Step 3.3 Creating Detectable Typology

3.3.1.b Hierarchical clustering of habitat classes

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script aims to perform Hierarchical Clustering (HC) on habitat classification levels using spectral and topographic features derived from remote sensing data. The goal is to:

- Explore the spectral and topographic similarity between habitat types across different classification levels;
- Identify natural groupings (clusters) of habitat classes without predefining the number of clusters;
- Generate dendrograms based on both mean and median feature values for each habitat class, allowing visual interpretation of the clustering structure;
- Produce and save the clustering outputs (figures and objects) for further ecological or remote sensing analyses.

The process is repeated for each habitat typology level (from level 1 to the maximum specified), allowing a multiscale analysis of spectral/topographic patterns in habitat classifications.

2 Script explanation

2.1 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)
```

2.2 Load required packages

```r
library(FactoMineR)  # For multivariate data analysis (not directly used here but often coupled
↪  with factoextra)
library(factoextra)  # For visualizing clustering and multivariate data
library(dplyr)  # For data manipulation (e.g., filtering, summarizing)
library(ggplot2)  # For plotting
library(stats)  # For statistical functions such as clustering, distance calculations, and other
↪  classic statistical methods (e.g., hclust, dist)
```

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 from clustering will be saved under `"data/Learning_data/PrimaryTypo"`

```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 violin plot results
save_learning_primary_path = paste0(localscratch, "data/Learning_data/PrimaryTypo")
```

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.

```r
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 Perform Hierarchical Clustering of habitat classes for each typology level

Hierarchical clustering is a technique used to explore and understand the structure of data by identifying natural groupings within it, without the need to specify the number of clusters in advance. In this context, it is used to identify spectral/terrain similarities between habitat classes.

- Each observation is initially treated as an individual cluster. If there are *Ntotal* observations, the process starts with *Ntotal* separate clusters.
- The distance between each pair of clusters is then computed. This distance can be measured in various ways, such as Euclidean distance, Manhattan distance, etc.
- The two clusters that are closest to each other (i.e. with the smallest distance) are merged to form a new cluster. The total number of clusters is thus reduced by one.
- After each merge, the distances between the newly formed cluster and all the remaining clusters are recalculated.
- The results of the HC are typically visualized using a dendrogram, a tree-like diagram that illustrates the successive merging steps.

Each level of the dendrogram represents a stage in the clustering process. By cutting the dendrogram at a specific height, one can choose the desired number of clusters.

```r
# Loop through typology levels (from 1 to maxTypoLevel)
for (l in seq(1:maxTypoLevel)) {

    print(paste0("Processing typology level: ", l))

    # Create output folder for this classification level
    LevelFolder = paste0(save_learning_primary_path, "/", "Hab_L", l)
    dir.create(LevelFolder, showWarnings = F)  # Avoid warning if folder already exists
```

```r
# Define columns to remove before analysis (ID and spatial coordinates)
variables_to_remove <- c("ID", "xcoord_m", "ycoord_m")
print(variables_to_remove)

# Remove unwanted columns from the dataset
spectral_df <- learning_data[, !(names(learning_data) %in% variables_to_remove)]

# Rename current classification level column to a common name
colnames(spectral_df)[colnames(spectral_df) == paste0("Hab_L", l)] <- "hab_col"

# Compute mean and median for each habitat class
means_df <- aggregate(. ~ hab_col, data = spectral_df, FUN = mean)
medians_df <- aggregate(. ~ hab_col, data = spectral_df, FUN = median)

# Assign row names using the habitat labels (for clustering)
rownames(means_df) <- means_df$hab_col
rownames(medians_df) <- medians_df$hab_col

# Keep only quantitative variables for clustering (remove label column)
quantitative_vars_means <- means_df[, -1]
quantitative_vars_medians <- medians_df[, -1]

# Perform hierarchical clustering (using Euclidean distance)
hc_means <- hclust(dist(quantitative_vars_means, method = "euclidean"))
hc_medians <- hclust(dist(quantitative_vars_medians, method = "euclidean"))

# Define maximum number of clusters for dendrogram visualization
n_classes <- nrow(quantitative_vars_means)
n_classes_max = 50
k_clusters <- min(n_classes_max, n_classes)  # Ensure the number of clusters does not exceed
↪ the number of classes

# Plot dendrogram (means)
png(filename = paste0(LevelFolder, "/Means_HC_", District, "_", Island, "_",
    Satellite1, "_level_", l, ".png"))
dendro_plot_means <- fviz_dend(hc_means, cex = 0.5, k = k_clusters, color_labels_by_k = TRUE,
    rect = TRUE) + labs(title = paste("Dendrogram of Means - Level", l))
print(dendro_plot_means)
dev.off()

NOMsvg = paste0(LevelFolder, "/", "Means_HC_", District, "_", Island, "_", Satellite1,
    "_level_", l, ".svg")
svg(file = NOMsvg)
print(dendro_plot_means)
dev.off()

# Plot dendrogram (medians)
png(filename = paste0(LevelFolder, "/Medians_HC_", District, "_", Island, "_",
    Satellite1, "_level_", l, ".png"))
dendro_plot_medians <- fviz_dend(hc_medians, cex = 0.5, k = k_clusters, color_labels_by_k =
↪ TRUE,
    rect = TRUE) + labs(title = paste("Dendrogram of Medians - Level", l))
print(dendro_plot_medians)
dev.off()

NOMsvg = paste0(LevelFolder, "/", "Medians_HC_", District, "_", Island, "_",
    Satellite1, "_level_", l, ".svg")
svg(file = NOMsvg)
```

```
    print(dendro_plot_medians)
    dev.off()


}  # End of loop for typology levels
```