Step 3.3 Creating Detectable Typology

3.3.1.a Analyzing variable distribution within classes

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script aims to analyze the distribution of spectral and topographic variables across different habitat classification levels using **violin plots**. It normalizes predictor variables, computes summary statistics for each habitat class, and generates visualizations to explore variability and central tendencies of environmental variables. The results, including summary statistics and plots, are saved for further ecological interpretation and reporting.

Violin plot analysis allows to:

- Identify spectral similarities between classes
- Detect outliers
- Identify habitats that are poorly discriminated by the variables (too much variation within a variable)

A violin plot is a type of graph used in statistics to visualize the distribution of a numerical variable. It shows the shape of the data distribution, including its density, and incorporates elements of boxplots such as the median, quartiles, and outliers.

It represents the shape of a dataset using a probability density function (PDF) or a density plot, which is essentially a smoothed histogram. The width of the density plot indicates how frequently a value appears in the dataset. In other words, wider regions of the density plot correspond to more frequent values, while narrower regions indicate less frequent values.

For more explanation on how to interpret a violin plot, click here: https://www.labxchange.org/library/items/lb:LabXchange:46f64d7a:html:1

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

```
library(dplyr)  # For data manipulation
library(ggplot2)  # For plotting
```

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

```
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 violin plots will be saved under `"data/Learning_data/PrimaryTypo"`

```
# 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.

```
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 Analyze distribution of each predictor value across habitat levels

For each habitat classification level (from 1 up to a predefined maximum), the script:

- Creates a dedicated directory to save outputs related to that classification level.
- Normalizes spectral and topographic variables on a scale from 0 to 1 to ensure comparability across different units and scales.
- Extracts the relevant variable names for analysis.

Then, for each variable within the normalized dataset:

- The script computes detailed summary statistics (mean, median, quartiles, standard deviation, standard error, minimum, and maximum) grouped by habitat classes. These summaries are saved as CSV files.
- It then generates violin plots to visualize the distribution of variable values across habitat classes. These plots combine violin shapes with boxplots and overlaid mean and median markers to provide comprehensive visual summaries of the data distribution.
- Y-axis and x-axis limits are adjusted according to predefined ranges tailored for each variable to optimize plot readability.
- Plots are saved both as PNG and SVG files for flexibility in usage and publication.

```
# Loop through each habitat classification level
for (l in seq(1:maxTypoLevel)) {

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

    # Create a directory to store results for the current level
    LevelFolder = paste0(save_learning_primary_path, "/", "Hab_L", l)
    dir.create(LevelFolder, showWarnings = F)  # ShowWarnings=F to remove warnings message if file
     ↪  already exists

    # Normalize spectral/topographic variables (between 0 and 1)
    ihab = which(colnames(learning_data) == paste0("Hab_L", l))
    ihab4 = which(colnames(learning_data) == "Hab_L4")
    learning_normalized <- learning_data %>%
        mutate(across((ihab4 + 1):(ncol(learning_data) - 2), ~(. - min(.))/(max(.) -
            min(.))))

    # Get list of variable names to analyze
    variable_list <- colnames(learning_normalized)[(ihab4 + 1):ncol(learning_normalized)]
```

```
# Loop through each variable
for (v in variable_list) {

    # Compute statistics for each habitat
    # class--------------------------------

    print(paste0("Computing statistics for variable: ", v))

    # Prepare data for current variable
    spectral_df <- learning_normalized %>%
        rename(Type_habitat = paste0("Hab_L", l)) %>%
        mutate(Type_habitat = as.factor(Type_habitat))

    # Compute summary statistics per habitat class
    summary_stats <- spectral_df %>%
        group_by(Type_habitat) %>%
        summarise(Mean = mean(get(v), na.rm = TRUE), Median = median(get(v),
            na.rm = TRUE), Q1 = quantile(get(v), 0.25, na.rm = TRUE), Q3 = quantile(get(v),
            0.75, na.rm = TRUE), SD = sd(get(v), na.rm = TRUE), SE = SD/sqrt(n()),
            Min = min(get(v), na.rm = TRUE), Max = max(get(v), na.rm = TRUE)) %>%
        mutate(Variable = v)

    # Save summary statistics
    FILE2 = paste0(LevelFolder, "/", "Variable_distribution_", District, "_",
        Island, "_", Satellite1, "_", Year1, "_", v, "_L", l, "_ALL_SOURCES.csv")
    write.table(summary_stats, FILE2, sep = ";", dec = ".", row.names = FALSE)

    # Generate violin plot for current variable
    # ---------------------------------

    print(paste0("Generating violin plot for variable: ", v))

    ivar = which(colnames(learning_normalized) == v)
    ihab = which(colnames(learning_normalized) == paste0("Hab_L", l))

    # Define y-axis and x-axis limits based on variable type
    ylim_values <- c(B = 0.5, R = 0.5, G = 0.5, Brightness = 0.5, GCCI = 0.5,
        NIR = 1, NDWI = 0.75, NDVI = 1, VARI = 0.75, BSI = 0.475, Dtm = 600,
        Slope = 75)
    xlim_values <- c(B = 0, R = 0, G = 0, Brightness = 0, GCCI = 0, NIR = 0,
        NDWI = 0, NDVI = 0.4, VARI = 0.6, BSI = 0.425, Dtm = 0, Slope = 0)

    # Set default axis limits if variable is not in predefined list
    default_ylim <- 1
    default_xlim <- 0
    ylim <- ylim_values[v] %>%
        coalesce(default_ylim)
    xlim <- xlim_values[v] %>%
        coalesce(default_xlim)

    # Violin plots with png format
    NOMpng = paste0(LevelFolder, "/", "Variable_distribution_", District, "_",
        Island, "_", Satellite1, "_", Year1, "_", v, "_L", l, "_ALL_SOURCES.png")
    png(file = NOMpng, width = 1000, height = 600)
    p <- ggplot(learning_normalized, aes_string(x = paste0("Hab_L", l), y = v)) +
        geom_violin(aes_string(fill = paste0("Hab_L", l), color = paste0("Hab_L",
            l)), position = position_dodge(), draw_quantiles = c(0), show.legend = TRUE) +
        geom_boxplot(width = 0.1, color = "black", fill = "white", position =
         ↪  position_dodge(width = 0.9),
```

```
            show.legend = FALSE) + stat_summary(fun = mean, geom = "crossbar",
        width = 0.1, aes(color = "Mean")) + stat_summary(fun = median, geom = "point",
        size = 2, aes(color = "Median")) + scale_color_manual(name = "Statistics",
        values = c(Mean = "blue", Median = "red")) + theme_classic() + theme(legend.position =
        ↪  "right",
        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")) + labs(title = paste("Level ",
        l, "- Variable :", v), fill = "Habitat classes", x = "Habitat classes",
        y = paste0("Variable value ", v)) + scale_y_continuous(expand = c(0,
        0), limits = c(xlim, ylim))

    print(p)
    dev.off()

    # Violin plots with svg format
    NOMsvg = paste0(LevelFolder, "/", "Variable_distribution_", District, "_",
        Island, "_", Satellite1, "_", Year1, "_", v, "_L", l, "_ALL_SOURCES.svg")
    svg(file = NOMsvg)
    print(p)
    dev.off()

  }  # End of variable loop


}  # End of classification level loop
```

This systematic approach enables thorough exploration of spectral and topographic variable patterns across habitat classes, facilitating ecological insights and supporting further modeling or monitoring efforts.