Step 3.1 Extracting imagery data within plots

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script aims to perform **spatial extraction of raster data** by computing *weighted mean pixel values** within polygonal sampling units (quadrats) over a multispectral and topographic raster stack.

Utilizing `terra::extract()` with area-based weighting, it accounts for **fractional pixel coverage** along polygon boundaries, ensuring accurate spectral representation of heterogeneous and edge pixels.

Weights are **normalized per polygon** to prevent bias due to varying pixel overlaps or polygon sizes.

The approach mitigates noise and sampling artifacts by **excluding NA values** and applying weighted averaging, thus enhancing the reliability of extracted raster-derived predictors.

Extracted values are merged with original vector attributes, producing a georeferenced dataset optimized for subsequent ecological modeling and classification workflows.

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(terra)  # For raster data manipulation
library(dplyr)  # For data wrangling and manipulation
library(sf)  # For handling vector spatial data (e.g., shapefiles)
```

2.3 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 imagery
- the `"Year1"`: the acquisition year of the multispectral imagery
- the `"Res1"`: the resolution of the raster stack

```r
District = "CRO"  # 3-letter code for the archipelago
Island = "POS"  # 3-letter code for the island
Satellite1 = "Pleiades"  # satellite name of multispectral imagery
Year1 = "2022"  # acquisition year of multispectral imagery
Res1 = "50cm"  # spatial resolution of raster stack
```

2.4 Set working directory

To optimize memory, you must define two general root directory (`"localHOME"`and `"localscratch"`) that serves as the base path for your input and output data, respectively. These directories should point to the local environment where:

- input sample files are located under `"/data/samples"`
- outputs (e.g., plots or shapefiles) will be saved under `"/data/vector/Plots/PrimaryTypo"`

You can create additional subfolders within this structure to organize your outputs by data type, source, or year of analysis.

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

# Path to open input raster stack
open_cut_raster_path = paste0(localscratch, "data/raster/Cut_image")
# Path to open vector plots
open_plots_path = paste0(localHOME, "data/vector/Plots/PrimaryTypo")

# Path to save learning data
save_learning_primary_path = paste0(localscratch, "data/Learning_data/PrimaryTypo")
```

2.5 Open and prepare data

```
# Load the total raster stack
print("Loading raster stack")
raster_total = rast(paste0(open_cut_raster_path, "/", District, "_", Island, "_",
    Satellite1, "_Total_raster_stack_", Year1, "_", Res1, "_cut.TIF"))  #SpatRaster
names(raster_total) = c("R", "G", "B", "NIR", "NDVI", "GRVI", "VARI", "GCCI", "Brightness",
    "BSI", "NDWI", "Dtm", "Slope")

cat("Raster CRS:", crs(raster_total), "\n")  # Print raster Coordinate Reference System

# Load the quadrats (learning plots)
print("Loading learning plots")
FILE1 = paste0(open_plots_path, "/Quadrats_", District, "_", Island,
 ↪  "_ALL_SOURCES_Polygons_EPSG32739.shp")

quadrats_sf <- st_read(shapefile_path)  # Load shapefile as sf object
cat("Quadrats CRS:", st_crs(quadrats_sf)$epsg, "\n")  # Print CRS of shapefile

# Convert sf object to terra-compatible SpatVector
quadrats_vect <- vect(quadrats_sf)
```

##Extract raster data within quadrats

The `extract()` function uses a weighted averaging approach that accounts for partial pixel coverage within polygon boundaries, ensuring accurate representation of mixed pixels at edges.

- `fun = mean`: This specifies the summary function applied to the extracted pixel values per polygon (or line). If fun is not specified, all pixel values whose centers fall inside the polygon are extracted without summarization. Here, the mean of pixel values across all raster layers is calculated for each quadrat, ignoring any missing (NA) values.

- `weights = TRUE`: This option accounts for partial pixel coverage by calculating weight for each pixel proportional to its area covered by the quadrat. Pixels more covered by a quadrat receive higher weights, while less covered pixels receive lower weights, thereby refining the extraction.

- `normalizeWeights = TRUE`: When set to TRUE, the calculated weights are normalized so that their sum per polygon equals 1. This ensures that the weighted average reflects pixel coverage proportionally, avoiding bias due to polygon size or pixel overlap. (This option normalizes the weights so their sum equals 1.)

- `df = TRUE`: The extracted raster statistics are returned as a data frame, with each column corresponding to a raster variable. This data frame can then be merged with the original quadrat vector data (sf object), preserving plot attributes and geometry. (Each column of the data frame corresponds to a variable extracted from the raster.)

```
print("Extracting pixel values within each quadrat")

# Perform weighted extraction of raster values using polygon area coverage
# Returns mean values per polygon, accounting for partial pixel contributions
extracted_data <- terra::extract(
  x=raster_total,             # SpatRaster (raster stack) to extract from
  y=quadrats_vect,            #  SpatVector polygons (quadrats)
  fun = mean,                 # Function to apply to pixels (mean)
  weights = TRUE,             # Use area-based weights (partial pixel contributions)
  normalizeWeights = TRUE,    # Normalize weights to sum to 1 per polygon
  df = TRUE,                 # Return as data.frame
  na.rm = TRUE               # Exclude NA values
)


# Combine the extracted raster values with original quadrat attributes
learning_data <- cbind(quadrats_sf, extracted_data[,-1]) # Remove first column (ID) before merging
learning_data <- st_as_sf(learning_data, coords = c("xcoord_m", "ycoord_m"), crs = 32739) #
↪  Reconvert to sf object with geometry (coordinates must exist!)
```

2.6 Saving learning data

This part of the code handles the saving of datasets used for model training.

First, it saves the complete set of learning plots (`all_learning_plots`), which includes both field-based and photo-interpreted data. Then, a filtered version is created by excluding photo-interpreted plots, keeping only field-verified samples (`true_learning_plots`). Finally, the full R session is saved using `save.image()` to ensure reproducibility of the analysis, preserving the complete environment and all objects created during the session.

```
# Save complete dataset
all_learning_plots = learning_data
print(paste0("Number of all learning plots: ", nrow(all_learning_plots)))
save(all_learning_plots, file = paste0(save_learning_primary_path, "/Learning_plots_",
    District, "_", Island, "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.Rdata"))
write.table(all_learning_plots, file = paste0(save_learning_primary_path, "/Learning_plots_",
    District, "_", Island, "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.csv"),
    sep = ";", dec = ".", row.names = FALSE)

# Save a filtered dataset excluding photo-interpreted plots
true_learning_plots = subset(learning_data$Source != "PHOTO-INTERPRETATION")
print(paste0("Number of field learning plots: ", nrow(true_learning_plots)))
save(true_learning_plots, file = paste0(save_learning_primary_path, "/Learning_plots_",
    District, "_", Island, "_", Satellite1, "_", Year1, "_TRUE_SOURCES_EPSG32739.Rdata"))
write.table(true_learning_plots, file = paste0(save_learning_primary_path, "/Learning_plots_",
    District, "_", Island, "_", Satellite1, "_", Year1, "_TRUE_SOURCES_EPSG32739.csv"),
    sep = ";", dec = ".", row.names = FALSE)

# Save the entire R session image for reproducibility
save.image(paste0("Extraction_raster_data_", District, "_", Island, "_", Satellite1,
    "_", Year1, ".Rdata"))
```