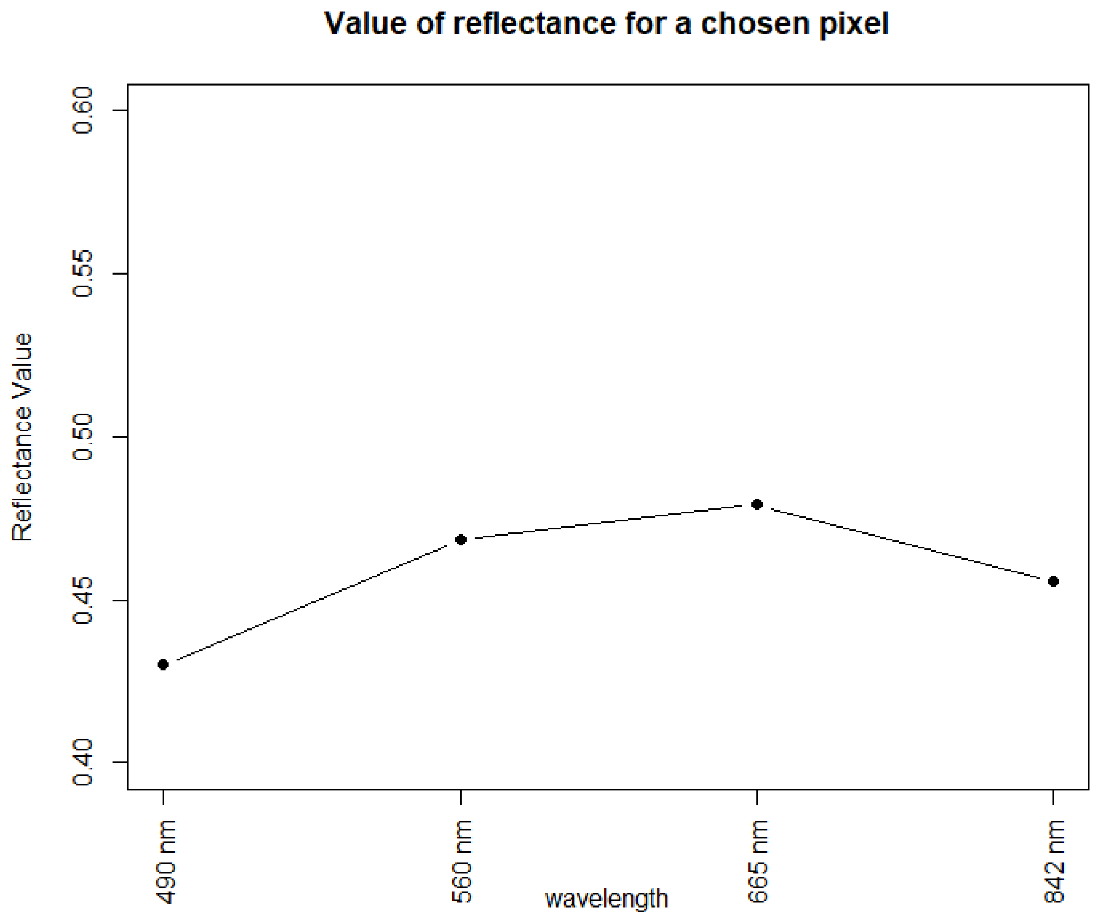
The set of coordinates identifying the pixel were chosen.

The pixel value at those coordinates was extracted.

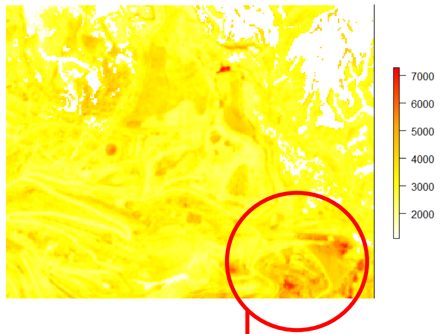
Beware, we're using masked layers (to outcrop vegetation).

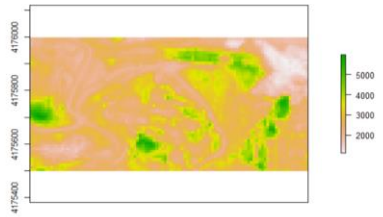


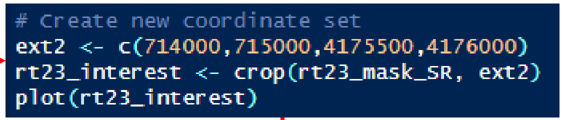
```
x_coord <- 714050  # give a set of coordinates to identify the pix
y_coord <- 4175700

# Extract the pixel values for the specified coordinates

val_at_coord <- extract(rt23_mask_SR, cbind(x_coord, y_coord))
val_at_coord # this will perform a pixel value extraction at those

yrange <- range(0.40,0.60) # set a range from 0 to 1 for y axis.
yrange

layer_names <- c("490 nm","560 nm","665 nm","842 nm") # Create a v
plot(1:length(layer_names), val_at_coord,
     main = "Value of reflectance for a chosen pixel", # Create a 
     type = "b", pch = 19,
     xlab = "wavelength",
     ylab = "Reflectance Value",
     ylim = yrange,
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 2)

dev.off()
```



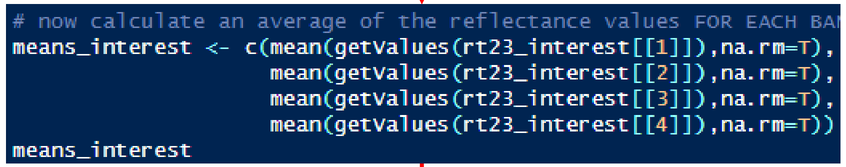
Simple comparison





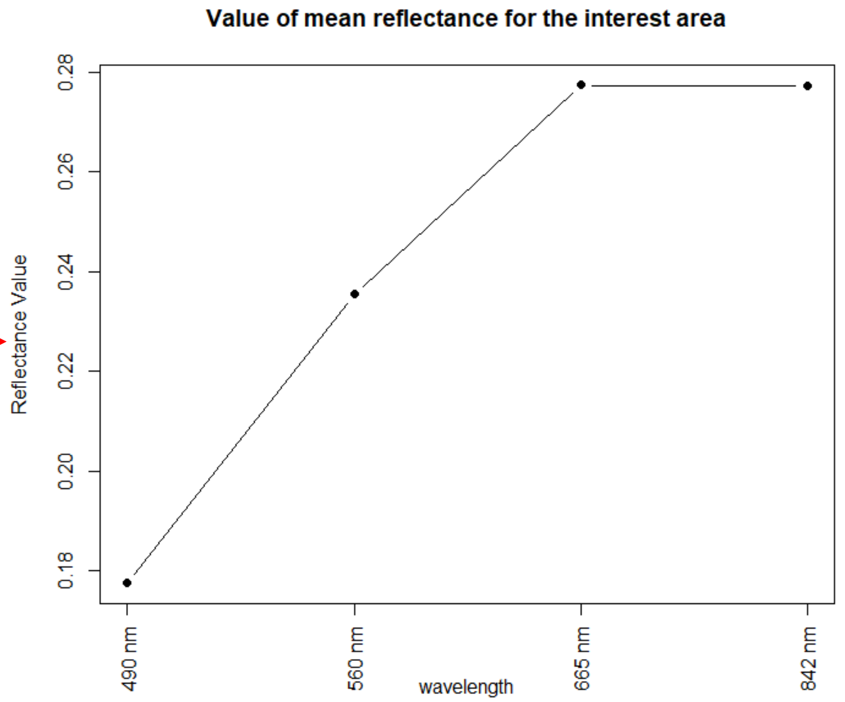


```
# Create new coordinate set
ext2 <- c(714000,715000,4175500,4176000)
rt23_interest <- crop(rt23_mask_SR, ext2)
plot(rt23_interest)
```



```
# now calculate an average of the reflectance values FOR EACH BA
means_interest <- c(mean(getValues(rt23_interest[[1]]),na.rm=T),
                    mean(getValues(rt23_interest[[2]]),na.rm=T),
                    mean(getValues(rt23_interest[[3]]),na.rm=T),
                    mean(getValues(rt23_interest[[4]]),na.rm=T))
means_interest
```

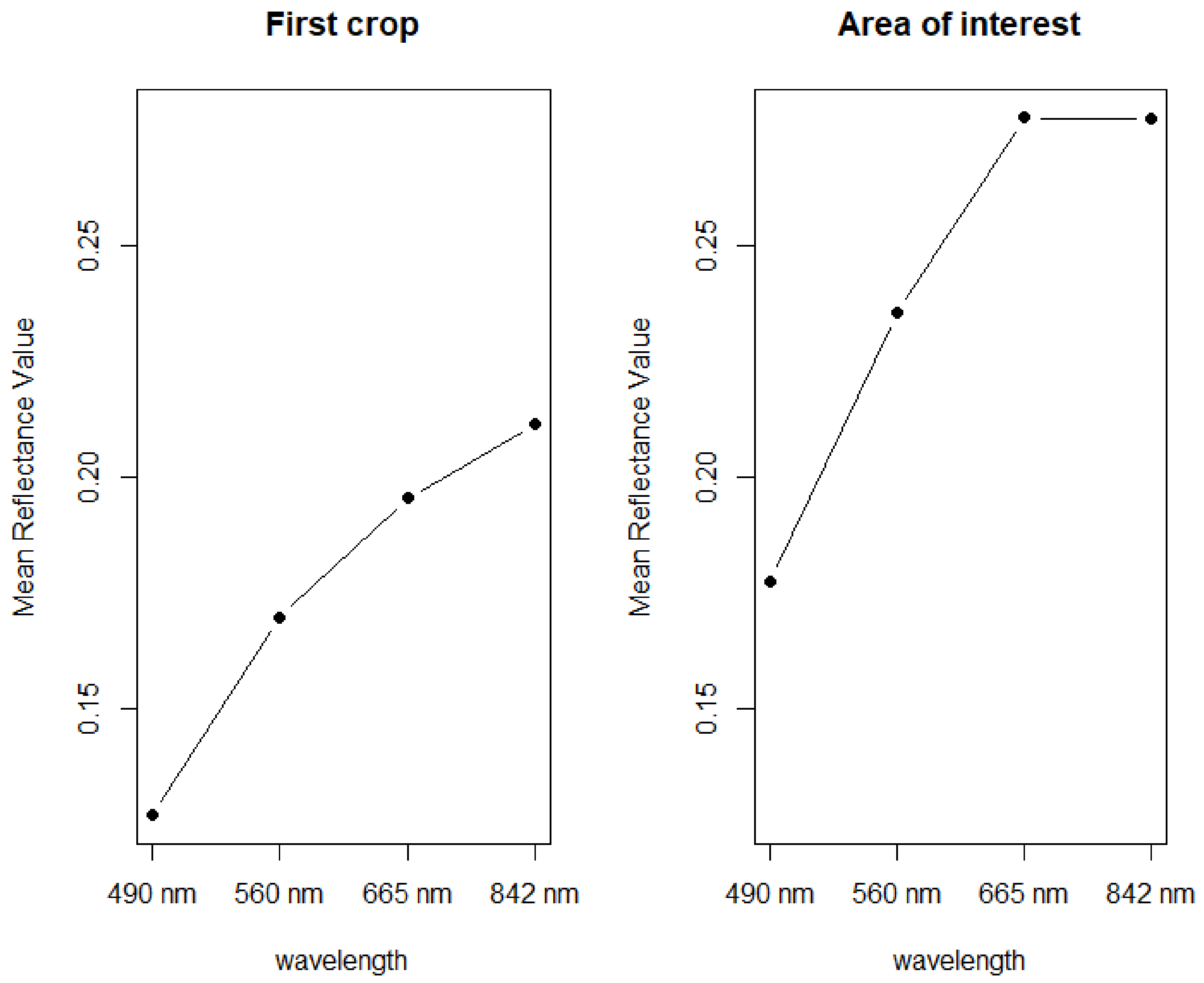
A new extent was calculated on a spotted high reflectance area of the starting crop. The mean of pixel values was calculated and plotted as mean reflectance value of the area.



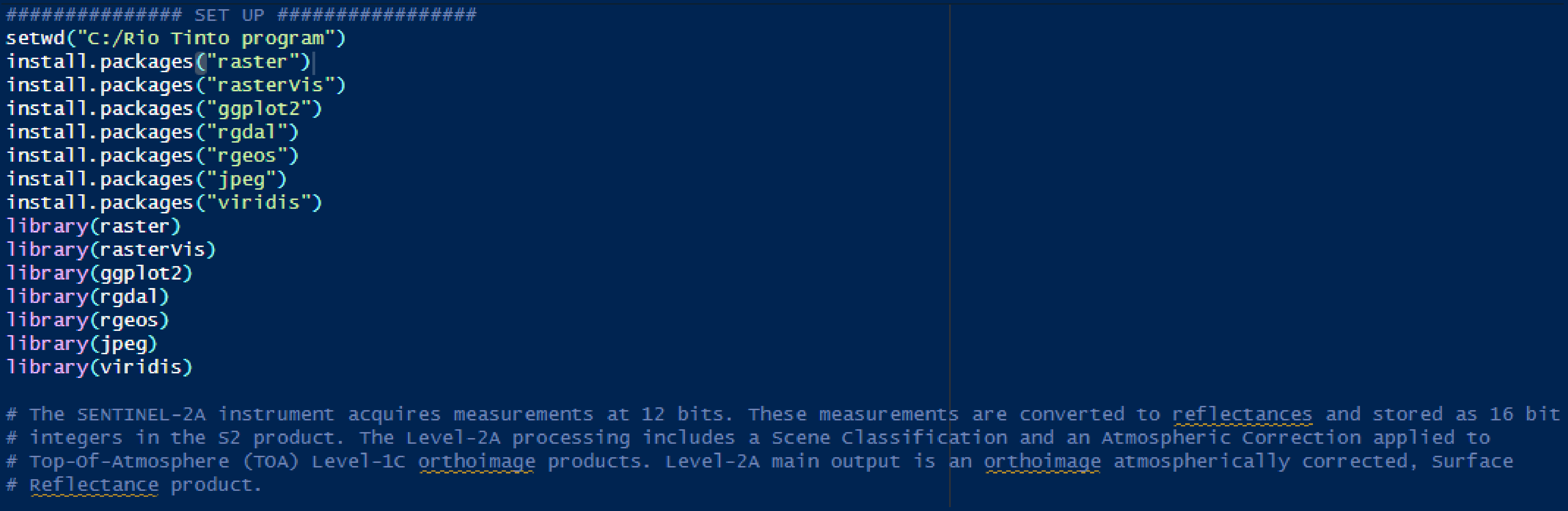
Simple comparison

The same was done, but for the whole initial crop. Then the two graphs were compared next to each other.

The comparison showed a much higher mean reflectance values for the area of interest (south-east in the image) than there is in the area previously cropped.

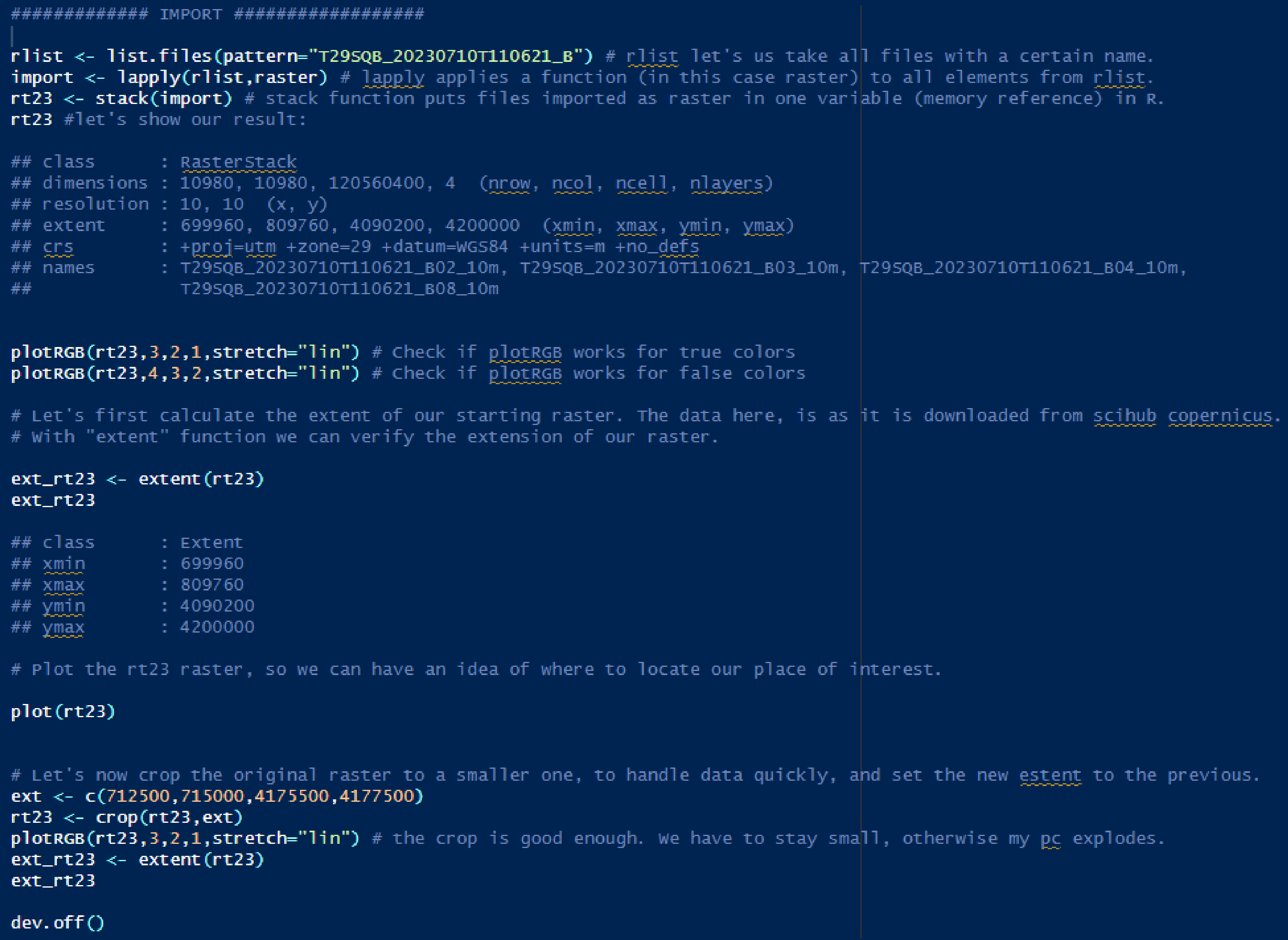


Script (set-up)



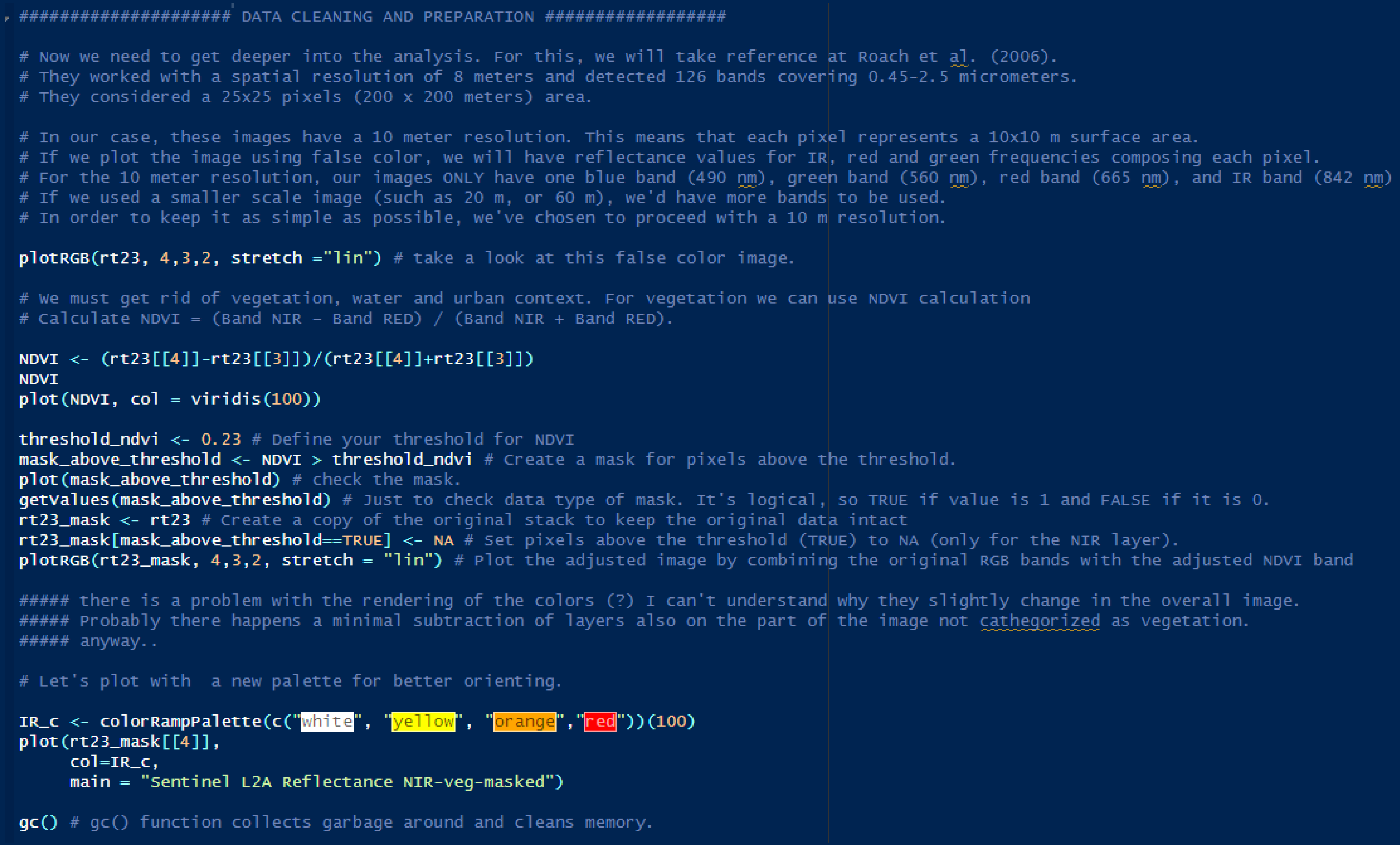
```
############### SET UP #################
setwd("C:/Rio Tinto program")
install.packages("raster")
install.packages("rasterVis")
install.packages("ggplot2")
install.packages("rgdal")
install.packages("rgeos")
install.packages("jpeg")
install.packages("viridis")
library(raster)
library(rasterVis)
library(ggplot2)
library(rgdal)
library(rgeos)
library(jpeg)
library(viridis)

# The SENTINEL-2A instrument acquires measurements at 12 bits. These measurements are converted to reflectances and stored as 16 bit
# integers in the S2 product. The Level-2A processing includes a Scene Classification and an Atmospheric Correction applied to
# Top-Of-Atmosphere (TOA) Level-1C orthoimage products. Level-2A main output is an orthoimage atmospherically corrected, Surface
# Reflectance product.
```

Script (dataset)



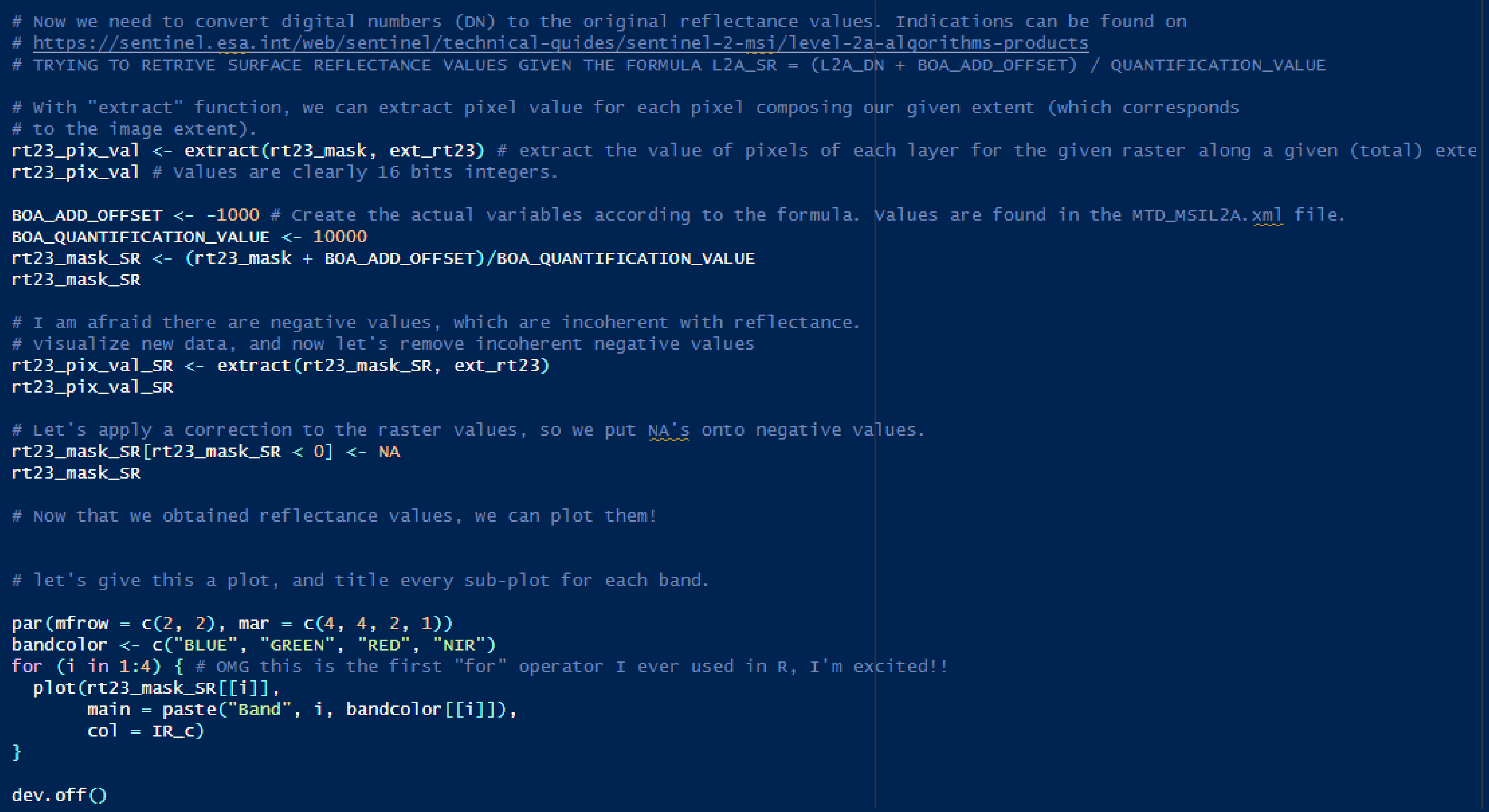
```
############# IMPORT ##################

rlist <- list.files(pattern="T29SQB_20230710T110621_B") # rlist let's us take all files with a certain name.
import <- lapply(rlist,raster) # lapply applies a function (in this case raster) to all elements from rlist.
rt23 <- stack(import) # stack function puts files imported as raster in one variable (memory reference) in R.
rt23 #let's show our result:

## class      : RasterStack
## dimensions : 10980, 10980, 120560400, 4  (nrow, ncol, ncell, nlayers)
## resolution : 10, 10  (x, y)
## extent     : 699960, 809760, 4090200, 4200000  (xmin, xmax, ymin, ymax)
## crs        : +proj=utm +zone=29 +datum=WGS84 +units=m +no_defs
## names      : T29SQB_20230710T110621_B02_10m, T29SQB_20230710T110621_B03_10m, T29SQB_20230710T110621_B04_10m,
##              T29SQB_20230710T110621_B08_10m


plotRGB(rt23,3,2,1,stretch="lin") # Check if plotRGB works for true colors
plotRGB(rt23,4,3,2,stretch="lin") # Check if plotRGB works for false colors

# Let's first calculate the extent of our starting raster. The data here, is as it is downloaded from scihub copernicus.
# With "extent" function we can verify the extension of our raster.

ext_rt23 <- extent(rt23)
ext_rt23

## class      : Extent
## xmin       : 699960
## xmax       : 809760
## ymin       : 4090200
## ymax       : 4200000

# Plot the rt23 raster, so we can have an idea of where to locate our place of interest.

plot(rt23)


# Let's now crop the original raster to a smaller one, to handle data quickly, and set the new estent to the previous.
ext <- c(712500,715000,4175500,4177500)
rt23 <- crop(rt23,ext)
plotRGB(rt23,3,2,1,stretch="lin") # the crop is good enough. We have to stay small, otherwise my pc explodes.
ext_rt23 <- extent(rt23)
ext_rt23

dev.off()
```

Script (cleaning)



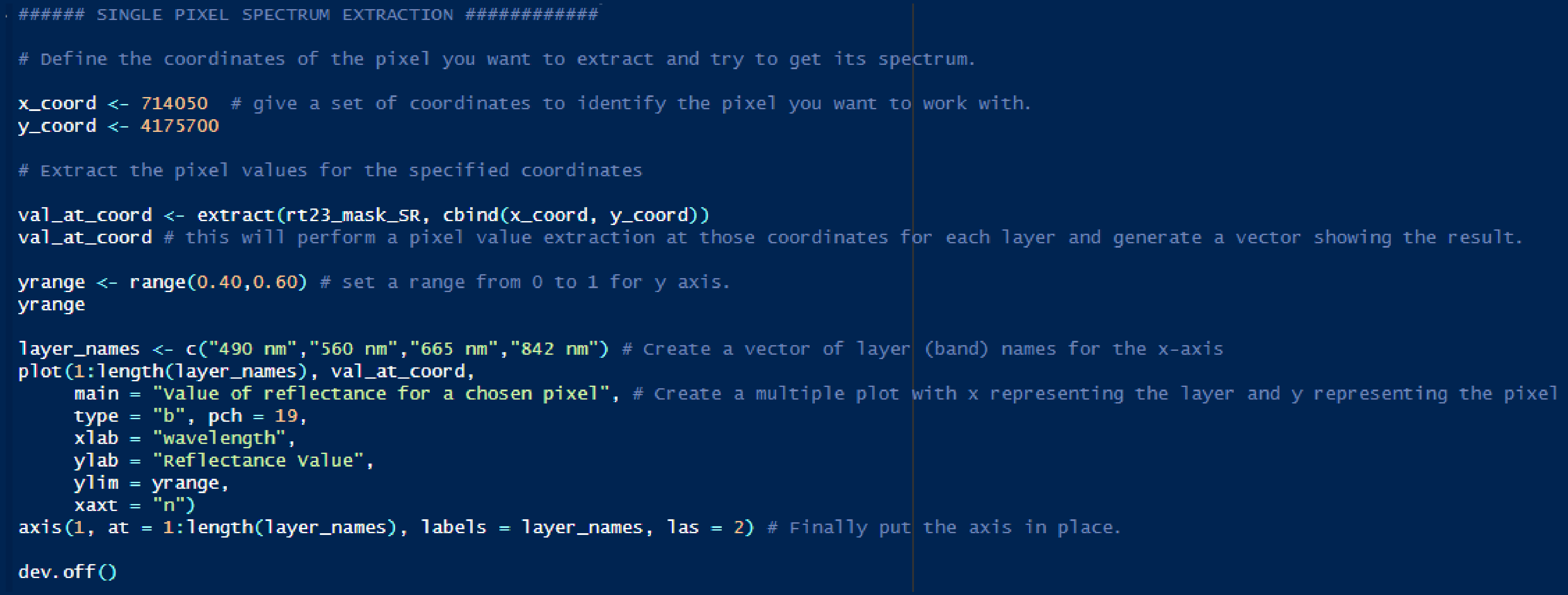
```
#################### DATA CLEANING AND PREPARATION #################

# Now we need to get deeper into the analysis. For this, we will take reference at Roach et al. (2006).
# They worked with a spatial resolution of 8 meters and detected 126 bands covering 0.45-2.5 micrometers.
# They considered a 25x25 pixels (200 x 200 meters) area.

# In our case, these images have a 10 meter resolution. This means that each pixel represents a 10x10 m surface area.
# If we plot the image using false color, we will have reflectance values for IR, red and green frequencies composing each pixel.
# For the 10 meter resolution, our images ONLY have one blue band (490 nm), green band (560 nm), red band (665 nm), and IR band (842 nm)
# If we used a smaller scale image (such as 20 m, or 60 m), we'd have more bands to be used.
# In order to keep it as simple as possible, we've chosen to proceed with a 10 m resolution.

plotRGB(rt23, 4,3,2, stretch ="lin") # take a look at this false color image.

# We must get rid of vegetation, water and urban context. For vegetation we can use NDVI calculation
# Calculate NDVI = (Band NIR - Band RED) / (Band NIR + Band RED).

NDVI <- (rt23[[4]]-rt23[[3]])/(rt23[[4]]+rt23[[3]])
NDVI
plot(NDVI, col = viridis(100))

threshold_ndvi <- 0.23 # Define your threshold for NDVI
mask_above_threshold <- NDVI > threshold_ndvi # Create a mask for pixels above the threshold.
plot(mask_above_threshold) # check the mask.
getValues(mask_above_threshold) # Just to check data type of mask. It's logical, so TRUE if value is 1 and FALSE if it is 0.
rt23_mask <- rt23 # Create a copy of the original stack to keep the original data intact
rt23_mask[mask_above_threshold==TRUE] <- NA # Set pixels above the threshold (TRUE) to NA (only for the NIR layer).
plotRGB(rt23_mask, 4,3,2, stretch = "lin") # Plot the adjusted image by combining the original RGB bands with the adjusted NDVI band

##### there is a problem with the rendering of the colors (?) I can't understand why they slightly change in the overall image.
##### Probably there happens a minimal subtraction of layers also on the part of the image not cathegorized as vegetation.
##### anyway..

# Let's plot with  a new palette for better orienting.

IR_c <- colorRampPalette(c("white", "yellow", "orange","red"))(100)
plot(rt23_mask[[4]],
     col=IR_c,
     main = "Sentinel L2A Reflectance NIR-veg-masked")

gc() # gc() function collects garbage around and cleans memory.
```

Script (cleaning)



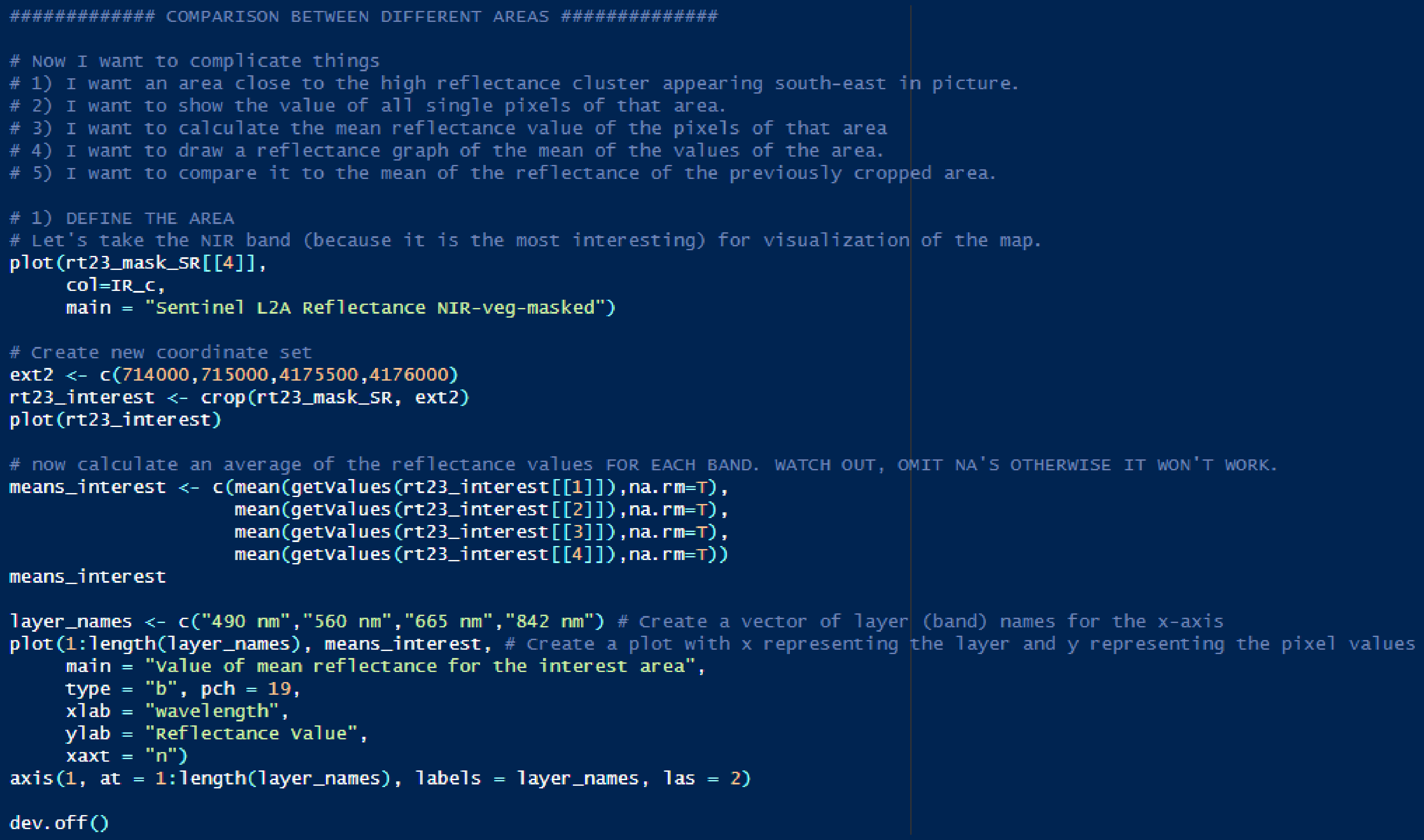
```
# Now we need to convert digital numbers (DN) to the original reflectance values. Indications can be found on
# https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a-algorithms-products
# TRYING TO RETRIVE SURFACE REFLECTANCE VALUES GIVEN THE FORMULA L2A_SR = (L2A_DN + BOA_ADD_OFFSET) / QUANTIFICATION_VALUE

# With "extract" function, we can extract pixel value for each pixel composing our given extent (which corresponds
# to the image extent).
rt23_pix_val <- extract(rt23_mask, ext_rt23) # extract the value of pixels of each layer for the given raster along a given (total) exte
rt23_pix_val # Values are clearly 16 bits integers.

BOA_ADD_OFFSET <- -1000 # Create the actual variables according to the formula. Values are found in the MTD_MSIL2A.xml file.
BOA_QUANTIFICATION_VALUE <- 10000
rt23_mask_SR <- (rt23_mask + BOA_ADD_OFFSET)/BOA_QUANTIFICATION_VALUE
rt23_mask_SR

# I am afraid there are negative values, which are incoherent with reflectance.
# visualize new data, and now let's remove incoherent negative values
rt23_pix_val_SR <- extract(rt23_mask_SR, ext_rt23)
rt23_pix_val_SR

# Let's apply a correction to the raster values, so we put NA's onto negative values.
rt23_mask_SR[rt23_mask_SR < 0] <- NA
rt23_mask_SR

# Now that we obtained reflectance values, we can plot them!


# let's give this a plot, and title every sub-plot for each band.

par(mfrow = c(2, 2), mar = c(4, 4, 2, 1))
bandcolor <- c("BLUE", "GREEN", "RED", "NIR")
for (i in 1:4) { # OMG this is the first "for" operator I ever used in R, I'm excited!!
  plot(rt23_mask_SR[[i]],
       main = paste("Band", i, bandcolor[[i]]),
       col = IR_c)
}

dev.off()
```

Script (single pixel-value extraction)



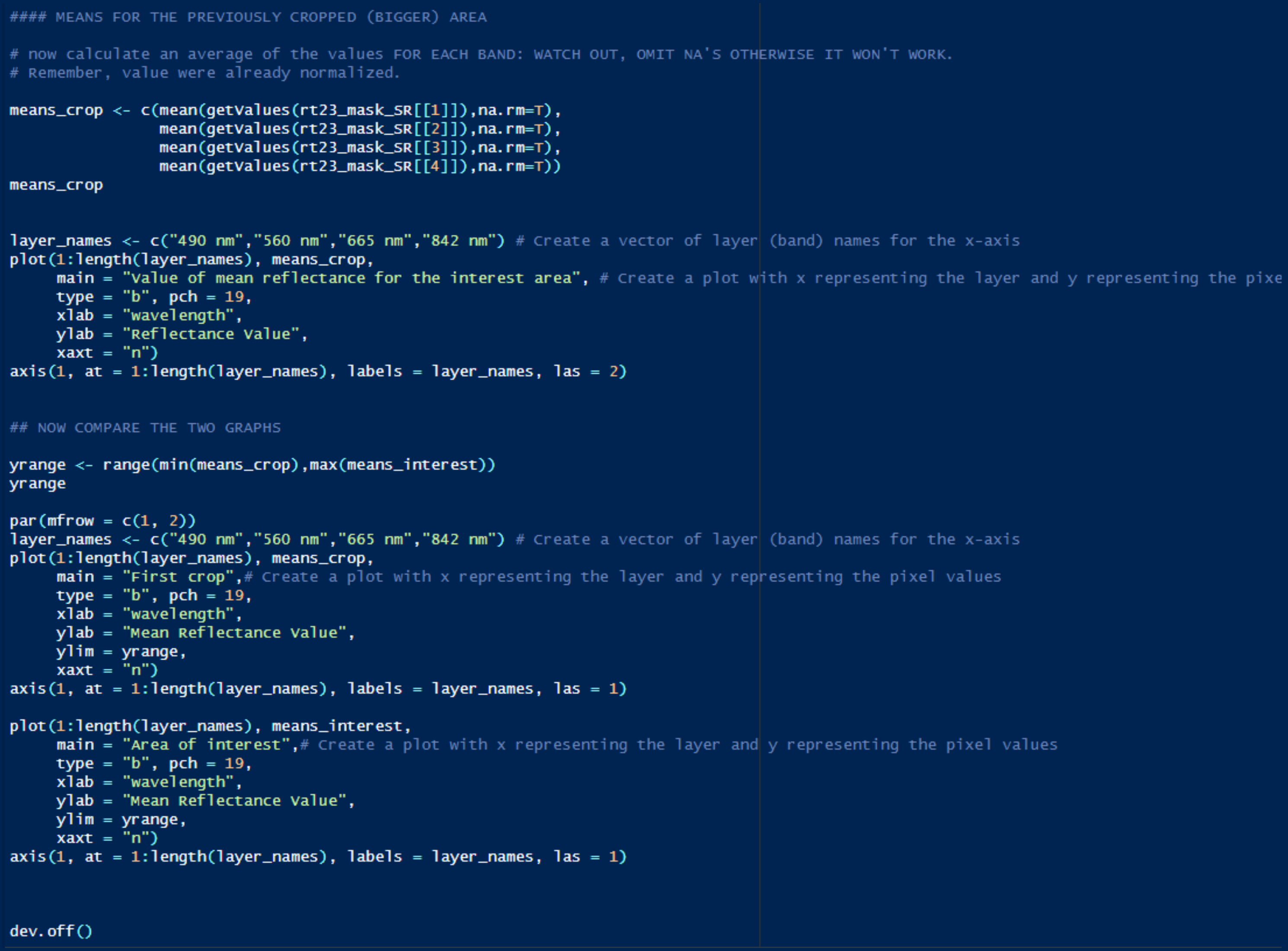
```r
###### SINGLE PIXEL SPECTRUM EXTRACTION ############

# Define the coordinates of the pixel you want to extract and try to get its spectrum.

x_coord <- 714050  # give a set of coordinates to identify the pixel you want to work with.
y_coord <- 4175700

# Extract the pixel values for the specified coordinates

val_at_coord <- extract(rt23_mask_SR, cbind(x_coord, y_coord))
val_at_coord # this will perform a pixel value extraction at those coordinates for each layer and generate a vector showing the result.

yrange <- range(0.40,0.60) # set a range from 0 to 1 for y axis.
yrange

layer_names <- c("490 nm","560 nm","665 nm","842 nm") # Create a vector of layer (band) names for the x-axis
plot(1:length(layer_names), val_at_coord,
     main = "Value of reflectance for a chosen pixel", # Create a multiple plot with x representing the layer and y representing the pixel
     type = "b", pch = 19,
     xlab = "wavelength",
     ylab = "Reflectance Value",
     ylim = yrange,
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 2) # Finally put the axis in place.

dev.off()
```

Script (comparison)



```
############# COMPARISON BETWEEN DIFFERENT AREAS ##############

# Now I want to complicate things
# 1) I want an area close to the high reflectance cluster appearing south-east in picture.
# 2) I want to show the value of all single pixels of that area.
# 3) I want to calculate the mean reflectance value of the pixels of that area
# 4) I want to draw a reflectance graph of the mean of the values of the area.
# 5) I want to compare it to the mean of the reflectance of the previously cropped area.

# 1) DEFINE THE AREA
# Let's take the NIR band (because it is the most interesting) for visualization of the map.
plot(rt23_mask_SR[[4]],
     col=IR_c,
     main = "Sentinel L2A Reflectance NIR-veg-masked")

# Create new coordinate set
ext2 <- c(714000,715000,4175500,4176000)
rt23_interest <- crop(rt23_mask_SR, ext2)
plot(rt23_interest)

# now calculate an average of the reflectance values FOR EACH BAND. WATCH OUT, OMIT NA'S OTHERWISE IT WON'T WORK.
means_interest <- c(mean(getValues(rt23_interest[[1]]),na.rm=T),
                    mean(getValues(rt23_interest[[2]]),na.rm=T),
                    mean(getValues(rt23_interest[[3]]),na.rm=T),
                    mean(getValues(rt23_interest[[4]]),na.rm=T))
means_interest

layer_names <- c("490 nm","560 nm","665 nm","842 nm") # Create a vector of layer (band) names for the x-axis
plot(1:length(layer_names), means_interest, # Create a plot with x representing the layer and y representing the pixel values
     main = "Value of mean reflectance for the interest area",
     type = "b", pch = 19,
     xlab = "wavelength",
     ylab = "Reflectance Value",
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 2)

dev.off()
```

Script (comparison)



```
#### MEANS FOR THE PREVIOUSLY CROPPED (BIGGER) AREA

# now calculate an average of the values FOR EACH BAND: WATCH OUT, OMIT NA'S OTHERWISE IT WON'T WORK.
# Remember, value were already normalized.

means_crop <- c(mean(getValues(rt23_mask_SR[[1]]),na.rm=T),
                mean(getValues(rt23_mask_SR[[2]]),na.rm=T),
                mean(getValues(rt23_mask_SR[[3]]),na.rm=T),
                mean(getValues(rt23_mask_SR[[4]]),na.rm=T))
means_crop


layer_names <- c("490 nm","560 nm","665 nm","842 nm") # Create a vector of layer (band) names for the x-axis
plot(1:length(layer_names), means_crop,
     main = "Value of mean reflectance for the interest area", # Create a plot with x representing the layer and y representing the pixe
     type = "b", pch = 19,
     xlab = "Wavelength",
     ylab = "Reflectance Value",
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 2)


## NOW COMPARE THE TWO GRAPHS

yrange <- range(min(means_crop),max(means_interest))
yrange

par(mfrow = c(1, 2))
layer_names <- c("490 nm","560 nm","665 nm","842 nm") # Create a vector of layer (band) names for the x-axis
plot(1:length(layer_names), means_crop,
     main = "First crop",# Create a plot with x representing the layer and y representing the pixel values
     type = "b", pch = 19,
     xlab = "Wavelength",
     ylab = "Mean Reflectance Value",
     ylim = yrange,
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 1)

plot(1:length(layer_names), means_interest,
     main = "Area of interest",# Create a plot with x representing the layer and y representing the pixel values
     type = "b", pch = 19,
     xlab = "Wavelength",
     ylab = "Mean Reflectance Value",
     ylim = yrange,
     xaxt = "n")
axis(1, at = 1:length(layer_names), labels = layer_names, las = 1)



dev.off()
```