# Step 6.1 Identifying error sources at plot scale

6.1.1 Identifying misclassified plots

Diane ESPEL

2025-06-20

## Contents

## 1 Objectives

This script aims to evaluate the performance of Random Forest habitat classification models by identifying and analyzing classification errors across different hierarchical classification levels and modeling strategies (FLAT vs. HIERARCHICAL). Specifically, it flags misclassified samples, spatializes them, summarizes class-specific error rates, and visualizes the distribution of errors to help identify confusion-prone habitat types and potential sources of misclassification. # **Script explanation**

### 1.1 Clean environment and graphics

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

### 1.2 Load required packages

```r
library(ade4)  # Multivariate data analysis and ecological data exploration
library(dplyr)  # Data manipulation verbs like filter, select, mutate, summarize
library(sf)  # Manage spatial vector data using simple features (points, lines, polygons)
library(ggplot2)  # Create elegant and customizable graphics and plots
```

## 1.3 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 `"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
maxTypoLevel = 4  # Define maximum typology level
```

## 1.4 Set working directory

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

- input predicted samples are located under `"results/Model/Final_model`
- outputs results will be saved under `"results/Model/Final_model"`

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

# Path to open Final Model
open_final_model_path = paste0(localscratch, "results/Model/Final_model")

# Path to save results
save_final_model_path = paste0(localscratch, "results/Model/Final_model")
```

## 1.5 Identify local errors

At the beginning, the script iterates over two modeling strategies defined in `type_model_list`: `FLAT`, where habitat classes are modeled independently, and `HIERARCHICAL`, where nested class relationships are considered. Within each modeling strategy, a second loop processes each classification level (from level 1 to the maximum defined by `maxTypoLevel`).

To achieve this, for each level,:

- The script loads a previously saved dataframe containing predicted vs. observed habitat labels (`ObsPred_df`).
- It then compares predictions with observations to flag misclassified samples by assigning an `"Error"` label (`"YES"` for mismatches, `"NO"` otherwise).
- The script then spatializes these misclassified records by converting them into a spatial object using their X/Y coordinates and saves them as shapefiles. This enables the visual inspection of spatial patterns of classification errors in a GIS environment, which can be useful to detect systematic spatial biases or confusion zones.

- Next, the script computes the total number of misclassifications and total sample counts for each habitat class. These statistics are stored in a summary table (`Class_error_count`) and exported as a CSV file.
- To support interpretation, a **stacked barplot** is generated for each classification level and modeling strategy. These plots illustrate the absolute number of errors in relation to the total number of samples per habitat class.

```r
# Analyse data errors -------------------------------------------------------------

# Define the list of model types to test
type_model_list = c("FLAT", "HIERARCHICAL")


# Loop through each model type
for (type_model in type_model_list) {

  print(paste0("Modeling strategy: ", type_model))

  # Loop through classification levels from 1 to maxTypoLevel
  for (l in seq(1, maxTypoLevel)) {

    print(paste0("Processing classification level ", l))

    # Define input and output folders for the current level
    newFolder = paste0(open_final_model_path, "/", "Hab_L", l)
    newFolder2 = paste0(save_final_model_path, "/", "Hab_L", l)

    # Load the observed vs predicted data frame -------------------------------
    print("Loading observed vs predicted dataframe")
    FILE1 = paste0(newFolder, "/", "Comparison_Obs_Pred_Final_RF_", type_model, "_model_",
↪   District, "_", Island, "_", Satellite1, "_level_", l, ".csv")
    ObsPred_df <- read.csv(FILE1, sep = ";", dec = ".", stringsAsFactors = FALSE)  # Keep character
↪   columns as character

    # Identify errors between observed and predicted classes ------------------
    print("Creating a dataframe that counts prediction errors (differences between observations and
        ↪ predictions)")

    iobs  <- which(colnames(ObsPred_df) == paste0("Hab_L", l))  # Column index for observed habitat
↪   class
    ipred <- which(colnames(ObsPred_df) == "Ypred")             # Column index for predicted
↪   habitat class

    # Add Error column: "NO" if prediction equals observation, else "YES"
    ObsPred_df$Error <- ifelse(ObsPred_df[, ipred] == ObsPred_df[, iobs], "NO", "YES")

    # Save updated dataframe with errors ----------------------------------------

    FILE2 = paste0(newFolder2, "/", "Local_errors_Final_RF_", type_model, "_model_", District, "_",
↪   Island, "_", Satellite1, "_level_", l, "_EPSG32739.csv")
    write.table(ObsPred_df, file = FILE2, row.names = FALSE, sep = ";", dec = ".")

    # Spatialize misclassified plots ------------------------------------------

    spdfError <- st_as_sf(ObsPred_df, coords = c("xcoord_m", "ycoord_m"), crs = "utm39s")  #
↪   Convert to spatial sf object
    st_write(spdfError, paste0(newFolder2, "/", "Local_errors_Final_RF_", type_model, "_model_",
        ↪ District, "_", Island, "_", Satellite1, "_level_", l, "_EPSG32739.shp"), append = TRUE)  #
        ↪ Save as shapefile
    # Summarize error counts and total observations by class ------------------

    print("Counting the number of 'YES' errors per class and total number of observations")
    colnames(ObsPred_df)[iobs] = "Hab"  # Rename for easier grouping

    Class_error_count <- ObsPred_df %>%
      group_by(Hab) %>%
      summarise(
        Error_count = sum(Error == "YES"),      # Number of errors per class
        total_observations = n()                 # Total samples per class
      )

    print(Class_error_count)

    # Save summary counts per class
    write.table(Class_error_count, file = paste0(newFolder2, "/", "Nb_errors_by_class_level", l,
     ↪  ".csv"), row.names = FALSE, sep = ";", dec = ".")

    # Create stacked barplot of errors per class ------------------------------

    print("Creating stacked barplot showing proportion of errors per class and total observations")

    NOMpng = paste0(newFolder2, "/", "Nb_errors_by_class_Final_RF_", type_model, "_model_",
 ↪  District, "_", Island, "_", Satellite1, "_level_", l, ".png")
    png(file = NOMpng)

    p <- ggplot(Class_error_count, aes(x = Hab)) +
      geom_bar(aes(y = Error_count, fill = "Errors"), stat = "identity") +
      geom_bar(aes(y = total_observations, fill = "Total observations"), stat = "identity", alpha =
       ↪  0.5) +
      labs(title = "Proportion of errors relative to total observations per habitat class",
           x = "Habitat types",
           y = "Number of observations",
           fill = "Count type") +
      scale_fill_manual(values = c("Errors" = "skyblue", "Total observations" = "gray")) +
      theme_classic() +
      theme(
        axis.text.x = element_text(size = 13, angle = 90, hjust = 1),
        axis.text.y = element_text(size = 13),
        axis.title.x = element_text(size = 16),
        axis.title.y = element_text(size = 16),
        plot.title = element_text(size = 22, face = "bold"),
        legend.text = element_text(size = 14),
        legend.title = element_text(size = 16, face = "bold")
      )

    print(p)
    dev.off()

    NOMsvg = paste0(newFolder2, "/", "Nb_errors_by_class_Final_RF_", type_model, "_model_",
 ↪  District, "_", Island, "_", Satellite1, "_level_", l, ".svg")
    svg(file = NOMsvg)
    print(p)
    dev.off()

  } # End of classification level loop

} # End of model type loop
```

In short, this script provides a detailed, multi-level analysis of classification performance by identifying misclassified samples, localizing spatial error patterns, and quantifying per-class error rates. This step is essential to assess model robustness and guide future improvements in habitat mapping workflows.