Step 1.3 Merging all learning plots

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script is used to combine spatial vegetation data from different data sources and to resolve spatial overlaps by applying a defined priority rule.

It merges plots from three sources:

— plots from contemporary field survey (`"FIELD_true_plots"`), - plots from historical field survey (i.e. from database) (`"HFi_true_plots`), - plots from photointerpretation (`"photointerpreted_plots"`),

The resulting dataset retains the most reliable data in overlapping areas, ensuring consistency and minimizing redundancy.

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)
```

##Load required packages

```r
library(sf)  # For handling spatial vector data (modern replacement for 'rgdal')
```

2.2 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)

```r
District = "CRO"  # 3-letter code for archipelago (e.g. Crozet)
Island = "POS"  # 3-letter code for island (e.g. Possession)
```

2.3 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 random plots is located under `"data/vector/Plots/PrimaryTypo"`
- outputs of merged plots will be saved under `"data/vector/Plots/PrimaryTypo"`

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

# Path to open input multisources plots
open_plots_path = paste0(localHOME, "data/vector/Plots/PrimaryTypo")

# Path to save all learning plots
save_plots_path = paste0(localHOME, "data/vector/Plots/PrimaryTypo")
```

2.4 Open all data sources

This section loads the available shapefiles corresponding to the `FIELD_true`, `HFI_true`, and `photointerpreted` -plots. The `FIELD_true` dataset is assumed mandatory, while the other two are included conditionally, depending on whether the corresponding files exist in the input directory. The use of conditional loading allows the script to adapt to varying data availability.

```r
# Define the file paths for the three possible shapefile sources: 1. FIELD
# samples (mandatory) 2. HFI samples (optional) 3. Photointerpreted plots
# (optional)
FILE1 <- paste0(open_plots_path, "/Quadrats_", District, "_", Island,
 ↪  "_ALL_FIELD_SAMPLES_Polygons_corrected_EPSG32739.shp")
FILE2 <- paste0(open_plots_path, "/Quadrats_", District, "_", Island,
 ↪  "_ALL_HFI_SAMPLES_Polygons_corrected_EPSG32739.shp")
FILE3 <- paste0(open_plots_path, "/Quadrats_", District, "_", Island,
 ↪  "_PHOTO-INTERPRETED_Polygons_EPSG32739.shp")
```

```r
# Load the FIELD shapefile (mandatory)
FIELD_true_plots <- st_read(FILE1)

# If HFI file exists, load it
if (file.exists(FILE2)) {
    assign("HFI_true_plots", st_read(FILE2))
}

# If photointerpreted file exists, load it
if (file.exists(FILE3)) {
    assign("photointerpreted_plots", st_read(FILE3))
}
```

2.5 Merge pots and filter overlapping plots

The objective of this section is to intersect the vector layers to check whether any plots overlap. It implements a spatial filtering process by comparing polygons-based plots for spatial intersection and keeping only the one with the highest priority in each overlapping pair.

To achieve this :

- All avaiable plots are merged (`all_plots`)
- A priority index (`priority_order`) is assigned to each plot based on its source: FIELD = 1 (highest priority), PHOTO-INTERPRETATION = 2, HFI = 3 (lowest priority)
- Plots are intersected and spatial overlaps between plots are checked

When two plots overlap, the one with the lower priority index is removed.

The result is a filtered dataset of non-overlapping plots (`filtered_plots`), keeping only the most reliable (highest priority) plot in each overlapping group.

```r
# Start with FIELD plots (always present)
all_plots <- FIELD_true_plots

# If photointerpreted plots exist, append them to the dataset
if (exists("photointerpreted_plots")) {
    all_plots <- rbind(all_plots, photointerpreted_plots)
}

# If HFI plots exist append them to the dataset
if (exists("HFI_true_plots")) {
    all_plots <- rbind(all_plots, HFI_true_plots)
}

# At this stage, all_plots contains FIELD plots and other sources if they exist

# Define a priority order for conflict resolution: lower number = higher
# priority
priority_order <- c(FIELD = 1, `PHOTO-INTERPRETATION` = 2, HFI = 3)  # assuming HFI correspond to
 ↪  historical values
all_plots$priority <- priority_order[all_plots$Source]  # Create a priority column based on the
 ↪  Source column

# Initialize a logical vector to keep track of plots to retain
keep <- rep(TRUE, nrow(all_plots))  # Initially assume all plots are first marked as TRUE (to keep)
```

```r
# Loop to detect and resolve spatial overlaps by priority
for (i in seq_len(nrow(all_plots) - 1)) {
    print(paste0("Processing quadrat", i))  # Debug message

    if (!keep[i])
        next  # Skip if quadrat i is already marked for removal, we skip it
    for (j in (i + 1):nrow(all_plots)) {
        if (!keep[j])
            next  # Skip if quadrat j is already marked for removal, we skip it

        # Check if quadrat i and quadrat j intersect
        if (st_intersects(all_plots[i, ], all_plots[j, ], sparse = FALSE)[1, 1]) {

            # Compare priorities: lower number means higher priority
            if (all_plots$priority[i] <= all_plots$priority[j]) {
                # If quadrat i is of equal or higher priority than j, mark j
                # for removal
                keep[j] <- FALSE
            } else {
                # Otherwise, mark i for removal and exit the inner loop
                keep[i] <- FALSE
                break
            }
        }
    }
}

# Filter dataset to retain only non-overlapping, highest-priority plots
filtered_plots <- all_plots[keep, ]

# Print final retained plots to the console
print(filtered_plots)
```

2.6 Save filtered plots

The filtered plot layer is finally save as shapefile and its centroids, as well.

```r
# Save the filtered plots to a shapefile
st_write(filtered_plots, paste0(save_plots_path, "/Quadrats_", District, "_", Island,
    "_ALL_SOURCES_Polygons_EPSG32739.shp"), driver = "ESRI Shapefile", append = FALSE)

# Extract centroids from filtered plots and save as a new shapefile and as a
# csv file for tabular use
filtered_plots_sf <- st_as_sf(filtered_plots)  # Ensure it's an sf object
centroids <- st_centroid(filtered_plots_sf)  # Compute centroids
st_write(centroids, paste0(save_plots_path, "/Quadrats_", District, "_", Island,
    "_ALL_SOURCES_Centroids_EPSG32739.shp"), driver = "ESRI Shapefile", append = FALSE)
write.table(centroids, file = paste0(save_plots_path, "/Quadrats_", District, "_",
    Island, "_ALL_SOURCES_Centroids_EPSG32739.csv"), sep = ";", dec = ".", row.names = FALSE)
```

This script merges plot data from multiple sources (recent and historical field survey, photo-interpretation), resolves spatial overlaps based on source priority, and exports the final cleaned polygons and their centroids for further analysis.

‘‘‘