Step 3.2 Selecting relevant predictors

3.2.2 Creating filtered final raster stack

Diane ESPEL

2025-06-18

Contents

1 Objectives

This script aims to create a final raster stack (called `raster_stack`).by loading and filtering multispectral and topographic raster layers for one or multiple years. It keeps only the relevant predictor variables identified from prior analysis (see script 3.2.1), ensuring that the raster stack is optimized for subsequent ecological modeling or monitoring. The final filtered raster stack is saved in both R data format (.Rdata) and GeoTIFF format (.TIF) to facilitate easy reuse and integration with other tools.

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(terra)  # For handling raster data and spatial operations
library(resample)  # For resampling raster layers to match resolution
```

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 `"Satellite2"`: the name of satellite providing terrain imagery
- the `"Year1"`: the acquisition year of the multispectral imagery
- the `"Month1"`: the acquisition month of the multispectral imagery
- the `"Year2"`: the acquisition year of terrain imagery
- the `"Res1"`: the resolution of the multispectral imagery
- the `"Res2"`: the resolution of terrain imagery

```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
Month1 = "02"  # acquisition month of multispectral imagery
Res1 = "50cm"  # spatial resolution of multispectral imagery
Satellite2 = "SRTM"  # satellite name of DEM
Year2 = "2012"  # acquisition year of DEM
Res2 = "30m"  # spatial resolution of DEM
```

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 results from PCA is located under `"data/Learning_data/PrimaryTypo"`
- input rasters is located under `"data/raster/Cut_image"`
- output raster stack will be saved under `"data/raster/Cut_image"`

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

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

# Path to open input raster layers or stack
open_cut_raster_path = paste0(localscratch, "data/raster/Cut_image")

# Path to save the final raster stack
save_cut_raster_path = paste0(localscratch, "data/raster/Cut_image")
```

2.5 Open selected variables data

The script reads a CSV file containing learning plot data, which includes previously selected predictor variables for modeling. Habitat classes and coordinate columns are removed to retain only the relevant variables. The names of these variables are extracted (`list_variables`), serving as a filter to keep only necessary raster layers in subsequent steps.

```r
# Load selected variables used for model training
FILE1 <- paste0(open_learning_primary_path, "/Selected_learning_plots_", District,
    "_", Island, "_", Satellite1, "_", Year1, "_ALL_SOURCES_EPSG32739.csv")
learning_data <- read.csv(FILE1, sep = ";", dec = ".", stringsAsFactors = FALSE)

# Drop habitat classes and coordinate/id columns to keep only predictor
# variables
variable_df <- learning_data %>%
    select(-matches("^Hab_L[1-4]$"), -id, -xcoord_m, -ycoord_m)

# Extract variable names
list_variables <- names(variable_df)
cat("Selected variables:", paste(list_variables, collapse = ", "), "\n")
```

2.6 Stack rasters

This section aims to stack the selected variable layers. To achieve this, the script:

- **Identify Available Years of Satellite Imagery :** Using file name patterns, the script detects all years for which raster data from the main satellite (e.g., Pleiades) are available in the input directory. This allows the processing loop to dynamically adapt to the dataset's temporal coverage.
- **Loop Over Available Years to Build Raster Stacks:**
  - Reference Year (e.g., 2022): For the reference year, which is typically used for model training, the full raster stack file is loaded and layer names are assigned (e.g., Red, Green, Blue, NIR, NDVI, DEM, Slope). Only the layers corresponding to the selected predictor variables are retained.This filtered raster stack is then saved both as an R data object and a GeoTIFF.
  - Other Years (Temporal Monitoring): For additional years, the script reconstructs the raster stack by loading each selected variable separately. It first tries to find the raster for the variable in the main satellite dataset for that year. If not found, it attempts to load a fallback raster from a secondary satellite source (e.g., SRTM for DEM-related layers). When necessary, lower-resolution rasters are resampled to match the higher resolution of the main dataset. Once all variables are gathered, they are stacked together and saved similarly as for the reference year.

Warnings are issued when expected raster files are missing or if resampling cannot be performed due to a missing reference raster. This ensures transparency during processing and helps detect potential data gaps.

```r
# Identify available years for Satellite1 rasters based on file names
all_satellite1_files <- list.files(open_cut_raster_path, pattern = paste0(District,
    "_", Island, "_", Satellite1, "_\\d{4}_.+_", Res1, "_.*\\.TIF$"))
available_satellite1_years <- sort(unique(str_extract(all_satellite1_files, "\\d{4}")))
available_satellite1_years <- available_satellite1_years[!is.na(available_satellite1_years)]
cat("Detected years:", paste(available_satellite1_years, collapse = ", "), "\n")


# Loop through each available year to create raster stacks
```

```r
for (Year in available_satellite1_years) {

    # Special treatment for Year1 (reference year)
    if (Year == Year1) {

        print("The raster stack will be used for model construction")

        cat("Processing Year:", Year, "\n")

        # Load full raster stack for this year and assign layer names
        raster_total <- rast(paste0(open_cut_raster_path, "/", District, "_", Island,
            "_", Satellite1, "_Total_raster_stack_", Year, "_", Res1, "_cut.TIF"))
        expected_names <- c("R", "G", "B", "NIR", "NDVI", "GRVI", "VARI", "GCCI",
            "Brightness", "BSI", "NDWI", "Dtm", "Slope")

        # Only rename layers if the number of layers matches
        if (nlayers(raster_total) == length(expected_names)) {
            names(raster_total) <- expected_names
        } else {
            warning("Unexpected number of layers in total raster stack")
        }

        # Keep only selected layers used in modeling
        raster_total_filtered <- raster_total[[list_variables]]

        # Save filtered raster as RData and GeoTIFF
        save(raster_total_filtered, file = paste0(save_cut_raster_path, "/", District,
            "_", Island, "_Final_raster_stack_", Year, "_", Res1, "_cut.Rdata"))
        writeRaster(raster_total_filtered, paste0(save_cut_raster_path, "/", District,
            "_", Island, "_Final_raster_stack_", Year, "_", Res1, "_cut.TIF"), overwrite = TRUE)

    } else {

        print("The raster stack will be used for temporal monitoring")

        # For other years, reconstruct stack layer by layer
        cat("\nProcessing year:", Year, "\n")

        raster_list <- list()  # Initialize list of rasters

        for (var in list_variables) {

            # Search for the raster file in Satellite1 (main)
            pattern_year <- paste0(District, "_", Island, "_", Satellite1, "_", Year,
                "_", Res1, "_", var, "_cut\\.TIF$")
            file_year <- list.files(open_cut_raster_path, pattern = pattern_year,
                full.names = TRUE)

            # If not found, search in Satellite2 (fallback)
            pattern_year2 <- paste0(District, "_", Island, "_", Satellite2, "_",
                Year2, "_", Res2, "_", var, "_cut\\.TIF$")
            file_year2 <- list.files(open_cut_raster_path, pattern = pattern_year2,
                full.names = TRUE)

            # If found in Satellite1
            if (length(file_year) > 0) {
                raster_ref <- rast(file_year[1])
                raster_list[[var]] <- raster_ref
                # If only found in Satellite2 (SRTM for example)
            } else if (length(file_year2) > 0) {
                raster_year2 <- rast(file_year2[1])

                # If a reference raster is already loaded, use it for
                # resampling
                if (exists("raster_ref")) {
                  if (res(raster_year2)[1] > res(raster_ref)[1]) {
                    cat("Resampling", var, "from", Satellite2, "to match higher resolution\n")
                    raster_resampled <- resample(raster_year2, raster_ref)  # Match resolution
                    raster_list[[var]] <- raster_resampled
                  } else {
                    raster_list[[var]] <- raster_year2
                  }
                } else {
                  warning(paste("Cannot resample", var, ": no reference raster defined."))
                  raster_list[[var]] <- raster_year2
                }

            } else {
                warning(paste("No raster found for variable", var, "in", Year, "or fallback year",
                  Year2))
            }

        }  # end variable loop

        # If at least one variable was loaded, stack and save
        if (length(raster_list) > 0) {
            raster_stack <- rast(raster_list)

            # Save final stack for this year
            save(raster_stack, file = paste0(save_cut_raster_path, "/", District,
                "_", Island, "_Final_raster_stack_", Year, "_", Res1, "_cut.Rdata"))
            writeRaster(raster_stack, paste0(save_cut_raster_path, "/", District,
                "_", Island, "_Final_raster_stack_", Year, "_", Res1, "_cut.TIF"),
                overwrite = TRUE)

        } else {
            warning(paste("No variables found for year", Year, "- stack not generated."))
        }

    }  # end if/else block

}  # end year loop
```

This approach ensures that the final raster stacks are both spectrally and spatially consistent, filtered to include only relevant variables, and ready for modeling or temporal analysis of habitat distribution.