Step 3.3 Creating Detectable Typology

3.3.2 Computing spectral separability and refining typology

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script aims to evaluate the **spectral(or other variable) separability** between habitat classes across multiple classification levels. It computes pairwise separability metrics, visualizes them through heatmaps, and proposes automatic grouping of spectrally similar classes. The output is used to refine typologies by merging poorly separable classes.

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(spatialEco)  # For spectral.separability() function
library(dplyr)  # For Data manipulation
library(ggplot2)  # For Plotting
library(reshape2)  # For Matrix reshaping for ggplot
library(RColorBrewer)  # For Color palettes
library(igraph)  # For Graph-based grouping of classes
library(rlang)  # For tidy evaluation (sym() function)
```

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

2.4 Set working directory

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

- input learning dataset is located under `"data/Learning_data/PrimaryTypo"`
- output new typology will be saved under `"data/Learning_data/NewTypo"`

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

# Path to open input learning data
open_learning_primary_path = paste0(localscratch, "data/Learning_data/PrimaryTypo")

# Path to save new typology results
save_learning_new_path = paste0(localscratch, "data/Learning_data/NewTypo")
```

2.5 Load learning data

The script reads a CSV file containing spectral and topographic data linked to learning plots. This dataset includes habitat classification labels across multiple levels and a variety of predictor variables derived from remote sensing data.

```
FILE1 <- paste0(open_learning_primary_path, "/Learning_plots_", District, "_", Island,
    "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.csv")
learning_data = read.csv(FILE1, sep = ";", dec = ".", stringsAsFactors = FALSE)  #
 ↪  `stringsAsFactors=F` ensures character strings don't import as factors
```

2.6 Compute spectral separability

At this stage, the script assesses how distinguishable habitat classes are based on their spectral signatures at different typology levels. The objective is to highlight class pairs with strong spectral overlap, which may benefit from regrouping.

To achieve this:

- Classes with fewer than a minimum number of samples (e.g., 5) are excluded to avoid unstable statistical estimates and misleading separability measures.
- Only numeric columns corresponding to imagery variable values are retained, typically starting from the red band (R) onward. This ensures that the separability analysis is based solely on spectral and terrain data and not confounded by non-imagery metadata or categorical variables.
- The `spectral.separability()` function computes **multivariate separability between each class pair**, using the Jeffries-Matusita (JM) distance. This distance ranges from 0 (no separability) to approximately 2 (perfect separability). It accounts for both class means and variances, making it well-suited for remote sensing applications. The resulting symmetric distance matrix is reshaped into long format, retaining only the upper triangle to avoid redundancy.
- A **heatmap** is generated to visually assess separability across all class pairs. This provides an intuitive overview of which classes are difficult to separate and may require regrouping. High separability values (approaching 2) appear in cooler tones, while low values (close to 0) are shown in warmer tones.

Once numeric variable separability has been evaluated, the script identifies and **regroups class pairs that fall below a defined JM threshold** (e.g., 1.2), indicating they are not sufficiently distinguishable:

- Class pairs with JM distances below the threshold are flagged as potentially inseparable. These are assumed to exhibit high variable overlap and may confuse classification algorithms.
- These pairs are represented as edges in an undirected graph, where each class is a node. Connected components of the graph are identified using the *igraph* package, revealing groups of mutually inseparable classes.
- Each connected component (group of inseparable classes) is assigned a **new label** by concatenating the original class names. This approach preserves class identity while explicitly documenting the regrouping process.

While multivariate separability evaluates all bands jointly, **per-band (univariate) analysis** examines the contribution of each band individually:

- The JM distance is computed separately for each band. This helps to identify which bands are most informative for class discrimination and which provide little to no separability.
- Each band's separability matrix is visualized through a dedicated heatmap. These maps are saved in both PNG and SVG formats to accommodate different usage contexts (e.g., raster display or publication-quality vector editing).

- This analysis provides valuable information for potential dimensionality reduction or band selection strategies. Bands that consistently offer low separability across all class pairs may be excluded from future workflows to reduce noise.

```r
# Define threshold for minimum number of samples required per class
min_samples_class = 5

# Loop over each typology level
for (l in seq(1:maxTypoLevel)) {

    cat("Processing class level:", l, "\n")

    # Create output folder for current level
    LevelFolder <- paste0(open_learning_primary_path, "/", "Hab_L", l)
    dir.create(LevelFolder, showWarnings = FALSE)

    # Remove low frequency class
    # -----------------------------------------------------------

    # Filter out classes with fewer samples than threshold
    class_col = paste0("Hab_L", l)
    class_counts <- table(learning_data[, class_col])
    keep_classes <- names(class_counts[class_counts >= min_samples_class, ])
    Filtered_learning_data <- dplyr::filter(learning_data, !!sym(class_col) %in%
        keep_classes)

    # Select spectral features: from 'R' column to the end, keeping only
    # numeric columns
    spectral_vars <- Filtered_learning_data %>%
        select(which(colnames(.) == "R"):ncol(.)) %>%
        select_if(is.numeric)  # Keep only numeric columns

    # Extract class labels as factor
    classes <- as.factor(Filtered_learning_data[[class_col]])

    # Multivariate spectral separability
    # -----------------------------------------------------------

    # Compute multivariate separability using Jeffries-Matusita distance
    sep_multi <- spectral.separability(spectral_vars, classes, jeffries.matusita = TRUE)
    print(sep_multi)

    # Convert separability matrix to long format keeping upper triangle only
    # (to avoid duplicates)
    sep_multi_mep <- melt(sep_multi)
    colnames(sep_multi_mep) <- c("Class1", "Class2", "Separability")
    sep_multi_mep <- sep_multi_mep[as.character(sep_multi_mep$Class1) <
↪   as.character(sep_multi_mep$Class2),
        ]
    print(sep_multi_mep)

    # Save separability matrix as CSV
    FILE2 <- paste0(LevelFolder, "/Multibands_seperability_", District, "_", Island,
        "_", Satellite1, "_", Year1, "_level_", l, "_ALL_SOURCES_EPSG32739.csv")
    write.table(sep_multi_mep, FILE2, sep = ";", dec = ".", row.names = FALSE)

    # Plot multiband spectral separability heatmap
    png_name <- paste0(LevelFolder, "/Multibands_seperability_", District, "_", Island,
        "_", Satellite1, "_", Year1, "_level_", l, "_ALL_SOURCES_EPSG32739.png")
```

```r
png(file = png_name, width = 1000, height = 600)
p_multi <- ggplot(sep_multi_mep, aes(x = Class1, y = Class2, fill = Separability)) +
    geom_tile(color = "white") + scale_fill_gradientn(colours = brewer.pal(9,
    "YlGnBu"), name = "Separability") + theme_minimal() + labs(title = paste("Spectral
     ↪  Separability (Multiband) -",
    class_col), x = "Class", y = "Class") + theme(axis.text.x = element_text(angle = 45,
    hjust = 1))
print(p_multi)
dev.off()

svg_name <- paste0(LevelFolder, "/Multibands_seperability_", District, "_", Island,
    "_", Satellite1, "_", Year1, "_level_", l, "_ALL_SOURCES_EPSG32739.svg")
svg(file = svg_name)
print(p_multi)
dev.off()

# Group classes with low separability (JM <
# threshold)----------------------------------------------------------------------

# Define a threshold below which classes are not considered separable
threshold_sep <- 1.2  # You can change this value

# Find class pairs below threshold
low_sep <- sep_multi_mep %>%
    filter(Separability < threshold_sep)

# Initialize fusion_lookup with identity (each class maps to itself)
unique_classes <- unique(Filtered_learning_data[[class_col]])
fusion_lookup <- data.frame(original = unique_classes, new_class = unique_classes,
    stringsAsFactors = FALSE)

# If low separability pairs exist, create groups by connectivity
if (nrow(low_sep) > 0) {
    g <- graph_from_data_frame(low_sep[, c("Class1", "Class2")], directed = FALSE)
    comps <- components(g)
    fusion_map <- data.frame(original = names(comps$membership), group_id = comps$membership,
        stringsAsFactors = FALSE)
    fusion_map <- fusion_map %>%
        group_by(group_id) %>%
        mutate(new_class = paste0(sort(unique(original)), collapse = "")) %>%
        ungroup()

    # Update fusion_lookup for classes involved in regrouping
    for (i in 1:nrow(fusion_map)) {
        fusion_lookup$new_class[fusion_lookup$original == fusion_map$original[i]] <-
↪   fusion_map$new_class[i]
    }
}

# Create new corrected class column with regrouped classes
col_corr_name <- paste0(class_col, "_corr")
Filtered_learning_data[[col_corr_name]] <-
↪   fusion_lookup$new_class[match(Filtered_learning_data[[class_col]],
    fusion_lookup$original)]

# Univariate spectral separability (per
# band)----------------------------------------------------------------------

sep_by_band <- lapply(1:ncol(spectral_vars), function(i) {
```

```
        spectral.separability(spectral_vars[, i, drop = FALSE], classes, jeffries.matusita = TRUE)
    })
    names(sep_by_band) <- colnames(spectral_vars)

    # Plot heatmaps for each band separately
    for (i in 1:length(sep_by_band)) {
        sep_band_melt <- melt(sep_by_band[[i]])
        colnames(sep_band_melt) <- c("Class1", "Class2", "Separability")

        png_name <- paste0(LevelFolder, "/", names(sep_by_band)[i], "_band_separability_",
            District, "_", Island, "_", Satellite1, "_", Year1, "_level_", l,
↪   "_ALL_SOURCES_EPSG32739.png")
        png(file = png_name, width = 1000, height = 600)
        p_band <- ggplot(sep_band_melt, aes(x = Class1, y = Class2, fill = Separability)) +
            geom_tile(color = "white") + scale_fill_gradientn(colours = brewer.pal(9,
            "YlOrRd"), name = "Separability") + theme_minimal() + labs(title = paste("Band:",
            names(sep_by_band)[i], "-", class_col), x = "Class", y = "Class") + theme(axis.text.x =
↪            element_text(angle = 45,
            hjust = 1))
        print(p_band)
        dev.off()

        svg_name <- paste0(LevelFolder, "/", names(sep_by_band)[i], "_band_separability_",
            District, "_", Island, "_", Satellite1, "_", Year1, "_level_", l,
↪   "_ALL_SOURCES_EPSG32739.svg")
        svg(file = svg_name)
        print(p_band)
        dev.off()
    }
}
```

2.7 Export corrected dataset with regrouped classes

After spectral grouping and analysis are complete, a new version of the learning dataset is created, integrating the corrected class labels.

- Merging Original and Corrected Labels : The new dataset retains all original attributes but includes the corrected class columns (`_corr`) resulting from the spectral grouping process.
- Saving the Corrected Dataset : The dataset is exported as a .csv file, with a filename that includes relevant metadata (district, island, sensor, year, and typology level). This ensures consistency and clarity for downstream processing or archiving.

```
learning_data_corrected <- Filtered_learning_data %>%
    select(everything(), matches("_corr$"))

FILE3 = paste0(save_learning_new_path, "/", "Learning_plots_", District, "_", Island,
    "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.csv")
write.table(learning_data_corrected, file = FILE3, sep = ";", dec = ".", row.names = FALSE)
```