Step 2.0 Pansharpening (option)

Diane ESPEL

2025-06-18

Contents

1 Objectives **1**

2 Script explanation **1**

2.1 **Clean environment and graphics** 1

2.2 **Load required packages** 1

2.3 **Create functions** 2

2.4 **Define global variables** 2

2.5 **Set working directory** 2

2.6 **Load information about available PAN and XS image files** 3

2.7 **Pansharpen each PAN-XS image pair** 3

1 Objectives

This optional script aims to apply pansharpening to satellite images (PAN + XS)* acquired from one sensor across multiple dates, improving spatial resolution and saving the enhanced rasters.

*with PAN: panchromatic and XS : multispectral imagery

2 Script explanation

2.1 Clean environment and graphics

```
rm(list = ls())  # Clear all objects from the R environment to start fresh
graphics.off()  # Close all graphics devices (if any plots are open)
```

2.2 Load required packages

```r
library(sp)  # Classes and methods for spatial data
library(terra)  # For raster data handling
library(stringr)  # For string manipulation (regex, etc.)
library(RStoolbox)  # Includes panSharpen() function for image fusion
```

2.3 Create functions

We define a `normalize()` function to normalize raster values to a [0,1] scale when needed, ensuring compatibility between PAN and XS input images for pansharpening.

```r
# Function to normalize raster values between 0 and 1
normalize <- function(x) {
    (x - global(x, "min", na.rm = TRUE)[1])/(global(x, "max", na.rm = TRUE)[1] -
        global(x, "min", na.rm = TRUE)[1])
}
```

2.4 Define global variables

Note: A global variable is a variable defined outside of any function. This means the variable is accessible from any part of the code, including inside functions. A global variable retains its value throughout the execution of the R script unless it is explicitly modified in the code.

It is important to define at a minimum:

- the `"District"`: the archipelago of interest (e.g. "CRO" for Crozet archipelago)
- the `"Island"`: the island within the archipelago of interest (e.g. "POS" for Possession island)
- the `"Satellite1"`: the name of satellite providing multispectral and panchromatic imagery
- the `"Res1"`: the resolution of panchromatic imagery

```r
District = "CRO"  # 3-letter code for archipelago (e.g. Crozet)
Island = "POS"  # 3-letter code for island (e.g. Possession)
Satellite1 = "Pleiades"  # Name of satellite used for multispectral and panchromatic imagery
Res1 = "50cm"  # Spatial resolution of multispectral imagery
```

2.5 Set working directory

You must define a general root directory (`"localHOME"`) that serves as the base path for your input and output data. This directory should point to the local environment where:

- input MS and panchromatic imagery is located under `"data/raster/Precut_image`
- outputs pansharpened imagery will be saved under `"data/raster/Precut_image`

```r
# Base local path (customize to your local environment)
localHOME = paste0("/home/genouest/cnrs_umr6553/despel/CARTOVEGE/")
# localHOME = paste0('your_local_path/')

# Path to open input MS and panchromatic imagery
open_precut_raster_path = paste0(localHOME, "data/raster/Precut_image")

# Path to save pansharpened imagery
save_precut_raster_path = paste0(localHOME, "data/raster/Precut_image")
```

2.6 Load information about available PAN and XS image files

This code retrieves all PAN and XS image files from a specified directory. It then extracts the `year` and `month` from the PAN file names and stores this metadata in a data frame for further use.

```r
# Get list of all PAN and XS image files
pan_files <- list.files(path = open_precut_raster_path, pattern = "_PAN_precut.tif$",
    recursive = TRUE, full.names = TRUE)
xs_files <- list.files(path = open_precut_raster_path, pattern = "_XS_precut.tif$",
    recursive = TRUE, full.names = TRUE)

# Extract year and month from PAN filenames
pan_info <- data.frame(file = pan_files, year = str_extract(pan_files, "\\d{4}"),
    month = str_extract(pan_files, "_\\d{4}_(\\d{2})_PAN") %>%
        str_remove_all("_\\d{4}_|_PAN"))
```

2.7 Pansharpen each PAN-XS image pair

This section performs a looped pansharpening process for a series of satellite images across multiple years and months:

- For each entry in the `pan_info` table, it retrieves the associated panchromatic (`PAN_file`) and multispectral (`XS_file`) image files.
- If a matching XS image is found, both rasters are loaded and their value ranges are compared.
- If their scales differ, normalization is applied to rescale values between 0 and 1.
- It assigns the RGB channels from the XS image (based on the satellite band order) and applies **Brovey pansharpening** using the `panSharpen()` function.

```r
for (i in 1:nrow(pan_info)) {

    Year <- pan_info$year[i]
    Month <- pan_info$month[i]

    print(paste0("Processing Year ", Year, ", Month ", Month))

    # Get file paths for current PAN and corresponding XS image
    pan_file <- pan_info$file[i]
    xs_file <- xs_files[grepl(paste0(Year, "_", Month, "_XS_precut.tif"), xs_files)]

    if (length(xs_file) == 0) {
        warning(paste("No XS file found for Year", Year, "Month", Month))
        next  # Skip if no matching XS image
    }

    # Load raster images
    PAN <- rast(pan_file)
    XS <- rast(xs_file)

    # Extract min/max values for both rasters
    pan_range <- global(PAN, c("min", "max"), na.rm = TRUE)
    xs_range <- global(XS, c("min", "max"), na.rm = TRUE)

    # Check if PAN and XS have similar value ranges (within 1e-3 tolerance)
    same_range <- all(abs(pan_range - xs_range) < 0.001)

    # Normalize if ranges differ
    if (!same_range) {
        message("PAN and XS are not on the same scale - applying normalization to [0,1]")

        PAN <- normalize(PAN)
        XS <- normalize(XS)
    }

    # Assign RGB channels (Pleiades = BGRN => XS[[1]] = B, [[2]] = G, [[3]] =
    # R)
    B <- XS[[1]]  # blue band
    G <- XS[[2]]  # green band
    R <- XS[[3]]  # red band

    # Perform Brovey pansharpening (RStoolbox)
    Pansharpened_raster <- panSharpen(img = XS, pan = PAN, r = R, g = G, b = B, method = "brovey")
 ↪  # Ensure images are on the same scale (e.g. 0:1, or 0:255)

    # Save output pansharpened image
    NOMraster <- paste0(save_precut_raster_path, "/", District, "_", Island, "_",
        Satellite1, "_", Year, "_", Month, "_", Res1, "_PMS_precut.tif")  # Define output file name
    writeRaster(Pansharpened_raster, NOMraster, overwrite = TRUE)  # Save the pansharpened raster

}
```

This script performs Brovey pansharpening on PAN and XS satellite image pairs for each year and month, that ensures spatial and radiometric consistency across a time series of high-resolution satellite imagery.