Step 1.1 Creating field-based plots

1.1.2.b Correcting field-based plots location

Diane ESPEL

2025-07-22

Contents

1 Objectives

This R script aims to generate square quadrat plots of fixed size (4m sides) centered on ecological sampling points from both recent field and historical datasets. The main tasks include:

- Loading and merging original and corrected centroid data for each data source (recent `"FIELD"` and historical `"HFI"`).
- Applying coordinate corrections to GPS points.
- Computing quadrat corner coordinates based on corrected centroids.
- Creating spatial polygons (quadrats) from these coordinates.
- Exporting corrected centroids and quadrat polygons as shapefiles for GIS use.

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

```
library(sp)           # Classes and methods for spatial data
library(sf)           # Simple features for spatial data (modern alternative)
library(dplyr)        # Data manipulation verbs (filter, join, select, etc.)
library(data.table)   # Efficient data handling; here mainly for rbindlist()
```

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)

```
District='CRO' # 3-letter code for the archipelago
Island='POS'   # 3-letter code for the island
```

2.4 Set working directory

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

- input sample file is located under `"/data/samples"`
- outputs (e.g., plots or shapefiles) will be saved under `"/data/vector/Plots/PrimaryTypo"`

You can create additional subfolders within this structure to organize your outputs by data type, source, or year of analysis.

```
# Define local root directory
#localHOME = paste0("your_local_path/")
localHOME=paste0("/home/genouest/cnrs_umr6553/despel/CARTOVEGE/")

# Define path to load input data
open_plots_path = paste0(localHOME, "data/vector/Plots/PrimaryTypo")

# Define path to save generated outputs
save_plots_path = paste0(localHOME, "data/vector/Plots/PrimaryTypo")
```

2.5 Create field reference samples plots

The script processes two sources: `"FIELD"` (recent field samples) and `"HFI"` (historical field samples) in 7 main steps:

1) Load Sample Files

- Define the file paths for each source and data type (original and corrected centroids).

- Load the CSV files into data frames.
- Print the number of unique observation IDs for verification.

2) Replace Coordinates with Corrected Centroids

- Select relevant columns from the original sample file.
- Join corrected coordinates from the second file by observation ID (N_obs).
- Replace original coordinates (latitude, longitude, X, Y) with corrected ones.

3) Compute New Quadrat Corner Coordinates

- Define quadrat size (e.g., 4 meters side).
- Compute coordinates for corners of each square (N, S, E, W).
- Ensure no duplicated observation IDs.

4) Save Corrected Centroids

- Write the updated centroid table (with corner coordinates) to a new CSV file.

5) Draw and Save Quadrats

- Generate square polygons clockwise from NW corner.
- Build `SpatialPolygons` assigning each an ID and then a `SpatialPolygonsDataFrame`.
- Join back sample attributes to each polygon from the original observations (coordinates, date, source, habitat levels, etc.).
- Convert the quadrats to `sf` object and export to shapefile.

6) Export Results

- Export new quadrat polygons and centroid points as `.shp` files using `st_write()` from the `sf` package.

```r
# Create field reference samples plots-------------------------------------------------------

# Define list of sources: FIELD = recent samples, HFI = historical samples
source_list=list("FIELD","HFI")

# Loop over source list
for (source in source_list) {

  print(paste0("Working on source: ", source))

  # Load input CSV files -------------------------------------------------------

  FILE1 = paste0(open_plots_path, "/Quadrats_", District, "_", Island, "_ALL_", source,
↪  "_SAMPLES_Centroids_EPSG32739.csv")
  FILE2 = paste0(open_plots_path, "/Quadrats_", District, "_", Island, "_ALL_", source,
↪  "_SAMPLES_Centroids_correctedOnQgis_EPSG32739.csv")

  # Read original and corrected centroid data from CSVs
  Sample_points <- read.csv(FILE1, sep=";", dec=".", stringsAsFactors=FALSE)
  New_centroids <- read.csv(FILE2, sep=";", dec=".", stringsAsFactors=FALSE)

  print(paste0("Number of unique observations: ", length(unique(Sample_points$N_obs))))
```

```r
# Replace coordinates with corrected ones ------------------------------------

# Keep only relevant columns before merging
col_to_keep = c("N_obs","Date","Protocole","Source","Longitude","Latitude","xcoord_m","ycoord_m",
                "Surface","Longueur_m","Largeur_m","Hab_L1","Hab_L2","Hab_L3","Hab_L4")
New_sample_points = Sample_points[, col_to_keep]

# Join corrected coordinates from New_centroids by observation ID (N_obs)
joined_points <- dplyr::inner_join(New_sample_points,
                                   New_centroids %>%
↪  dplyr::select("N_obs","xcoord_m","ycoord_m","Longitude","Latitude"),
                                   by = "N_obs")

# Select corrected columns and rename for clarity
col_to_keep =
↪  c("N_obs","Date","Protocole","Source","Longitude.y","Latitude.y","xcoord_m.y","ycoord_m.y",
                "Surface","Longueur_m","Largeur_m","Hab_L1","Hab_L2","Hab_L3","Hab_L4")
final_points <- joined_points[, col_to_keep]
colnames(final_points) <- c("N_obs","Date","Protocole","Source","Longitude","Latitude",
                            "xcoord_m","ycoord_m","Surface","Longueur_m","Largeur_m",
                            "Hab_L1","Hab_L2","Hab_L3","Hab_L4")

# Compute quadrat corner coordinates -----------------------------------------

print("Defining quadrat side length (m) and calculating corner coordinates")
length_quadrat <- 4
radius <- length_quadrat / 2
poly_points <- data.frame(final_points)

poly_points$yPlus <- poly_points$ycoord_m + radius   # North edge
poly_points$xPlus <- poly_points$xcoord_m + radius   # East edge
poly_points$yMinus <- poly_points$ycoord_m - radius  # South edge
poly_points$xMinus <- poly_points$xcoord_m - radius  # West edge

# Check for duplicate observation IDs
print("Checking for duplicate observation IDs (should be unique)")
duplicates <- which(duplicated(poly_points$N_obs) | duplicated(poly_points$N_obs, fromLast=TRUE))
print(duplicates)

# Build quadrat polygons clockwise (NW → NE → SE → SW → NW) ------------------

quadrats <- cbind(poly_points$xMinus, poly_points$yPlus,    # NW corner
                  poly_points$xPlus, poly_points$yPlus,     # NE corner
                  poly_points$xPlus, poly_points$yMinus,    # SE corner
                  poly_points$xMinus, poly_points$yMinus,   # SW corner
                  poly_points$xMinus, poly_points$yPlus)    # Close polygon at NW corner

ID <- poly_points$N_obs  # Extract IDs
print("Creating spatial polygons (quadrats)")

# Create SpatialPolygons object from quadrats matrix and ID vector
polysHabitat <- SpatialPolygons(
  mapply(function(poly, id) {
    # Convert polygon coordinate vector into a 2-column matrix (x, y coordinates)
    xy <- matrix(poly, ncol=2, byrow=TRUE)
    # Create a Polygon object from the coordinates, then wrap it into Polygons with the given ID
    Polygons(list(Polygon(xy)), ID=id)
  },
```

```r
    # Apply the function to each row of 'quadrats' matrix and corresponding ID
    split(quadrats, row(quadrats)), ID),
    # Define the Coordinate Reference System (CRS) for the polygons (UTM zone 39S / EPSG:32739)
    proj4string = CRS("+init=EPSG:32739")
  )


  # Create a SpatialPolygonsDataFrame by combining the polygons and metadata ---

  # Construct SpatialPolygonsDataFrame using the polygons and a data.frame of IDs (set as row
  ↪  names)
   polys.df <- SpatialPolygonsDataFrame(polysHabitat, data.frame(N_obs=ID, row.names=ID))

  # Add additional attribute columns from the 'poly_points' dataframe to the
  ↪  SpatialPolygonsDataFrame
  polys.df$xcoord_m <- poly_points$xcoord_m       # X coordinate in meters
  polys.df$ycoord_m <- poly_points$ycoord_m       # Y coordinate in meters
  polys.df$Longitude <- poly_points$Longitude     # Geographic longitude
  polys.df$Latitude <- poly_points$Latitude       # Geographic latitude
  polys.df$Date <- poly_points$Date               # Date of observation or sampling
  polys.df$Source <- poly_points$Source           # Source or origin of data
  polys.df$Surface <- poly_points$Surface         # Surface area of the quadrat
  polys.df$Hab_L1 <- poly_points$Hab_L1           # Habitat classification level 1
  polys.df$Hab_L2 <- poly_points$Hab_L2           # Habitat classification level 2
  polys.df$Hab_L3 <- poly_points$Hab_L3           # Habitat classification level 3
  polys.df$Hab_L4 <- poly_points$Hab_L4           # Habitat classification level 4

  # Save the final shapefiles  -------------------

  # Export polygons as shapefile
  polys.sf <- st_as_sf(polys.df)  # Convert SpatialPolygonsDataFrame to sf object for modern
↪  spatial operations and writing
  st_write(polys.sf, paste0(save_plots_path, "/Quadrats_", District, "_", Island, "_ALL_", source,
  ↪  "_SAMPLES_Polygons_corrected_EPSG32739.shp"),
           driver = "ESRI Shapefile", append = FALSE)

  # Get centroids from polygons
  pts.in.polys <- sf::st_centroid(polys.sf)
  st_write(pts.in.polys, paste0(save_plots_path, "/Quadrats_", District, "_", Island, "_ALL_",
  ↪  source, "_SAMPLES_Centroids_corrected_EPSG32739.shp"),
           driver = "ESRI Shapefile", append = FALSE)

  # Save centroid coordinates as CSV
  FILE3 = paste0(save_plots_path, "/Quadrats_", District, "_", Island, "_ALL_", source,
↪  "_SAMPLES_Centroids_corrected_EPSG32739.csv")
  write.table(pts.in.polys, file=FILE3, sep=";", dec=".", row.names=FALSE)
} # end of source loop
```

This script enables the spatial representation of relocated sample plots and prepares geospatial layers suitable for GIS analysis or habitat modeling.