Step 6.2 Identifying misclassifications at the site scale

6.2.1 Correcting observed map with new typology

Diane ESPEL

2025-06-20

Contents

1 Objectives

This script aims to update an observed habitat polygon map by integrating a corrected habitat typology derived from field learning data. The goal is to ensure that all habitat classifications used in the observed map are consistent with the latest validated typology levels.

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(dplyr)  # Data manipulation (filter, select, mutate, etc.)
library(sf)  # Handling of spatial vector data using Simple Features
library(terra)  # Manipulation of raster and vector spatial data
```

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 satellite name for multispectral imagery
- the `"Year1"` : the acquisition year of the multispectral imagery used in photointerpretation
- the `"maxTypoLevel"`: the maximum typology level

```r
District = "CRO"  # 3-letter code for archipelago (e.g. Crozet)
Island = "POS"  # 3-letter code for island (e.g. Possession)
Satellite1 = "Pleiades"  # satellite name of multispectral imagery
Year1 = "2022"  # acquisition year of multispectral imagery used for photo interpretation
maxTypoLevel = 4  # maximum typology level
```

2.4 Set working directory

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

- input learning data with new typology are located under `"data/Learning_data/NewTypo"`
- input observed map are located under `"data/vector/Observed_map/PrimaryTypo"`
- outputs results will be saved under `"data/vector/Observed_map/NewTypo"`

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

# Path to open learning data with new typology
open_learning_new_path = paste0(localscratch, "data/Learning_data/NewTypo")

# Path to open observed map
open_ObsMap_path = paste0(localHOME, "data/vector/Observed_map/PrimaryTypo")

# Path to save results
save_NewObsMap_vector_path = paste0(localscratch, "data/vector/Observed_map/NewTypo")
```

::: {align="justify"} ## **Load and prepare observed habitat map**

The observed polygon map is then loaded and cleaned to ensure consistent formatting of habitat codes.

```r
# Load observed habitat shapefile
print("Loading the observed habitat polygon map...")
FILE1 = paste0(open_ObsMap_path, "/", "Corrected_observed_map_", District, "_", Island,
    "_", Satellite1, "_", Year1, "_EPSG32739.shp")
polygons_map <- st_read(FILE1)

# Clean: remove dots from all columns starting with 'Hab_L'
print("Removing any dots from all habitat code columns (columns starting with 'Hab_L')...")
hab_cols <- grep("^Hab_L", colnames(polygons_map), value = TRUE)
polygons_data <- polygons_map %>%
    mutate(across(all_of(hab_cols), ~gsub("\\.", "", .)))
```

2.5 Open the new typology

The updated typology is loaded from learning data containing corrected class labels for each habitat level.

```r
print("Loading the updated habitat typology from learning data...")
FILE2 <- paste0(open_learning_new_path, "/Learning_plots_", District, "_", Island,
    "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.csv")
learning_data = read.csv(FILE2, sep = ";", dec = ".", stringsAsFactors = FALSE)  #
 ↪  `stringsAsFactors=F` ensures character strings don't import as factors
```

##Reclassification: apply corrections to observed map

This section involves iterating through each typology level, comparing existing class labels (included in `original_col`) with the validated ones (included in `corrected_col`), and applying the most frequent corrected label (`most_common_corr`) to each matching polygon.

```r
# Copy the spatial layer to avoid modifying the original
polygons_map_corr <- polygons_data


# Initialize logical matrix to track rows with valid class matches at each
# level
match_matrix <- matrix(TRUE, nrow = nrow(polygons_map_corr), ncol = maxTypoLevel)

# Loop over habitat levels
for (l in seq(1:maxTypoLevel)) {

    original_col <- paste0("Hab_L", l)
    corrected_col <- paste0("Hab_L", l, "_corr")

    # Get valid class values from learning data
    valid_classes <- unique(learning_data[[original_col]])

    # Identify matching rows in polygons_map
    match_vector <- polygons_map_corr[[original_col]] %in% valid_classes
    match_matrix[, l] <- match_vector

    # Create a vector to hold corrected values
    corrected_values <- rep(NA, nrow(polygons_map_corr))

    # For each valid class, assign the most frequent corrected class
    for (class in valid_classes) {
        matched_rows <- which(learning_data[[original_col]] == class)
        most_common_corr <- names(sort(table(learning_data[[corrected_col]][matched_rows]),
```

```
            decreasing = TRUE))[1]

        # Apply correction to matching polygons
        corrected_values[polygons_map_corr[[original_col]] == class] <- most_common_corr
    }

    # Add corrected column to polygons_map_corr
    polygons_map_corr[[corrected_col]] <- corrected_values

}  #  End of typology level loop
```

2.6 Keep only polygons with valid classes at all levels

A logical matrix is used to track which polygons have valid classifications across all levels. Polygons without complete valid matches are excluded from the final map.

```
rows_to_keep <- apply(match_matrix, 1, all)
message(paste0("Keeping ", sum(rows_to_keep), " / ", nrow(polygons_map_corr), " polygons with valid
 ↪  classes at all levels."))
polygons_map_corr <- polygons_map_corr[rows_to_keep, ]
```

2.7 Final cleanup and save

In the final step, the original habitat columns are removed and replaced with their corrected versions. The cleaned and reclassified map is then exported as a new shapefile for future use.

```
# Remove original habitat class columns
original_cols <- paste0("Hab_L", 1:maxTypoLevel)
polygons_map_corr <- polygons_map_corr %>%
    select(-all_of(original_cols))

# Rename corrected columns to original names
corrected_cols <- paste0("Hab_L", 1:maxTypoLevel, "_corr")
names(polygons_map_corr)[match(corrected_cols, names(polygons_map_corr))] <- original_cols

# Save the updated polygon shapefile
FILE3 <- paste0(save_NewObsMap_vector_path, "/", "Corrected_observed_map_NewTypo_",
    District, "_", Island, "_", Satellite1, "_", Year1, "_EPSG32739.shp")
st_write(polygons_map_corr, FILE3, driver = "ESRI Shapefile", append = FALSE)
```