14/08/2025

QGIS AND R GUIDE FOR NOCTURNAL ROOST ANALYSIS IN FALCO NAUMANNI

SARA SAHILI TOUNTI

MSc Thesis Erasmus Mundus Joint Master Degree in Tropical Biodiversity and Ecosystems,
Université Libre de Bruxelles – ULB (Belgium), Vrije Universiteit Brussel – VUB (Belgium)

TABLE OF CONTENTS

1. Identification and mapping of communal roosts

1.1. Starting files

The initial dataset comprises records from multiple monitoring campaigns of the lesser kestrel conducted in different countries. Each record includes GPS coordinates, individual identifiers, and observation timestamps. Each subfolder comprises three Excel files, structured as follows:

1.2. Original file

For each campaign, the initial file contains all recorded data, including GPS coordinates, date, observation time, and individual identifiers, and represents the unmodified original dataset. An example file name is: "(EBD) Lesser Kestrel (Falco naumanni) Senegal, MERCURIO-SUMHAL.xlsx".

1.3. Hours_roost_campaignname

From the original dataset, only records corresponding to the nocturnal period (18:00 to 06:00), when kestrels typically rest, were filtered. The resulting file was saved as: *hours_roost_campaignname.xlsx*.

- **Time filtering in R if the column is named timestamp:**

filter(hour(timestamp) >= 18 | hour(timestamp) <= 6)

1.4. Generation of nocturnal centroids

To simplify the data and summarise relevant information for each night, nocturnal centroids were calculated. Each centroid represents the average location of an individual during the night, defined as the median of GPS coordinates recorded between 18:00 and 06:00 hours. This approach aims to reduce the number of points to analyse while maintaining spatial and temporal accuracy, thereby facilitating the identification of patterns in the use of communal roosts.

- Create a new column named "date_id" by combining the date (with the time component removed) and the individual identifier. This can be achieved using the following formula:

=TEXT(cell_date; "dd-mm-yyyy") & "_" & cell_individual

- Generate a list of unique values from the "date_individual" column by copying its contents and removing duplicates, ensuring that each night-individual combination appears only once.

- Create a new column named “date_individual_unique” to store these unique night-individual combinations.
- For each unique value in the “date_individual_unique” column, calculate the average longitude using the following formula:

=IFERROR(AVERAGEIFS([range of "date_individual"], [unique value of "date_individual_unique"], [range of "longitude"]), "")

- Similarly, calculate the average latitude coordinates with:

=IFERROR(AVERAGEIFS([range of "date_individual"], [unique value of "date_individual_unique"], [range of "latitude"]), "")

- This process yields a nocturnal centroid representing the average position of an individual during each night.
- Finally, save the resulting file using a concise name that reflects the campaign, such as the marking country.
- To extract the date and individual names from the unique dates only:
- Individuals:

=MID(unique_date_ind, FIND("_", unique_date_ind) + 1, LEN(unique_date_ind))

- Date:

=LEFT(unique_date_ind, FIND("_", unique_date_ind) - 1)

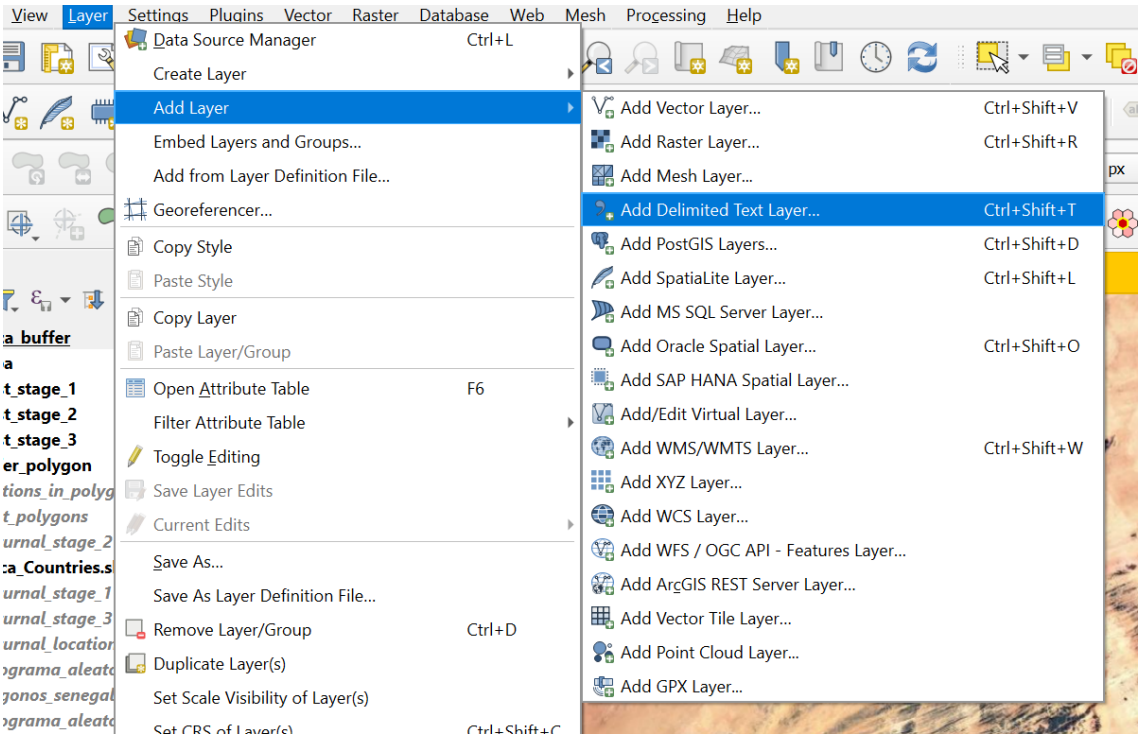
Note: All Excel files must use consistent column names for individual identifiers, longitude coordinates, latitude coordinates, and date.

1.5. Load the data into QGIS

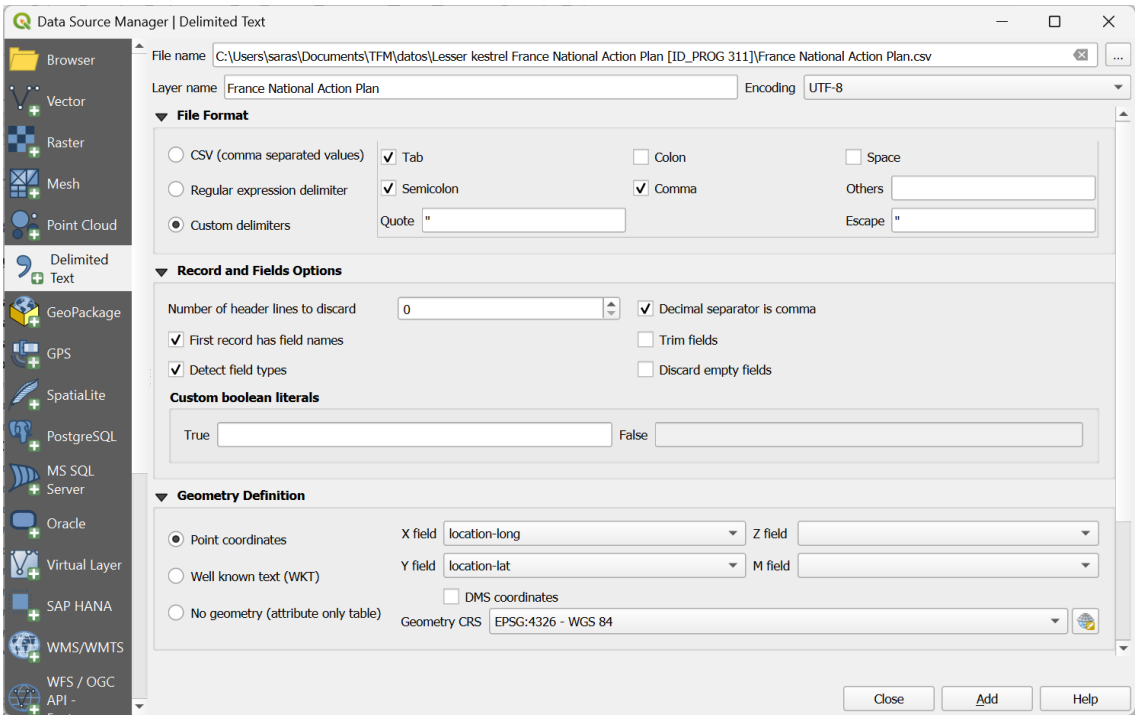
1.5.1. Load CSV file:

Load the CSV file into QGIS by adding it as a comma-delimited text layer.

- Add the layer using the option Add Layer > Add Delimited Text Layer.



- Customs delimiters were chosen.
- In the geometry definition, the X field is selected as longitude and the Y field as latitude.



1.6. Export to shapefile

Export the loaded CSV file in QGIS and save it as a shapefile to enable attribute layer editing. Save the file using the same name.

1.7. Filtering Non-Relevant Points

1.7.1. Exclude points:

Conduct a visual and spatial inspection to identify and exclude points located outside the wintering area (Sahel).

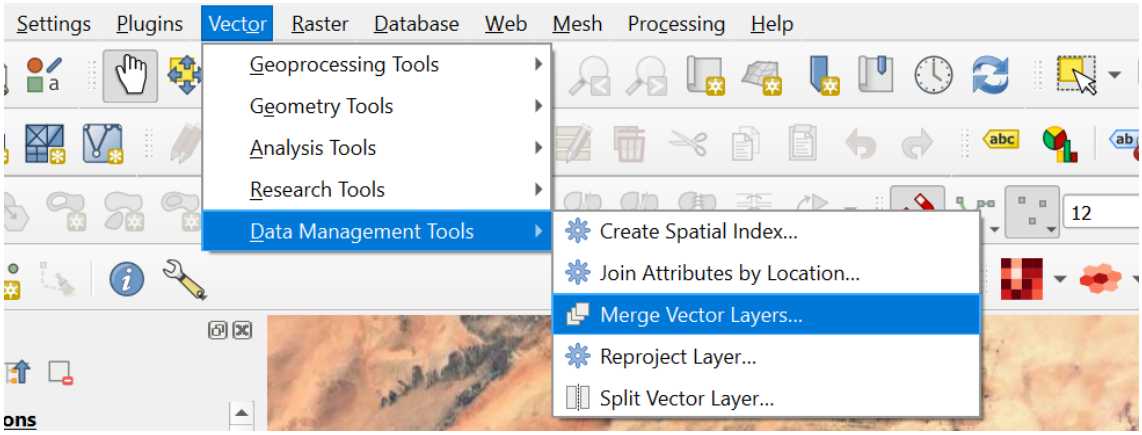
1.7.2. Define points:

Define migration points by identifying clear continuity or linearity within GPS tracking data.

1.8. Merge shapefile layers

1.8.1. Combine shapefile layers:

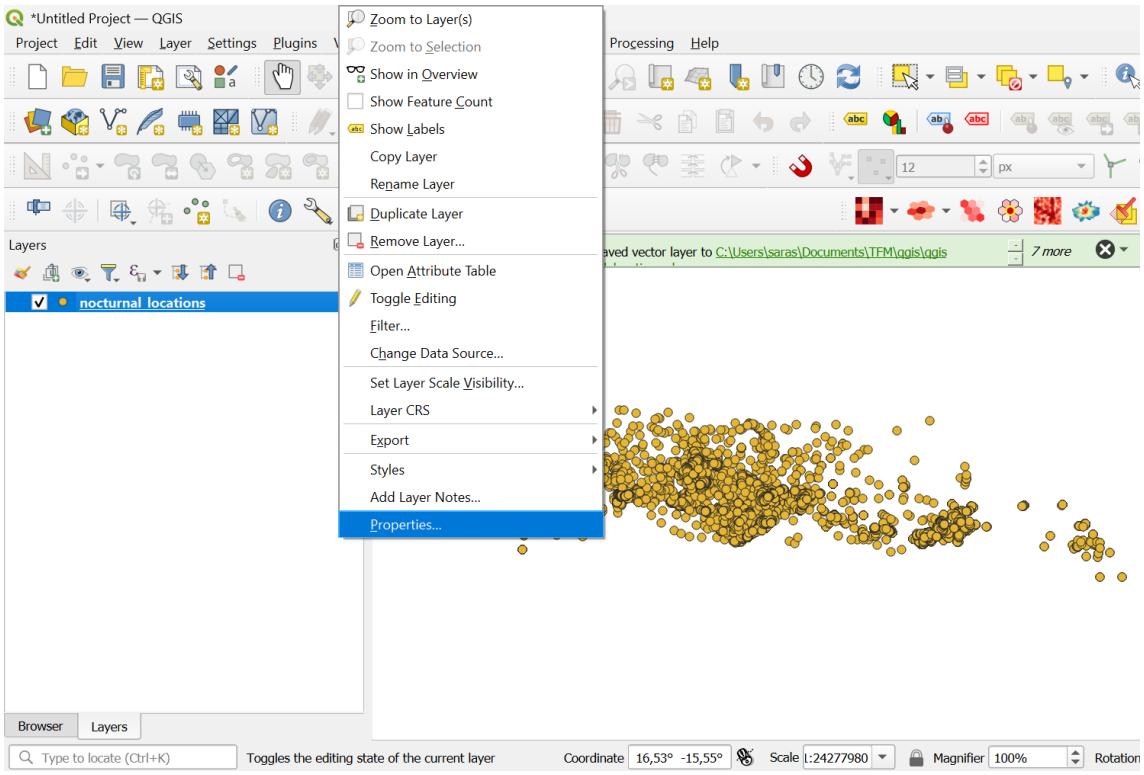
Upon completion of the preceding steps for all campaigns, merge the shapefile layers from each campaign using the *Merge Vector Layers* tool. Save the resulting merged layer as “nocturnal_locations”.



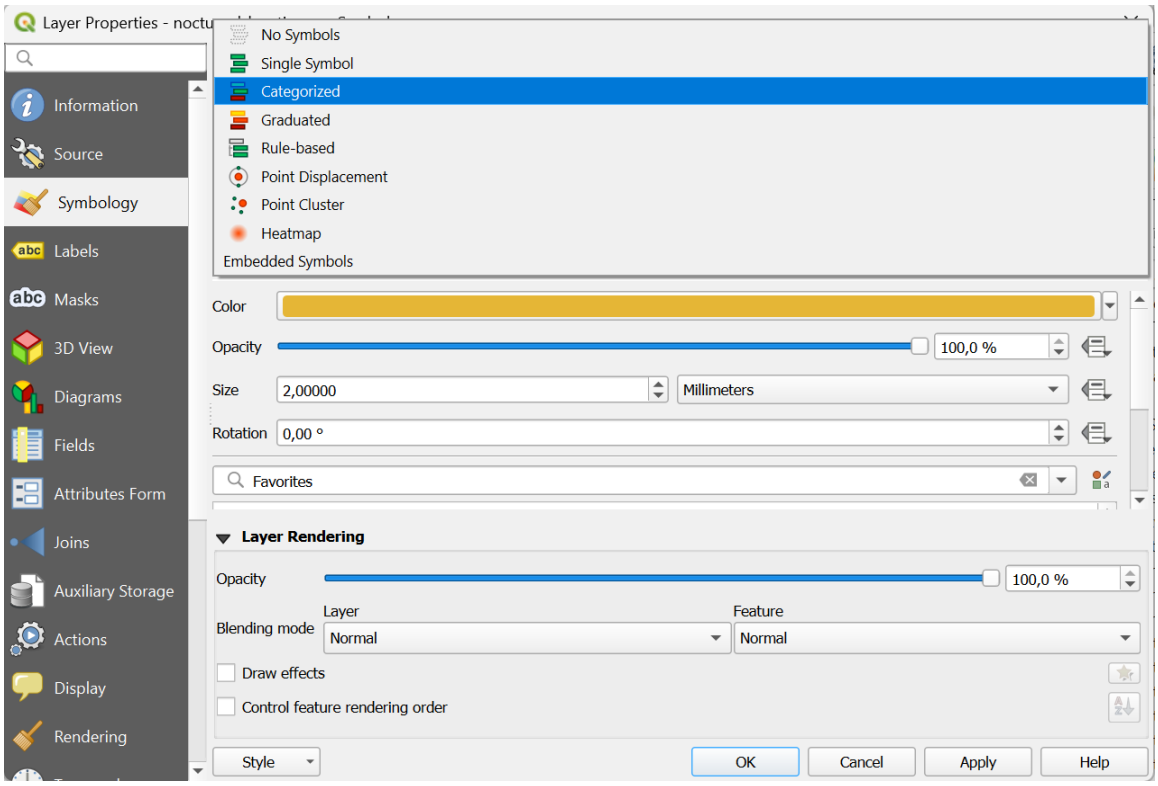
1.9. Modify symbology

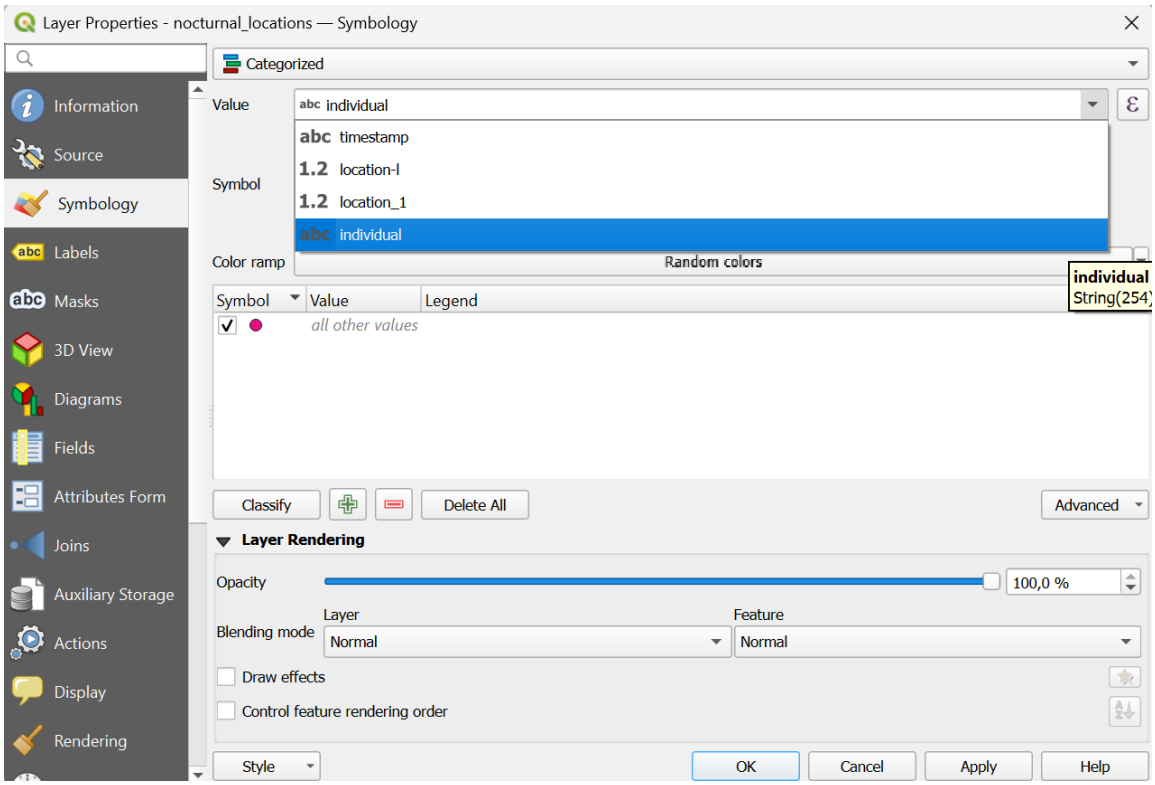
1.9.1. Adjust the symbology of the shapefile layers to visually distinguish the different individuals by:

- Selecting the properties of the “nocturnal_locations” layer.

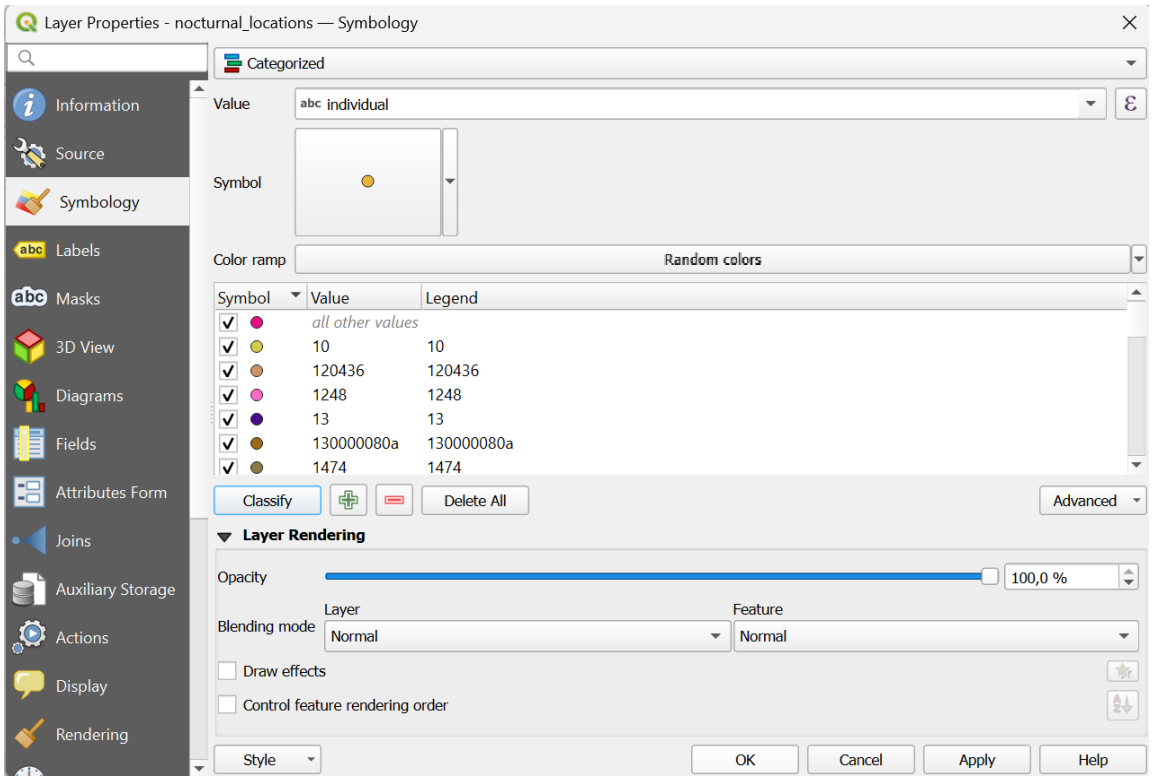


- Select “Symbology” in the layer properties of the shapefile layers.
- Choose the “Categorised” option and set “individual” as the value.





- Click the “Classify” button and confirm.



1.10. Kernel density calculation

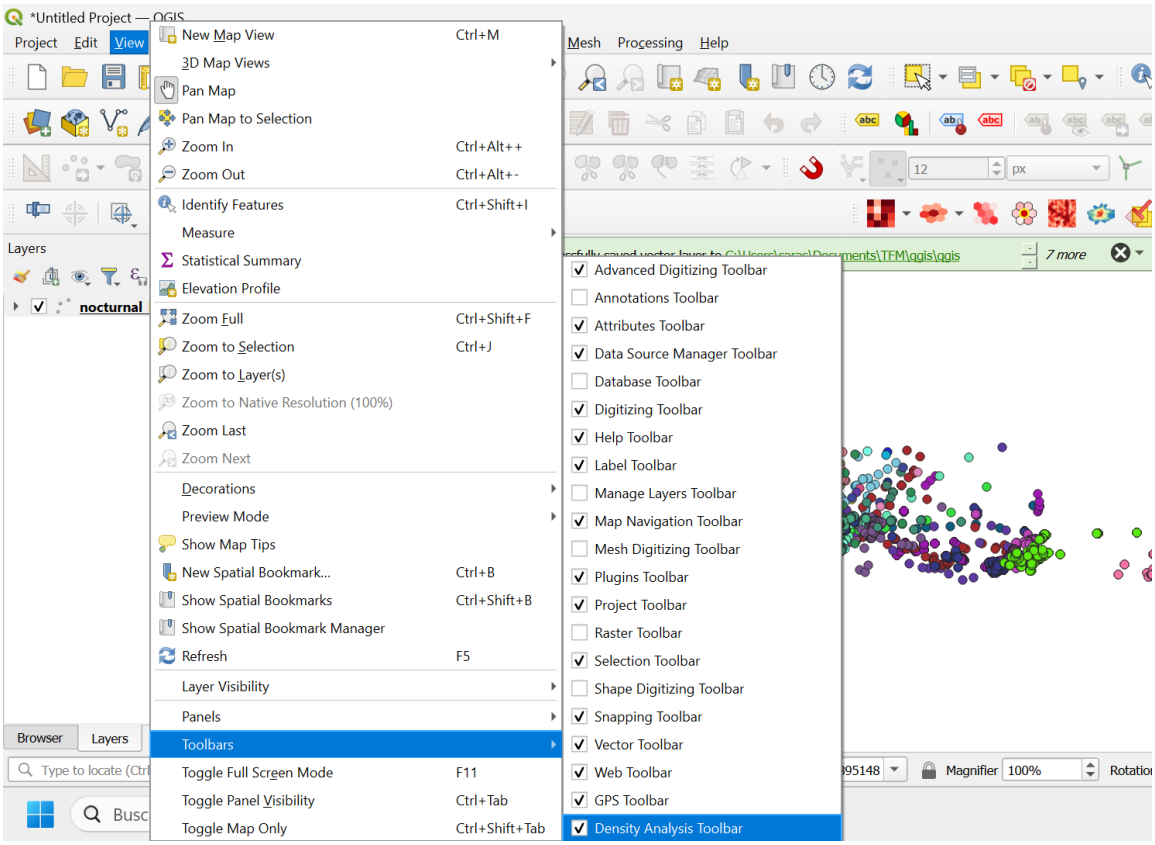
1.10.1. Install the density plugin:

- Open “Manage and Install Plugins” in QGIS, search for “Density Analysis”, and install the plugin.

1.10.2. Show the toolbar:

If the plugin toolbar is not visible, it can be enabled via the following application's settings menu:

View > Toolbars > select: Density Analysis Toolbar

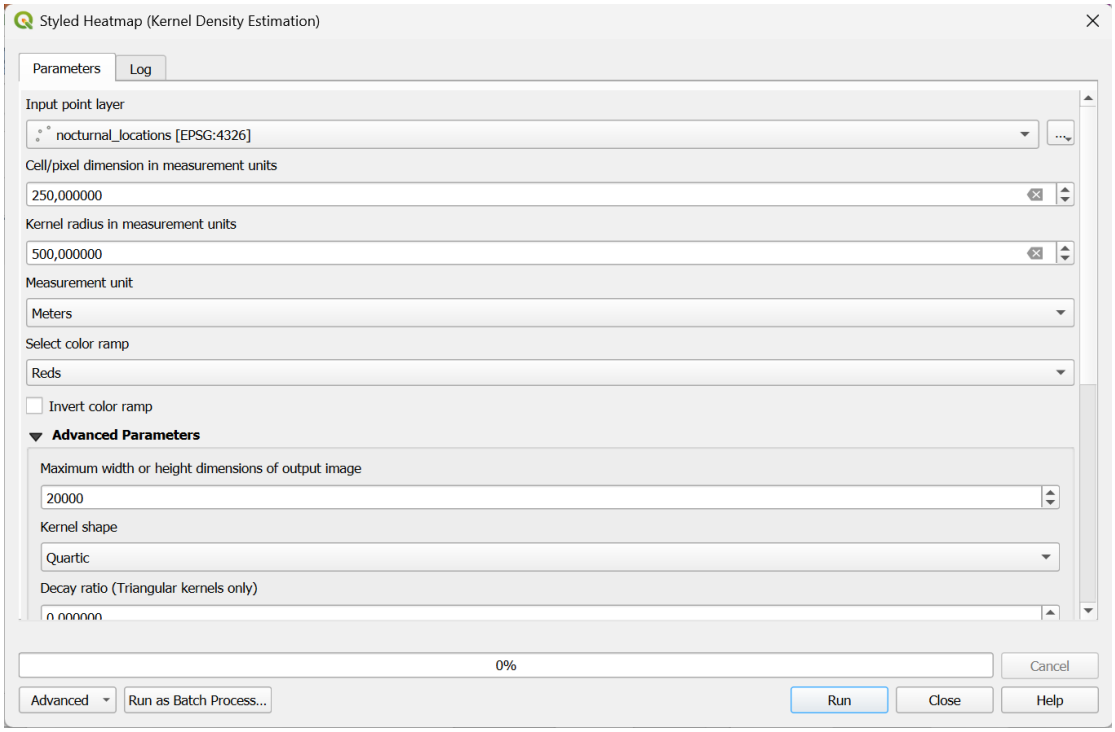


1.10.3. Generate the heatmap:

Use the tool “Styled Heatmap (Kernel Density Estimation)” and configure the parameters as follows:

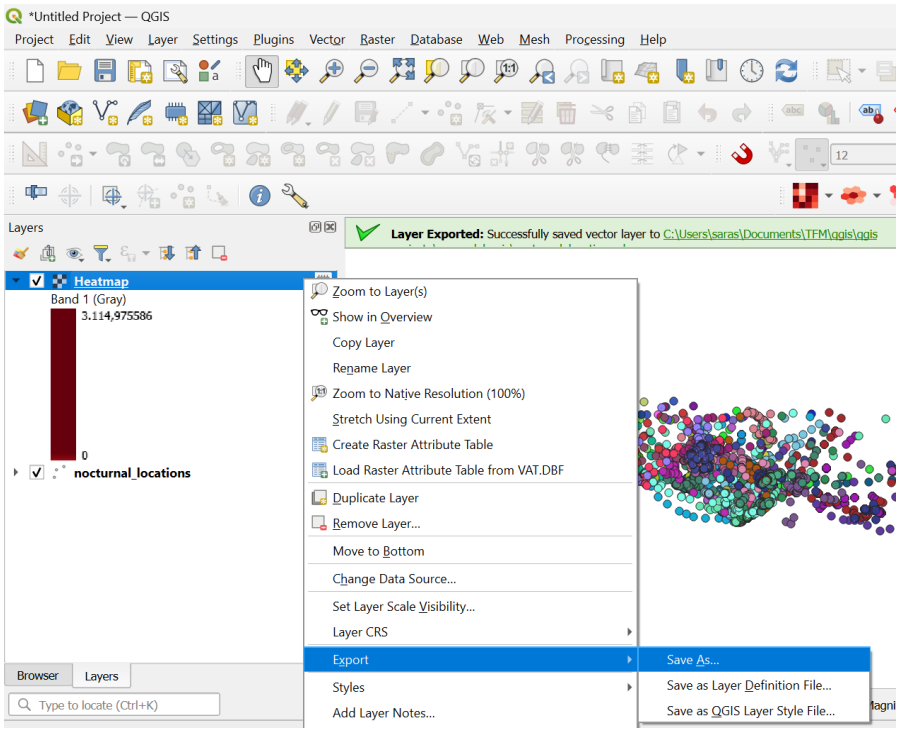
- *Input point layer*: nocturnal_locations
- *Cell/pixel* size: 250 m.
- *Kernel radius:* 500 m.
- *Measurement unit*: meters.

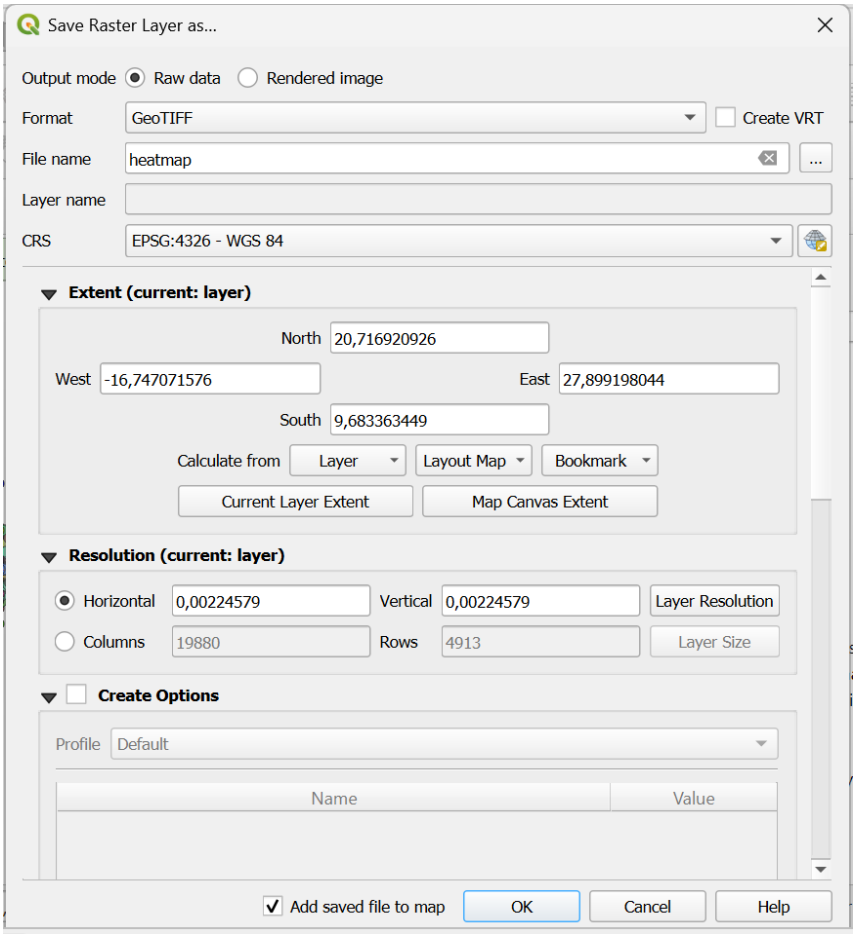
- Run the analysis.



1.10.4. Export the heatmap:

Export the heatmap generated by Kernel Density and save it as a GeoTIFF file named "heatmap".



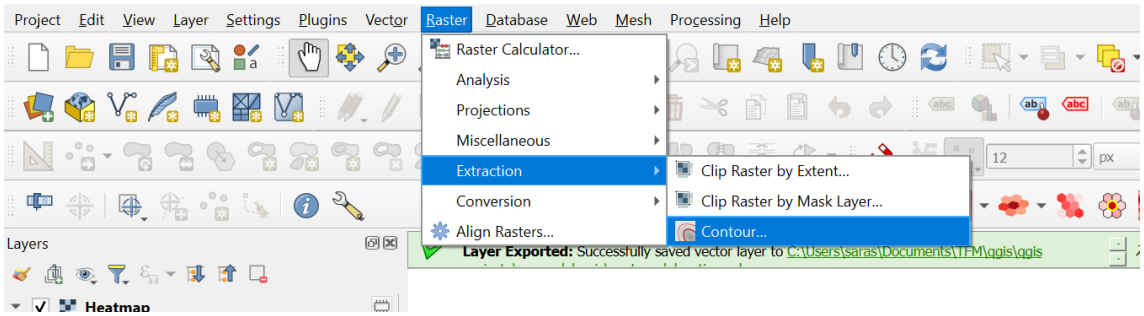

Save Raster Layer as...
Output mode
Raw data
Rendered image
Format
GeoTIFF
Create VRT
File name
heatmap
Layer name
CRS
EPSG:4326 - WGS 84
Extent (current: layer)
North
20,716920926
West
-16,747071576
East
27,899198044
South
9,683363449
Calculate from
Layer
Layout Map
Bookmark
Current Layer Extent
Map Canvas Extent
Resolution (current: layer)
Horizontal
0,00224579
Vertical
0,00224579
Layer Resolution
Columns
19880
Rows
4913
Layer Size
Create Options
Profile
Default
Name
Value
Add saved file to map
OK
Cancel
Help

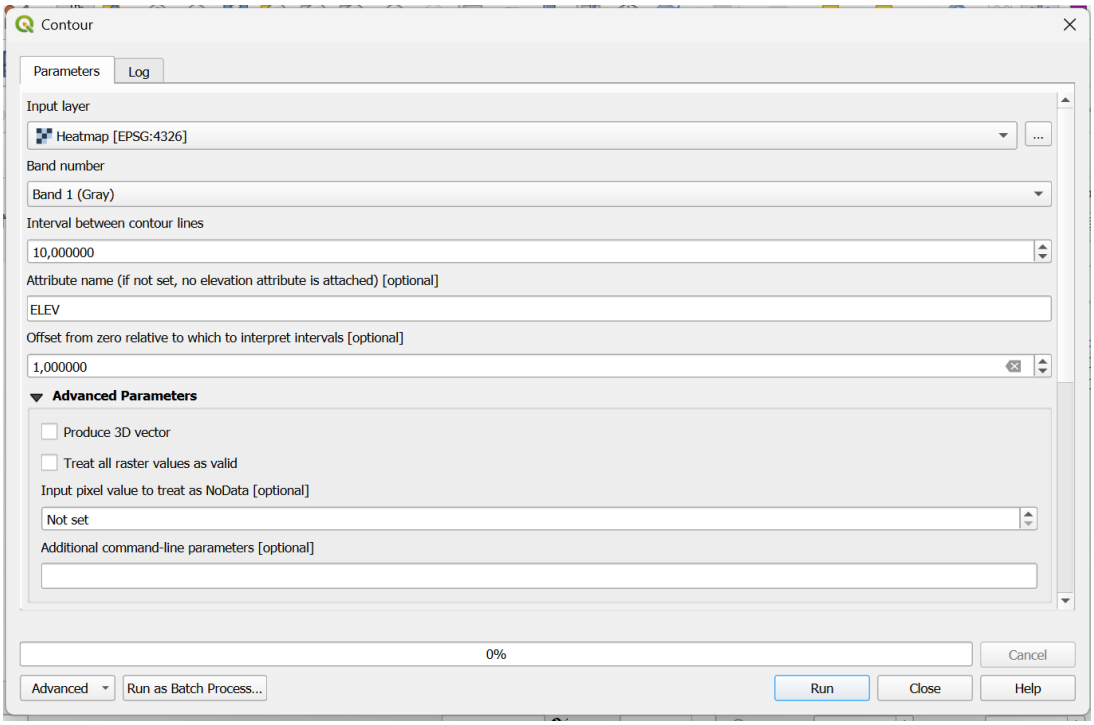
1.11. Create contour lines

1.11.1. Generate contour lines:

Use the “Contour” tool in QGIS with the following settings:

- Input layer: “heatmap”.
- Interval between contour lines: 10
- Attribute name: ELEV
- Offset from 0: 1
- Save the resulting layer as a shapefile named “contour_lines”.



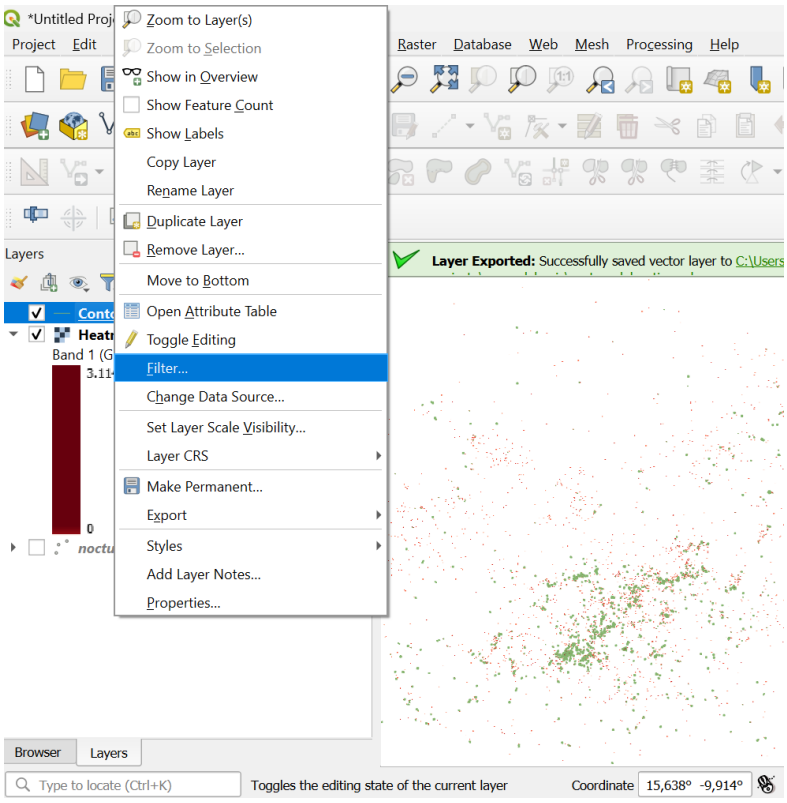


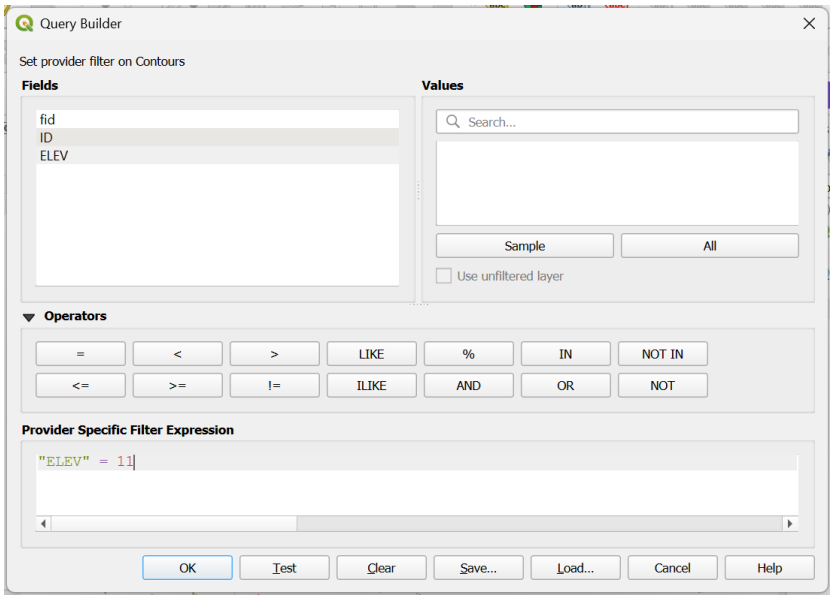
1.12. Create polygons

1.12.1. Filter contour lines:

Filter the contour lines using the ELEV attribute:

- Right-click the layer and select Filter.
- Use the expression: ELEV = 11.
- Save the filtered layer as a shapefile named contour_lines_filter10.

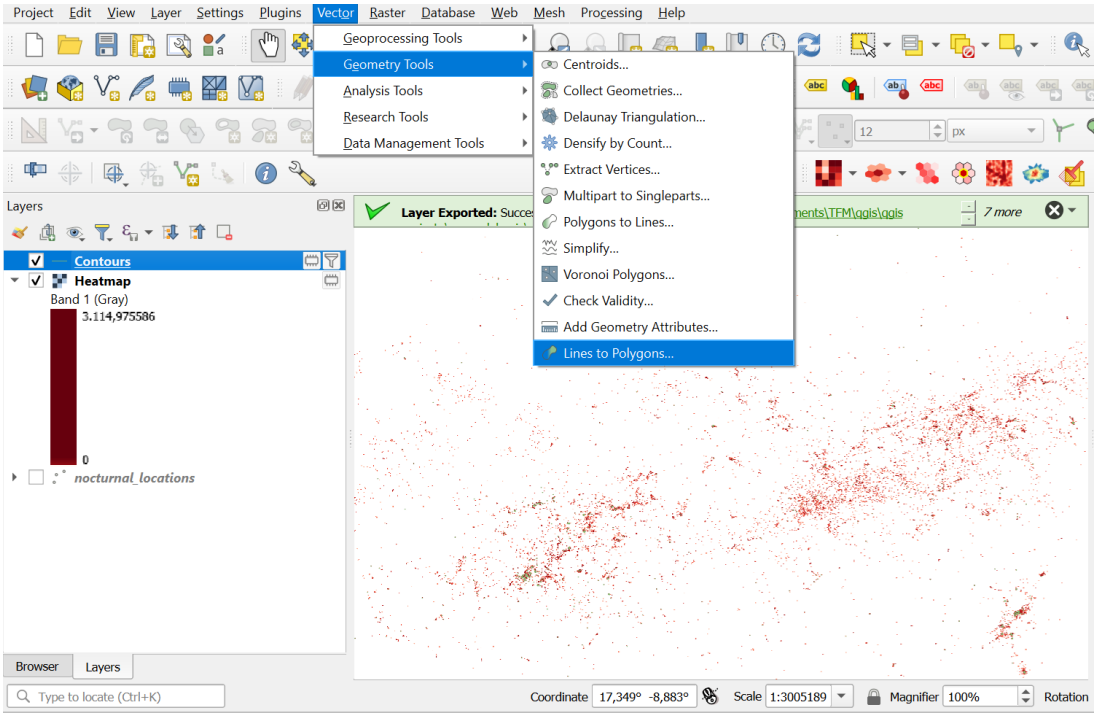




1.12.2. Convert contour lines to polygons:

Use the “Lines to Polygons” tool in QGIS:

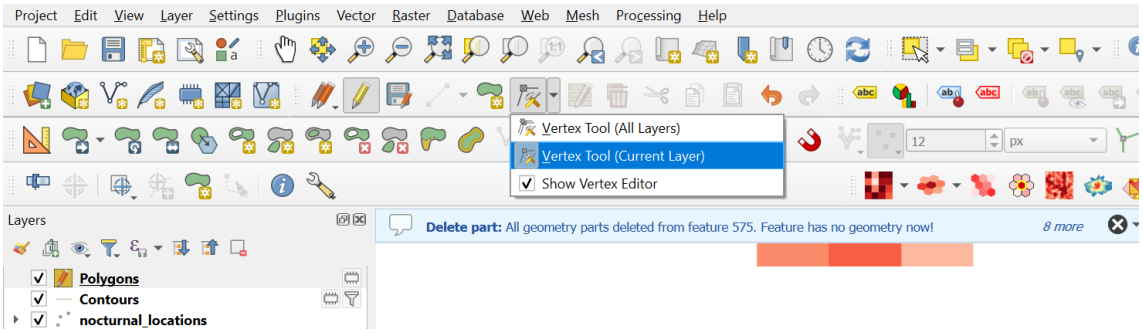
- Input layer: contour_lines_filter10
- Run the process and save the resulting polygon layer as a shapefile named roost_polygons.



1.13. Manually review and edit the roost polygons layer (roost_polygons)

This optional step allows refinement of automatically generated polygons. Closely spaced polygons may represent a single communal roost and should be merged, whereas some features may be missed during automatic processing.

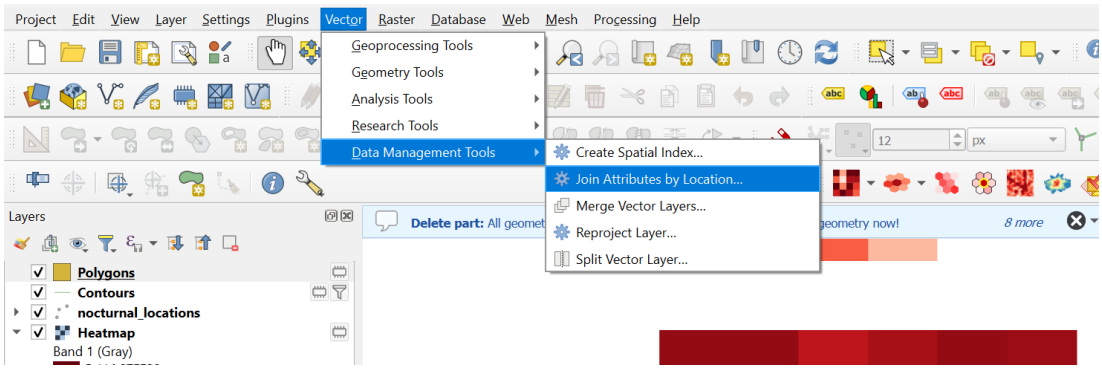
To edit the layer, open roost_polygons in QGIS and adjust or merge polygons using the Vertex Tool as necessary.



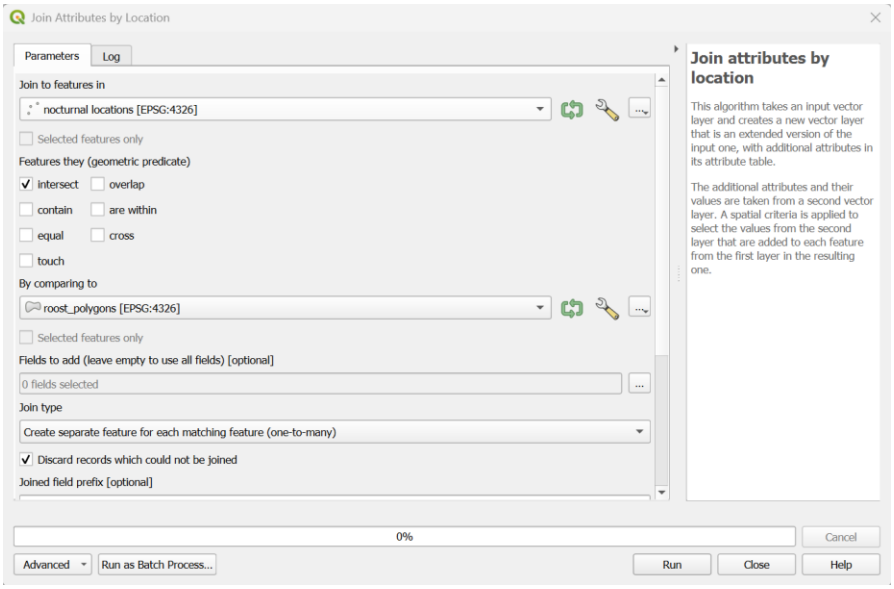
1.14. Select points within polygons

To evaluate the number of individuals using each roost per stage, individual nocturnal location points of Lesser Kestrels were spatially joined with polygons generated through Kernel Density Estimation (KDE).

This procedure was carried out using the *Join Attributes by Location* tool, following these steps:

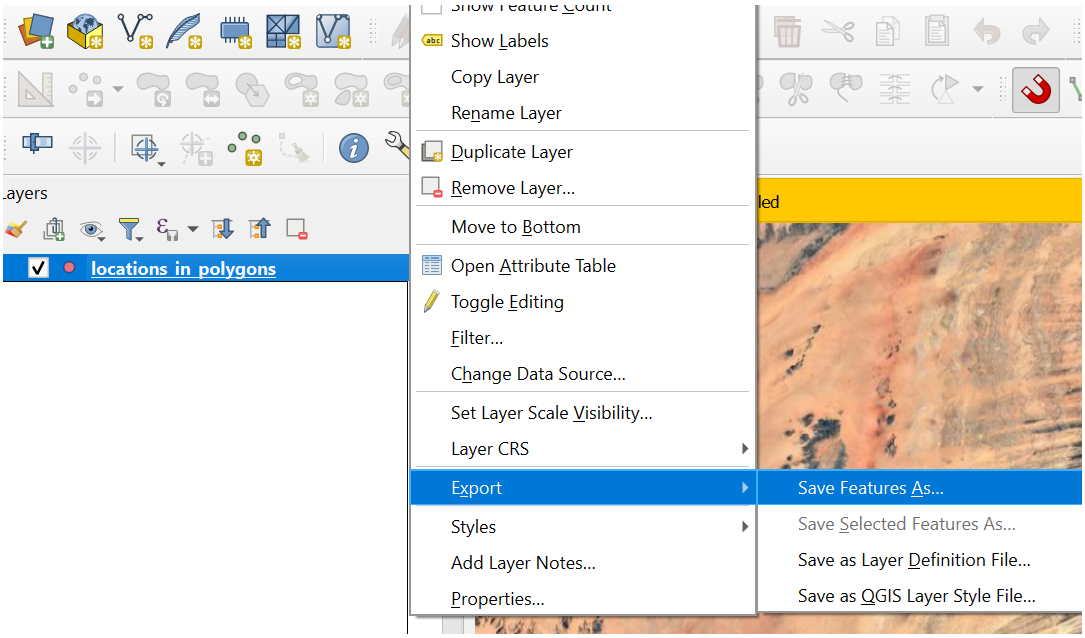


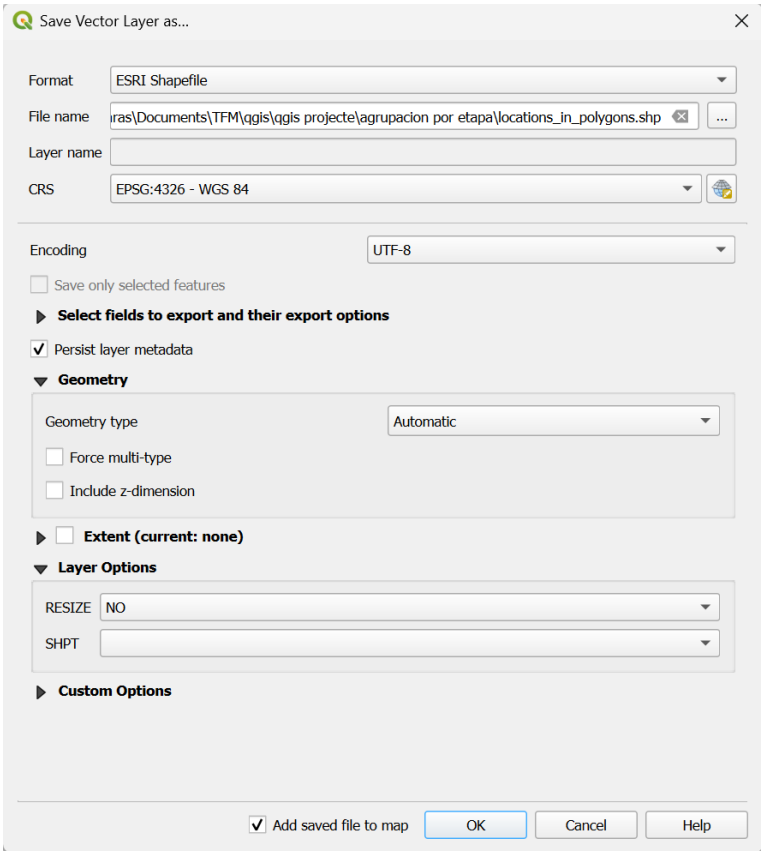
- The input layer consists of the point layer representing individual nocturnal locations, labelled “nocturnal_locations.”
- The join layer consists of the polygon layer of roosts obtained through KDE, labelled “roost_polygons.”
- The “intersects” spatial predicate was used to join only points within the polygons. Enabling “Discard records which could not be joined” is important, as it ensures that all points outside the defined roost polygons are excluded.



1.14.1. Save the layer

The output was saved as a new layer named “locations_in_polygons,” containing points classified by roost polygon and stored in shapefile format. This layer will be used in subsequent analyses throughout the study.





2. Temporal aggregation

2.1.Select points within polygons

This step employs a dataset integrating nocturnal GPS locations of Lesser Kestrels with roost polygons generated through Kernel Density Estimation (KDE), enabling analysis of individual use of communal roosts across distinct wintering stages.

- The input dataset, named “locations_in_polygons”, comprises GPS points assigned to the corresponding roost polygon in which they are spatially located. This dataset results from a spatial join operation conducted in QGIS, as detailed in point 1.14.

2.2. Data preparation and loading in R

2.2.1. The following packages were loaded:

library(readxl, library(dplyr), library(lubridate)

2.2.2. Process the dates and assign each record to a wintering stage.

Each GPS point was classified into one of the following wintering stages:

Stage 1: from September 20 to October 19

Stage 2: from October 20 to November 19

Stage 3: from November 20 to April 30

```
locations_in_polygons <- locations_in_polygons %>%
  mutate(
    date = dmy(timestamp),
    month = month(date),
    day = day(date),
    stage = case_when(
      (month == 9 & day >= 20) | (month == 10 & day <= 19) ~ "Stage 1",
      (month == 10 & day >= 20) | (month == 11 & day <= 19) ~ "Stage 2",
      (month == 11 & day >= 20) | month %in% c(12, 1, 2, 3, 4) ~ "Stage 3",
      TRUE ~ NA_character_
    )
  ) %>%
  filter(!is.na(stage), !is.na(ID))
```

2.3.Calculate the number of distinct individuals per roost and wintering stage.

This step quantifies the number of individual Lesser Kestrels utilising each communal roost during the distinct stages of the wintering period.

```
summary <- locations_in_polygons %>%
  group_by(stage, ID) %>%
  summarise(
    n_individuals = n_distinct(individual),
    .groups = "drop"
  ) %>%
  mutate(
    aggregated = ifelse(n_individuals >= 2, 1, 0)
  )
```

2.4. Calculate the proportion of aggregated roosts (used by ≥2 individuals) per wintering stage.

This step evaluates the proportion of roosts shared by two or more individuals during each stage of the wintering period, which may reveal patterns of communal roosting behaviour.

```
proportions <- summary %>%
  group_by(stage) %>%
  summarise(
    total_roosts = n(),
    aggregated_roosts = sum(aggregated),
    proportion_aggregated = aggregated_roosts / total_roosts
  )

print(proportions)
```

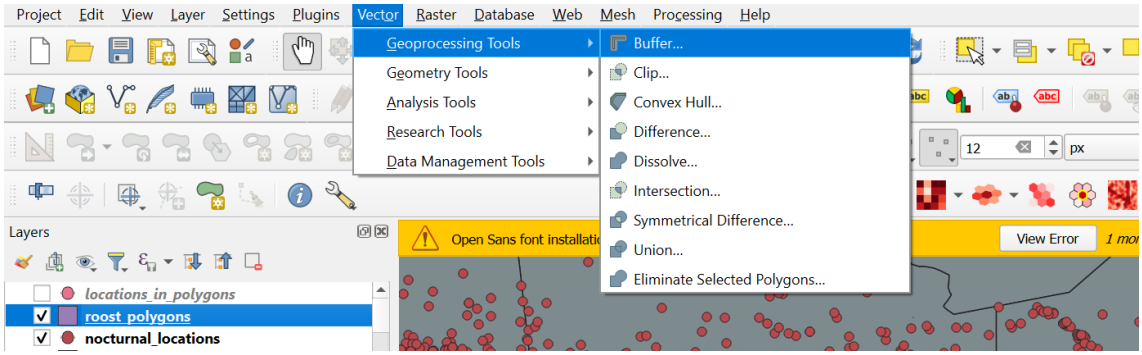
3. Habitat composition and selection at roost sites

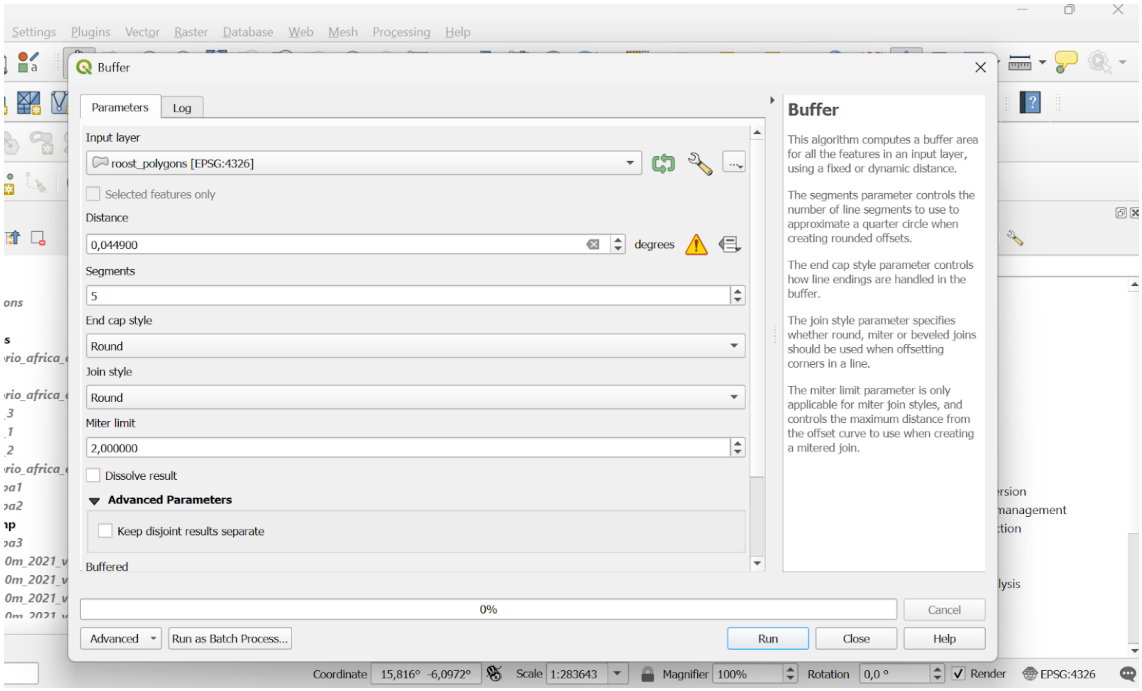
3.1. Creation and Processing of Roost Buffers

3.1.1. Create a 5 km buffer around each roost polygon.

A 5 km buffer was created for each communal roost area using the “roost_polygons” layer.

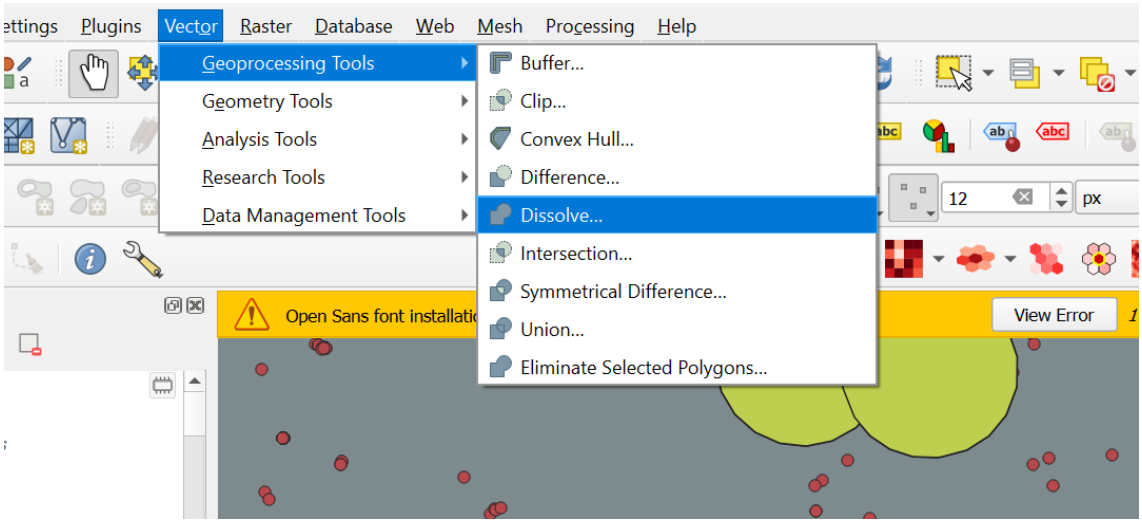
- In QGIS, go to: Vector > Geoprocessing Tools > Buffer**.**
- Select the input layer “roost_polygons” and specify a buffer distance of 0.0449 degrees, which corresponds approximately to 5 km at the latitude of the Sahel.
- Run the process to create the buffer layer.

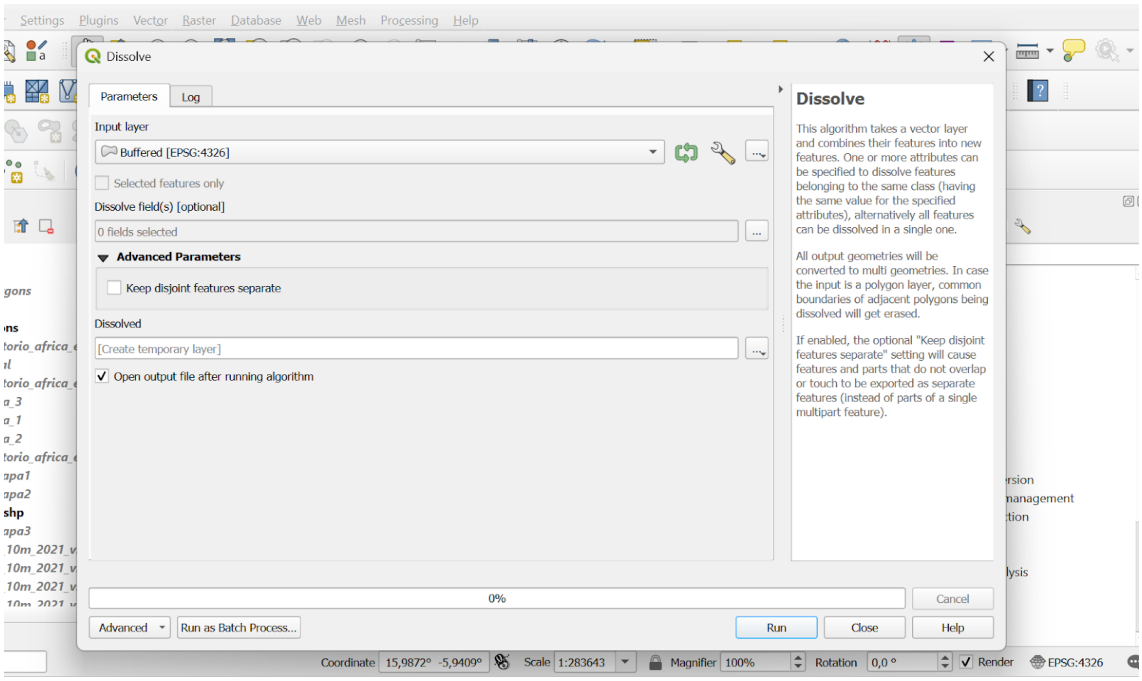




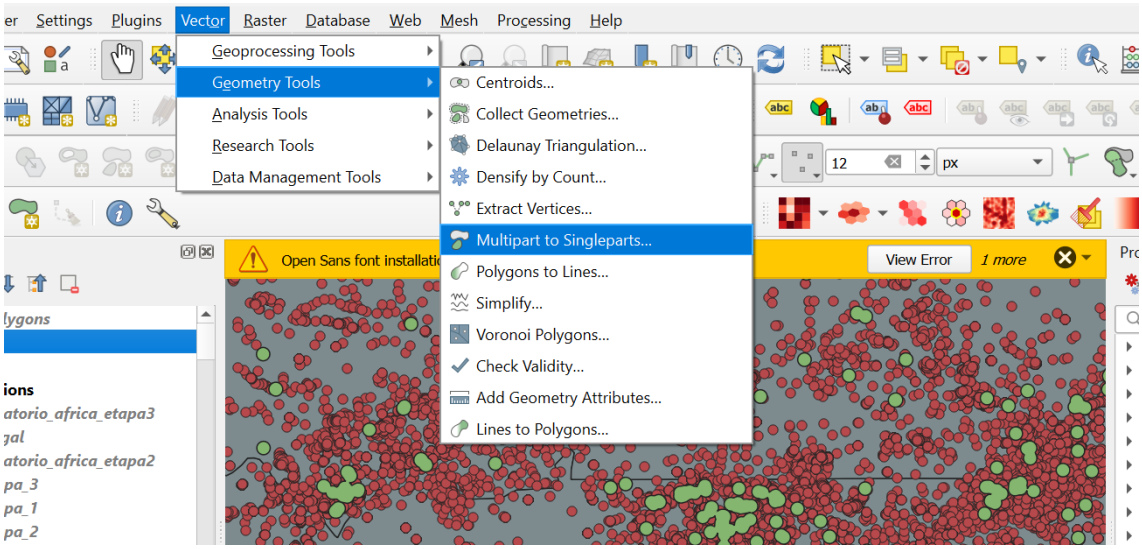
3.1.2. Dissolution of buffer overlapping

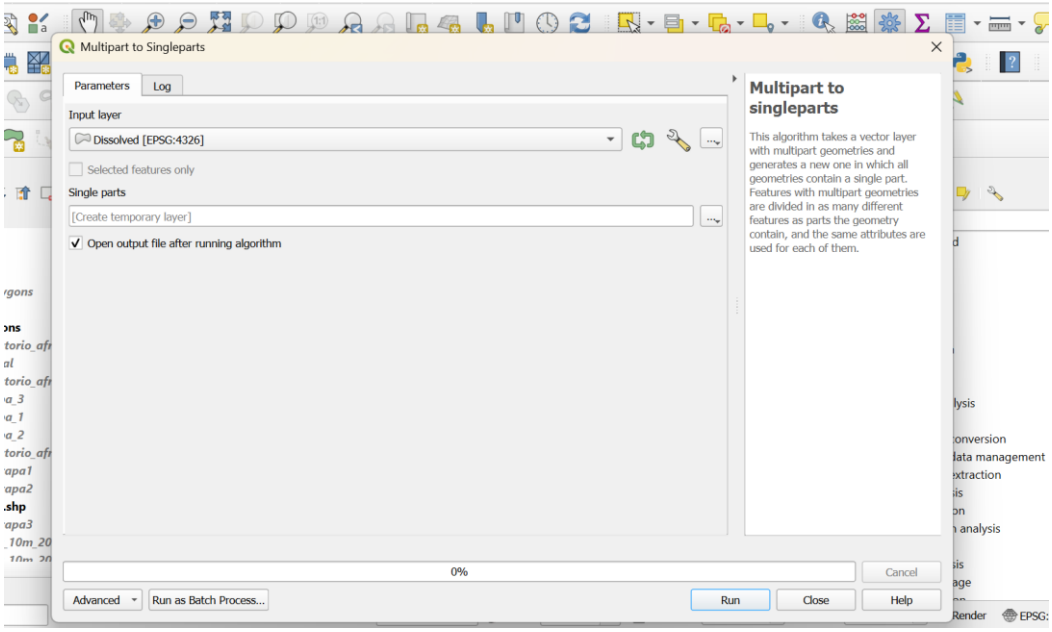
- Use the Dissolve tool (located under Vector > Geoprocessing Tools > Dissolve) without selecting any attribute fields, to merge all overlapping geometries, including those from the buffer layer, into a single unified polygon.



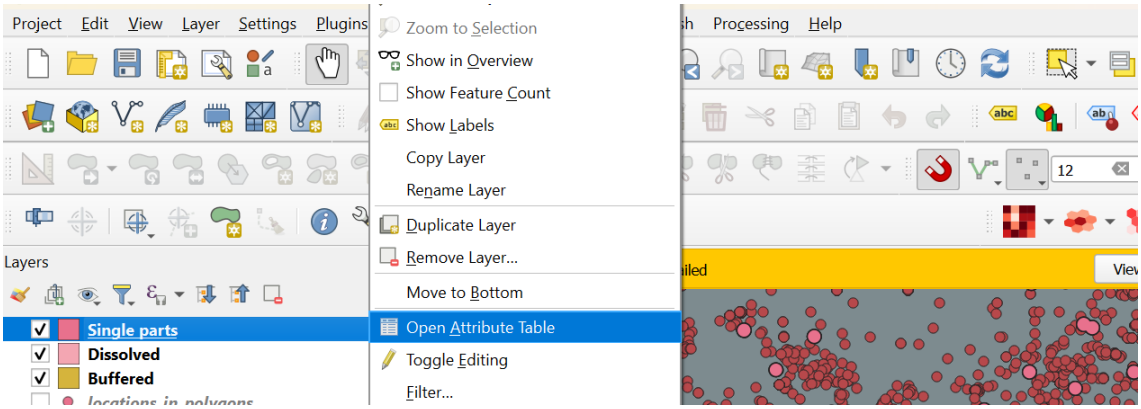


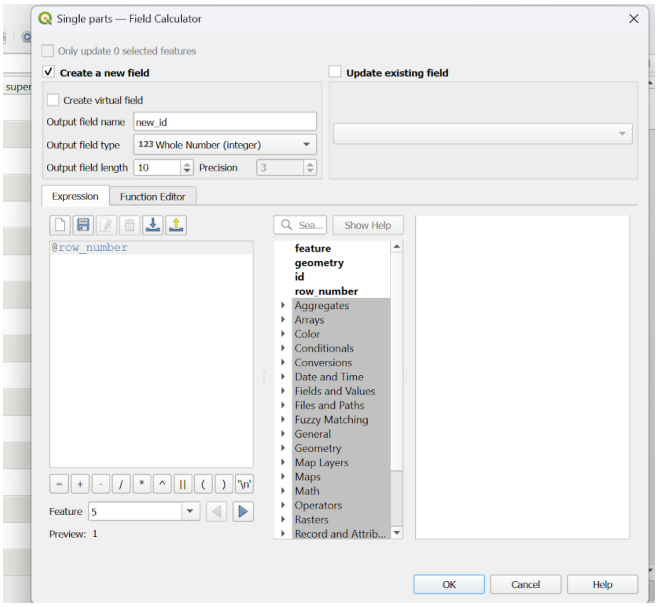
- Use the "Multipart to Singleparts" tool (Vector > Geometry Tools > Multipart to Singleparts) to split the resulting polygon into individual parts.





- Assign a unique ID to each polygon: open the attribute table of the buffer layer and use the Field Calculator to create a new field (e.g., buffer_ID). In the expression box,use @row_number to generate a distinct ID for each polygon. This step facilitates the identification and grouping of overlapping buffers in subsequent analyses.

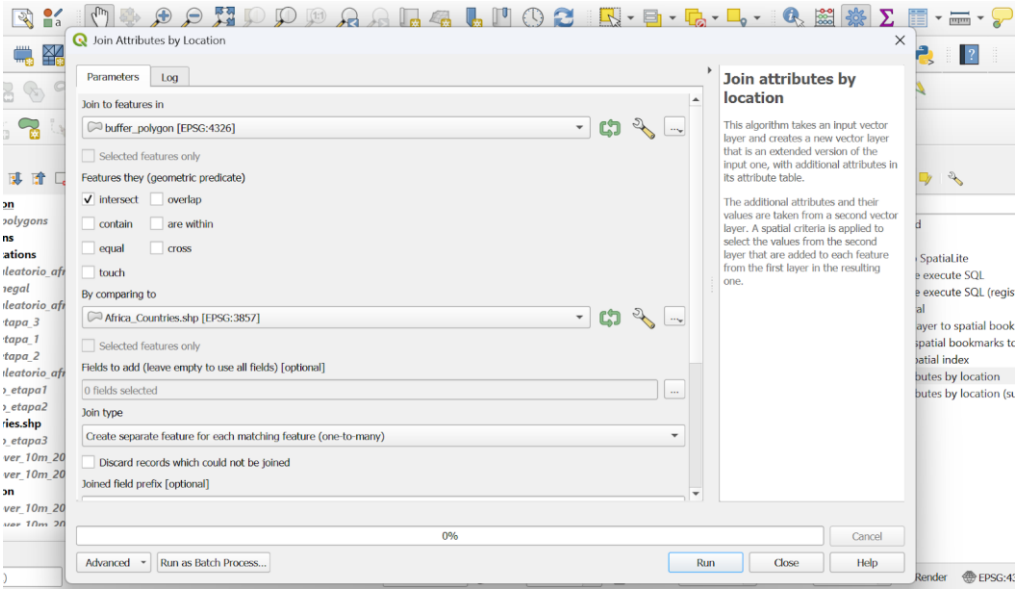




- Save the resulting layer as “buffer_polygon”.

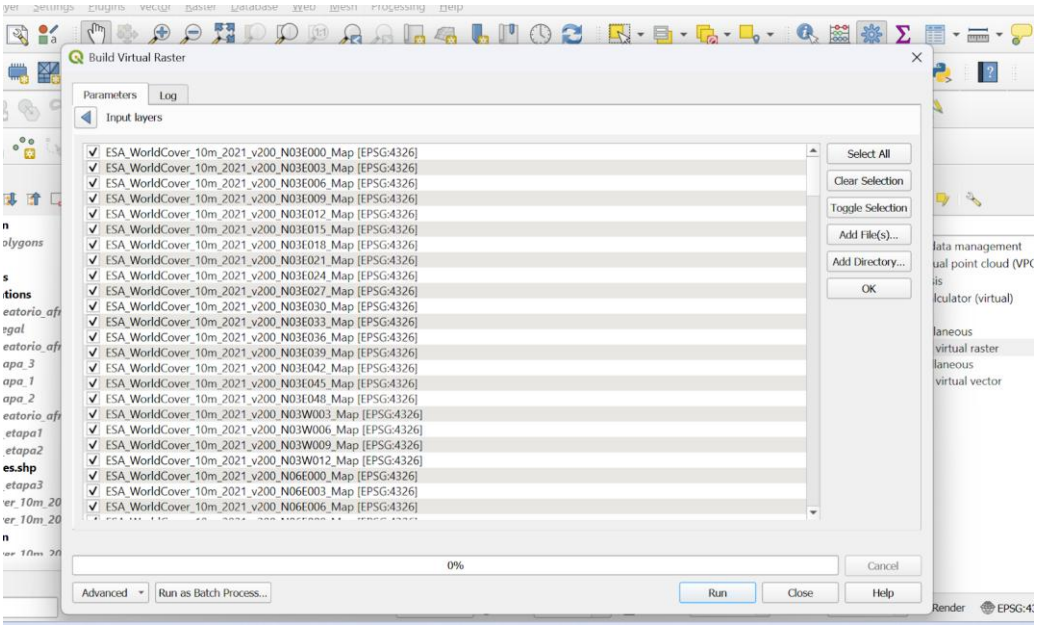
3.2. Union of the buffer layer with the Africa layer

- Use the "Join attributes by location" tool to join the buffer_polygon layer with the countries layer, selecting “intersect” as the geometric predicate. This operation assigns each roost to its corresponding country. Export the resulting layer as “africa_buffer.xlsx” and “africa_buffer.shp”.

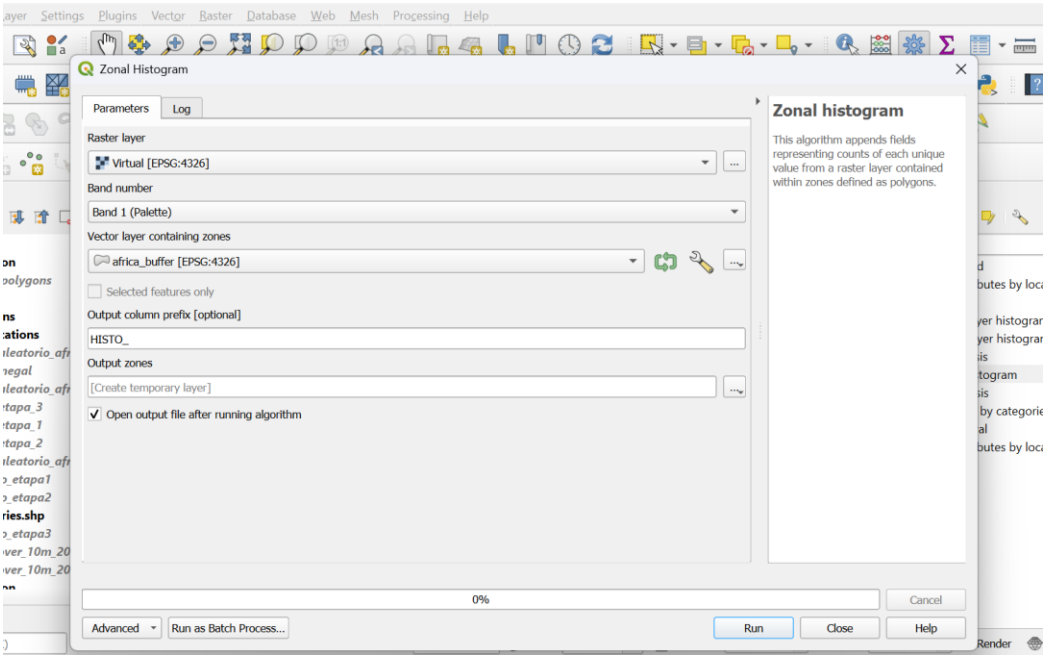


3.3.Data preparation in QGIS

- Create a virtual raster that combines the available coverage rasters.



- From “africa_buffer” (5 km buffers around the roost centroids), use the Zonal Histogram tool to extract the number of pixels of each land cover category from the ESA WorldCover 10 m virtual raster layer within each buffer. Set the Output column prefix to: HISTO_.



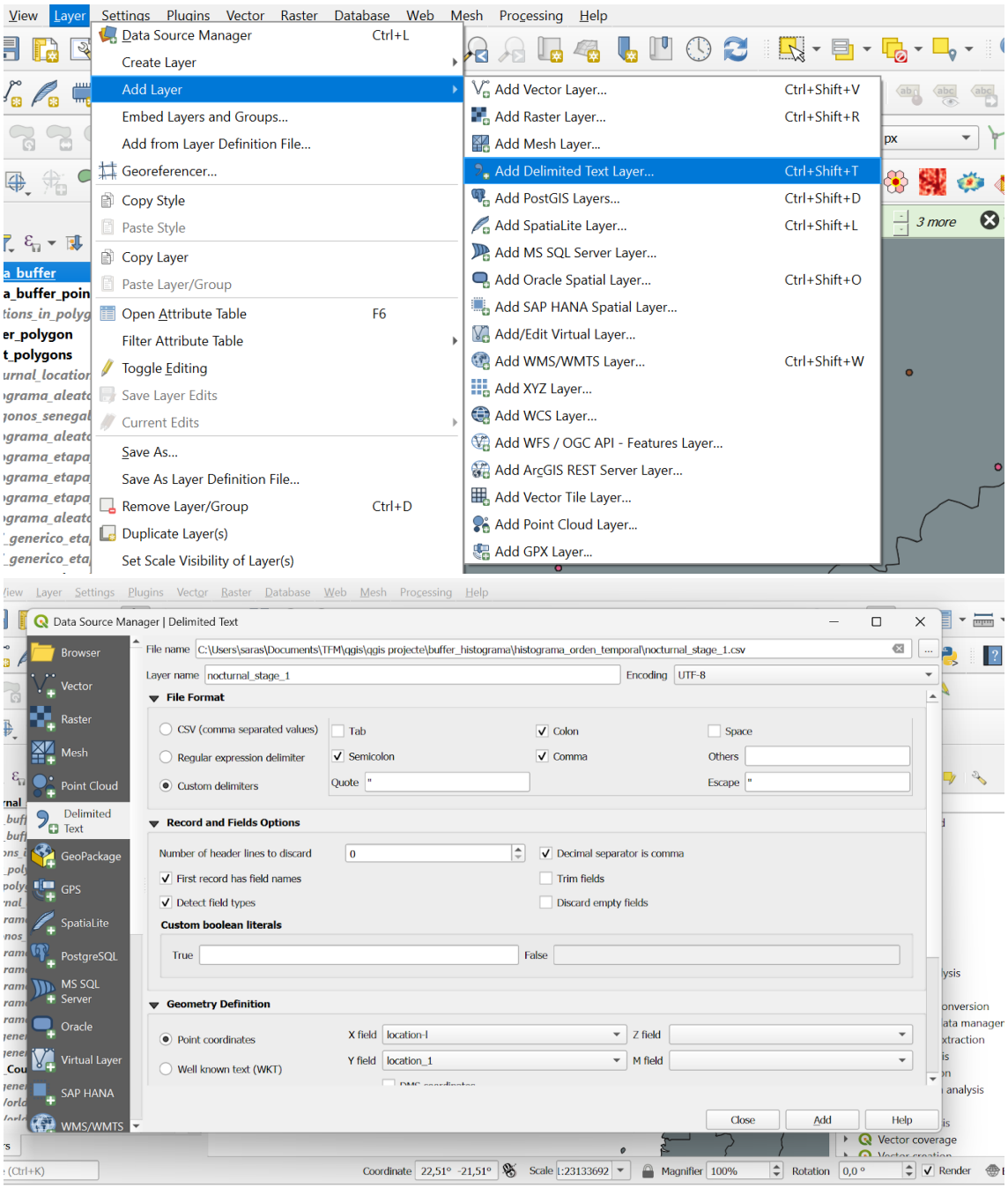
- Save the result as "africa_histo.xlsx".

3.4. Temporal histograms

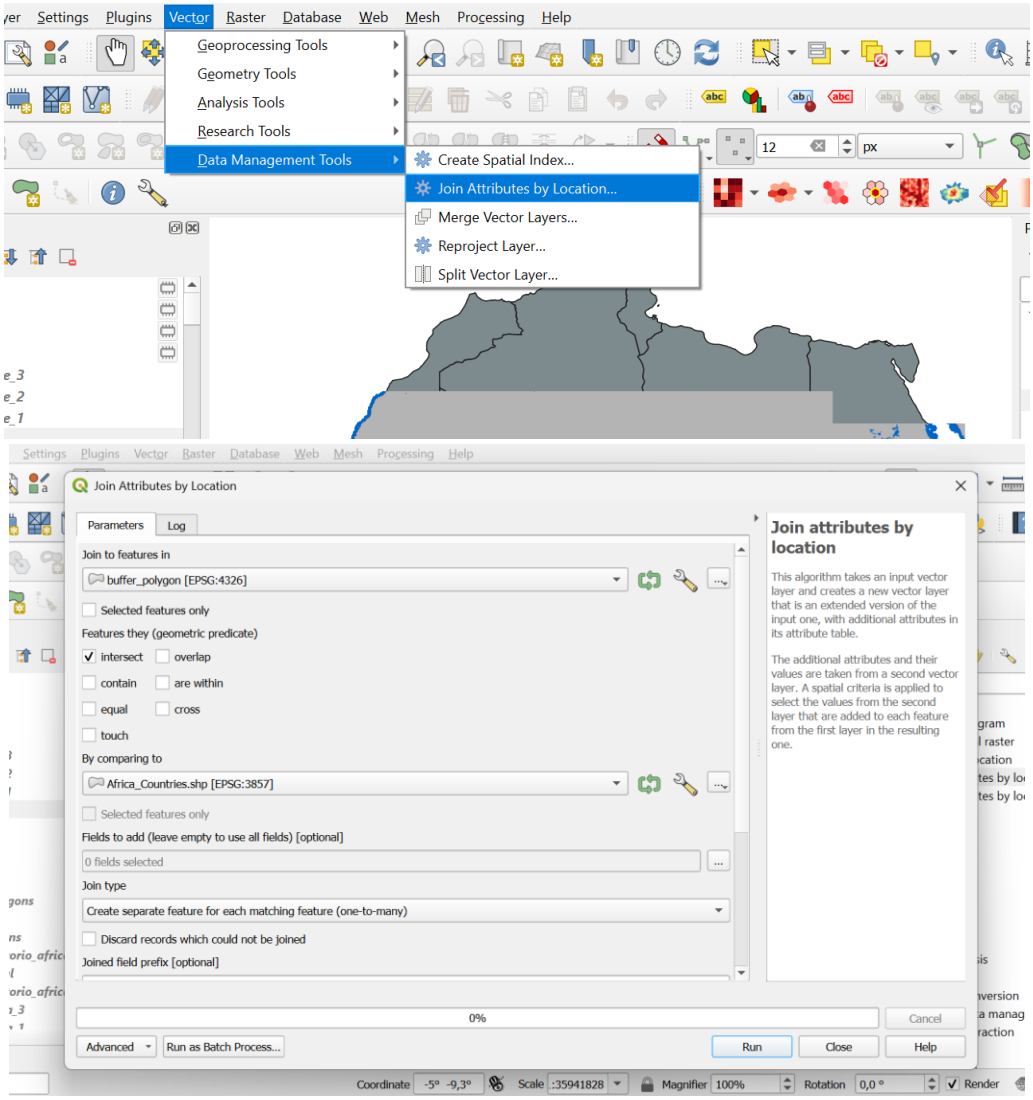
- Split the layer “nocturnal_locations” into 3 stages using R, and save the resulting CSV files as “nocturnal_stage_1”, “nocturnal_stage_2”, and “nocturnal_stage_3”:

```
nocturnal_locations <- nocturnal_locations %>%
  mutate(
    timestamp = as.Date(timestamp, format = "%d-%m-%Y"),
    day = day(timestamp),
    month = month(timestamp),
    stage = case_when(
      (month == 9 & day >= 20) | (month == 10 & day <= 19) ~ "Stage 1",
      (month == 10 & day >= 20) | (month == 11 & day <= 19) ~ "Stage 2",
      (month == 11 & day >= 20) | month %in% c(12, 1, 2, 3) | (month == 4 & day <= 25)
~ "Stage 3",
      TRUE ~ NA_character_
    )
```

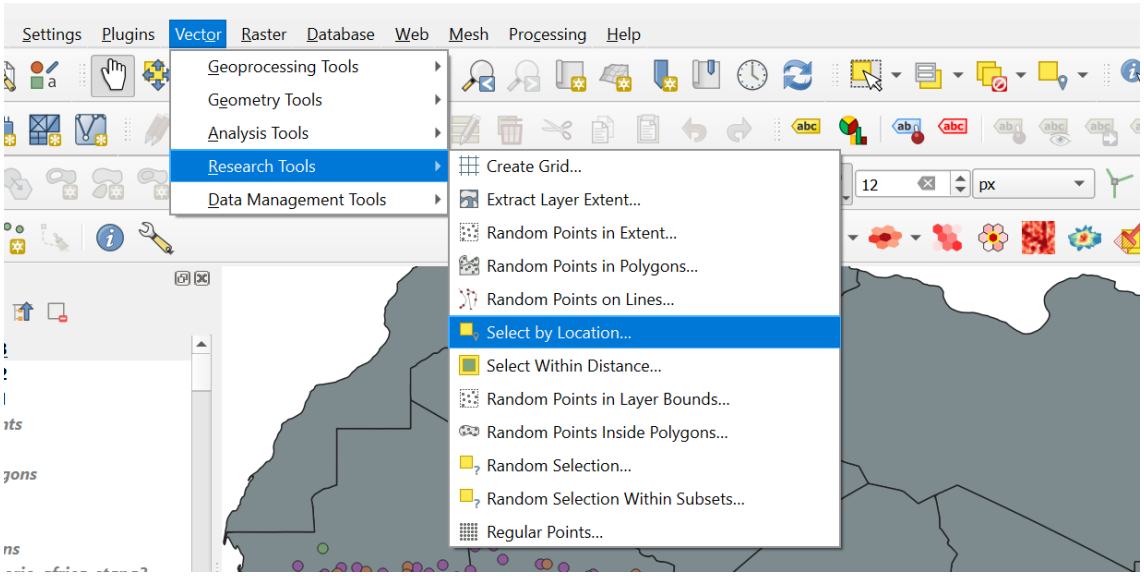
- Import the CSV files “nocturnal_stage_1”, “nocturnal_stage_2”, and “nocturnal_stage_3” into QGIS.

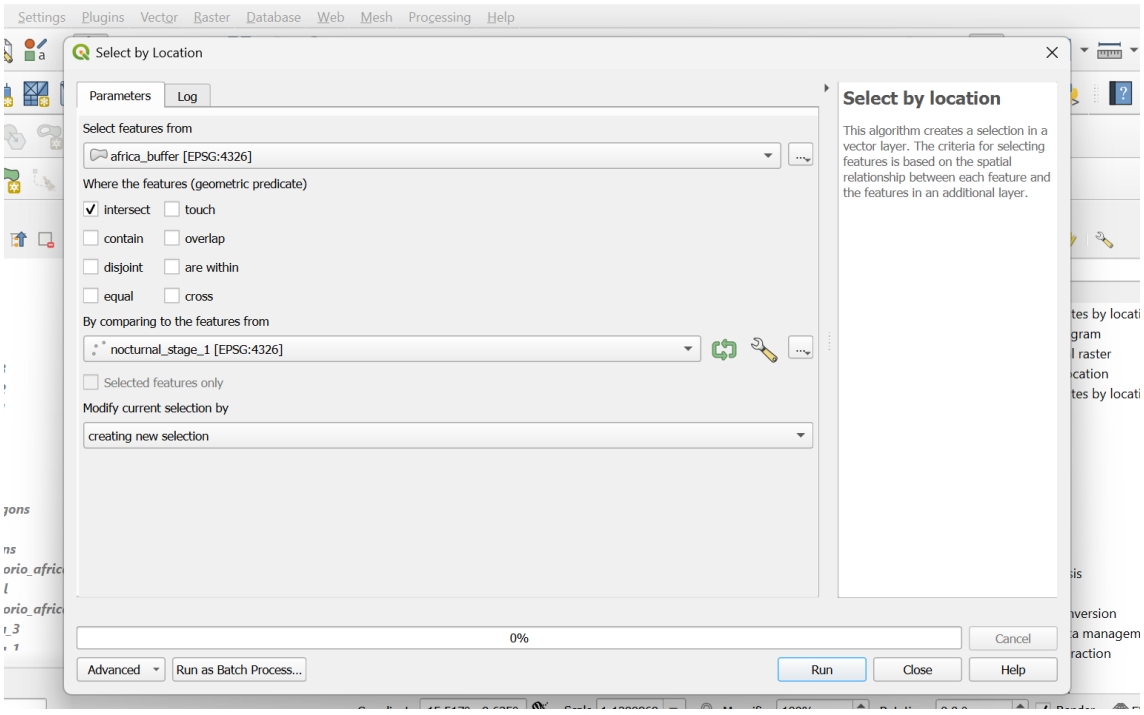


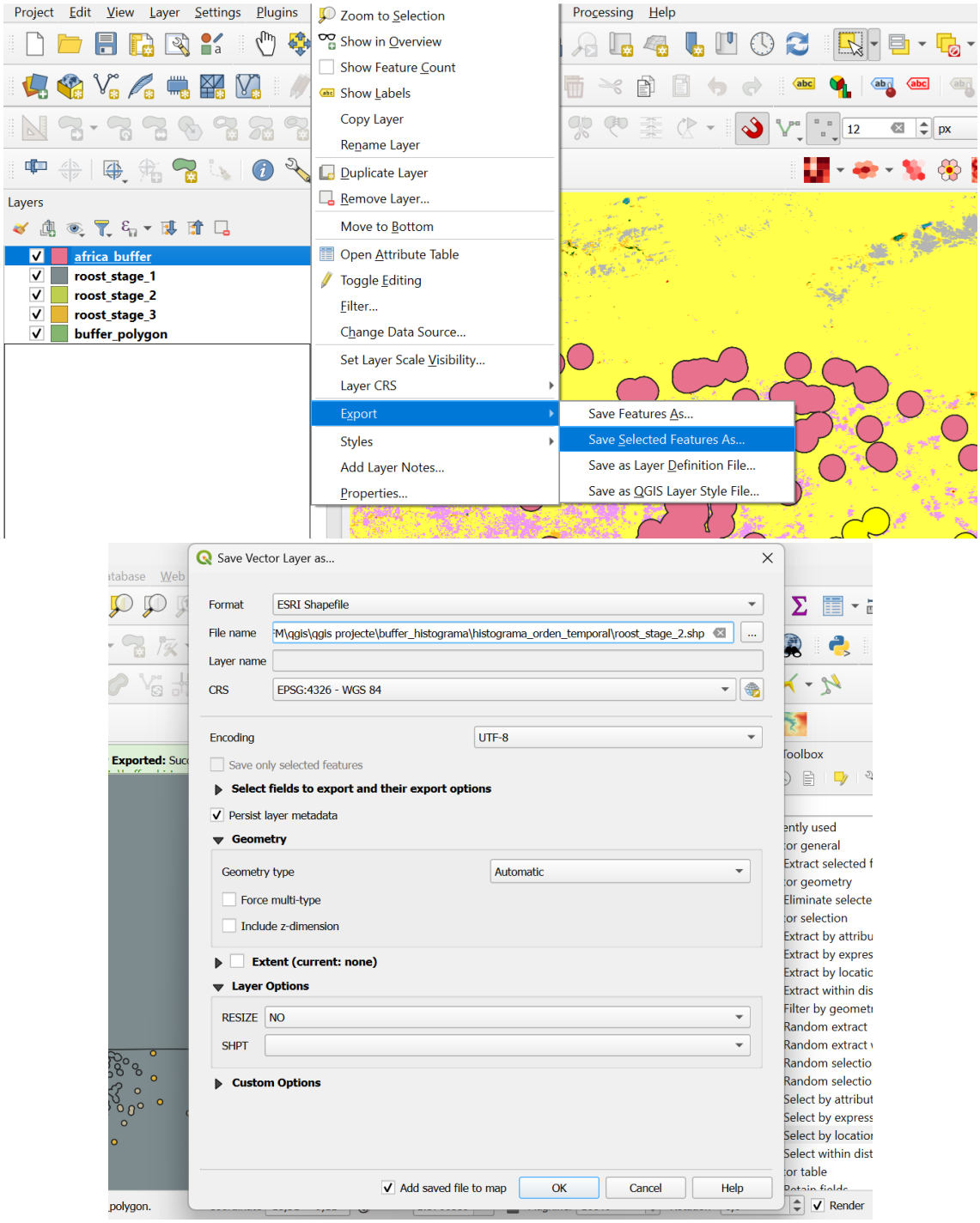
- Join the “buffer_polygon” layer with the Africa countries layer, and save the resulting layer as “africa_buffer.shp”.



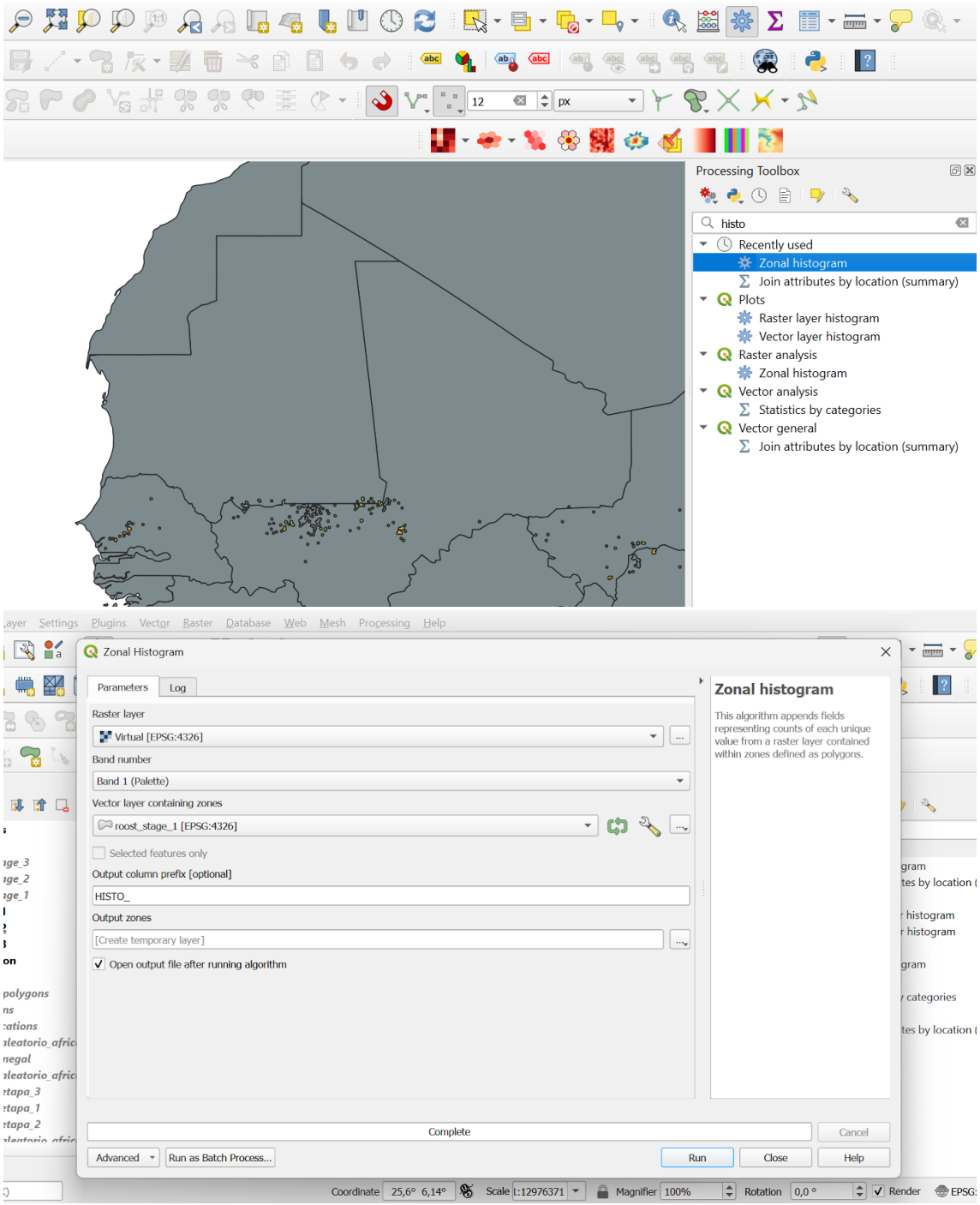
- Use “Select by Location” with the nocturnal stage layers to classify roosts by stages. Export and save each selection as “roost_stage_1”, “roost_stage_2”, and “roost_stage_3”.


Settings
Plugins
Vector
Raster
Database
Web
Mesh
Processing
Help
Geoprocessing Tools
Geometry Tools
Analysis Tools
Research Tools
Data Management Tools
Create Grid...
Extract Layer Extent...
Random Points in Extent...
Random Points in Polygons...
Random Points on Lines...
Select by Location...
Select Within Distance...
Random Points in Layer Bounds...
Random Points Inside Polygons...
Random Selection...
Random Selection Within Subsets...
Regular Points...
12
px


Select by Location
Parameters
Log
Select features from
africa_buffer [EPSG:4326]
Where the features (geometric predicate)
intersect
touch
contain
overlap
disjoint
are within
equal
cross
By comparing to the features from
nocturnal_stage_1 [EPSG:4326]
Selected features only
Modify current selection by
creating new selection
Select by location
This algorithm creates a selection in a vector layer. The criteria for selecting features is based on the spatial relationship between each feature and the features in an additional layer.
0%
Cancel
Advanced
Run as Batch Process...
Run
Close
Help



- In the Processing Toolbox, search for the “Zonal Histogram” tool to create histograms for each stage layer (roost_stage_1, roost_stage_2, roost_stage_3).



- Save the results as “roost_stage_histo_1.xlsx”, “roost_stage_histo_2.xlsx”, and “roost_stage_histo_3.xlsx”.

- In Excel, calculate the proportions manually by summing the pixel values in each row (the total pixels per buffer) in a new column called “total_pixel.” Then, divide the pixel count for each habitat category (for example, “HISTO_10,” “HISTO_20,” etc.) by that total. This will yield the proportions for each category, which can be named “PROP_10,” “PROP_20,” etc..

4. Dominant Habitat Identification and Analysis

4.1. Identification of the Dominant Habitat

For each buffer, the dominant habitat type was identified using an Excel formula.

- A new column, named “dominant habitat,” was created, in which the following formula was applied:
  =INDEX([row of habitat names], MATCH(MAX([row of habitat proportions]), [row of habitat proportions], 0))
- This formula compares the proportional values of each habitat type within the buffer (e.g., PROP_10, PROP_20, etc.) and returns the name of the habitat with the greatest proportional representation.
- The formula was applied to each row in the dataset to determine the dominant habitat type at each roosting site.

4.2. Country-Level Summary of Dominant Habitat Types

- The dataset was filtered by the "Country" column to isolate the records for each country.
- Using the “dominant habitat” column, the percentage of roosting sites dominated by each habitat type (e.g., PROP_10, PROP_20, etc.) was calculated for each country.
- This was done by creating a new column for each habitat type, named for example:
  - dominant habitat PROP_10
  - dominant habitat PROP_20
  - dominant habitat PROP_30
  - etc.
- In each column, the following formula was used:
  =COUNTIF([range of “dominant habitat” column], "PROP_30") / COUNTA([same range]) * 100
  (Replace "PROP_30" with the appropriate habitat code for each column.)
- These values represent the percentage of roosts in the country where that specific habitat type was dominant.
- The final percentages were compiled in a summary table, with each row representing a country and each column a habitat category.

- This table was saved in the sheet named “africa habitat” within each stage file, allowing for a comparative overview across countries.

4.3. Consolidation by Stage

The “africa habitat” sheets from the three stages were consolidated into a single Excel file titled “dominant_habitat_by_stage.xlsx.” This file includes one sheet per stage, each summarising the percentage of roosting sites dominated by each habitat type across the countries included in that stage.

4.4. Conversion to Absolute Frequencies

- Plan a Chi-squared test to assess whether significant differences exist in dominant habitat types across stages and countries.
- Convert the percentages of dominant habitats in the “dominant_habitat_by_stage.xlsx” file into absolute counts by multiplying each percentage by the total number of roosting sites for each country and stage. Absolute frequency = (Percentage / 100) × Total number of roosts
- Use the resulting absolute counts to create contingency tables for the Chi-squared analysis

4.5. Chi-squared analysis is performed in R

- The file “dominant_habitat_by_stage.xlsx” is loaded into R for analysis.
- Sum the occurrences of each habitat type classified as dominant within each stage (stage 1, stage 2, stage 3) to generate a summary table containing the absolute counts of dominant habitat types per stage.

```
freq_stages <- df %>%
  group_by(etapa) %>%
  summarise(
    freq_20 = sum(freq_20, na.rm = TRUE),  # Shrubland
    freq_30 = sum(freq_30, na.rm = TRUE),  # Grassland
    freq_40 = sum(freq_40, na.rm = TRUE),  # Cropland
    freq_60 = sum(freq_60, na.rm = TRUE),  # Bare vegetation
    freq_80 = sum(freq_80, na.rm = TRUE),  # Permanent water
    freq_90 = sum(freq_90, na.rm = TRUE)   # Herbaceous wetland
  )
```

- Convert the summarised table into a matrix, excluding the stage labels. This matrix will serve as the contingency table for the Chi-squared test.

```
chi_matrix <- as.matrix(freq_stages[, -1])
```

- Run a Chi-squared test using a simulated p-value approach (based on 10,000 permutations) to test whether the distribution of dominant habitats differs significantly across the three stages.

```
chi_test <- chisq.test(chi_matrix, simulate.p.value = TRUE, B = 10000)
```

- Display the results

```
print(freq_stages)
print(chi_test)
```

- If the global test indicates significant differences, perform Chi-squared tests between all stage combinations (Stage 1 vs Stage 2, Stage 1 vs Stage 3, and Stage 2 vs Stage 3) to identify which specific comparisons contribute to the overall pattern of variation.
- Prior to each test, remove any rows or columns with zero marginal totals to prevent errors and ensure valid comparisons.

```
clean_table <- function(table) {
  table <- table[rowSums(table) > 0, , drop = FALSE]
  table <- table[, colSums(table) > 0, drop = FALSE]
  return(table)
}
```

- For each pair of stages, construct a contingency table of habitat frequencies, remove any rows or columns with zero totals, and perform a Chi-squared test using simulated p-values.

```
pairs <- combn(unique(freq_stages$etapa), 2, simplify = FALSE)

for (combo in pairs) {
  cat("\nComparison between stages:", combo[1], "and", combo[2], "\n")

sub_freq <- freq_stages %>% filter(etapa %in% combo)

sub_matrix <- as.matrix(sub_freq[, -1])
  rownames(sub_matrix) <- sub_freq$etapa

  sub_matrix <- clean_table(sub_matrix)
  if (all(dim(sub_matrix) > 1)) {
    pair_test <- chisq.test(sub_matrix, simulate.p.value = TRUE, B = 10000)
    print(pair_test)
  } else {
    cat("Table too small for Chi-squared test\n")
  }
}
```

- Evaluate the resulting p-values to determine whether significant differences exist in dominant habitat types between stages, which may reveal potential temporal shifts in habitat use by the species.

4.6. Visualisation of Average Habitat Proportions by Country and Stage

Begin with the Excel file titled africa_buffer_histo_prop.xlsx, which contains habitat proportion data for roosting sites across various African countries.

- Use the sheet titled “Africa habitats,” which consolidates the proportional coverage of each habitat type within buffers by country, providing the necessary data to compare habitat dominance across countries.
- This dataset is used to calculate the average habitat proportions per country and to visualise differences in habitat dominance across the study area.
- Load the necessary libraries for data manipulation and visualisation: readxl to read Excel files, dplyr and tidyr for data wrangling, and ggplot2 for plotting.
- Read the worksheet titled “Africa habitats,” which contains habitat proportion data for each country.

```
df <- read_excel(ruta, sheet = "Africa habitats")
```

Rename the column NAME to Country to ensure consistent and clear column naming.

```
df <- df %>% rename(Country = NAME)
```

- Normalise the habitat proportion columns (PROP_columns) so that the sum of each row equals 1, ensuring correct proportional values even if the input data were not previously normalised.

```
df <- df %>%
  rowwise() %>%
  mutate(suma = sum(c_across(starts_with("PROP_")), na.rm = TRUE)) %>%
  mutate(across(starts_with("PROP_"), ~ . / suma)) %>%
  ungroup() %>%
  select(-suma)
```

- Group the dataset by country and calculate the mean proportion of each habitat type. This summarises the average habitat composition for each country.

```
df_resumen <- df %>%
  group_by(Country) %>%
  summarise(across(starts_with("PROP_"), \(x) mean(x, na.rm = TRUE)))
```

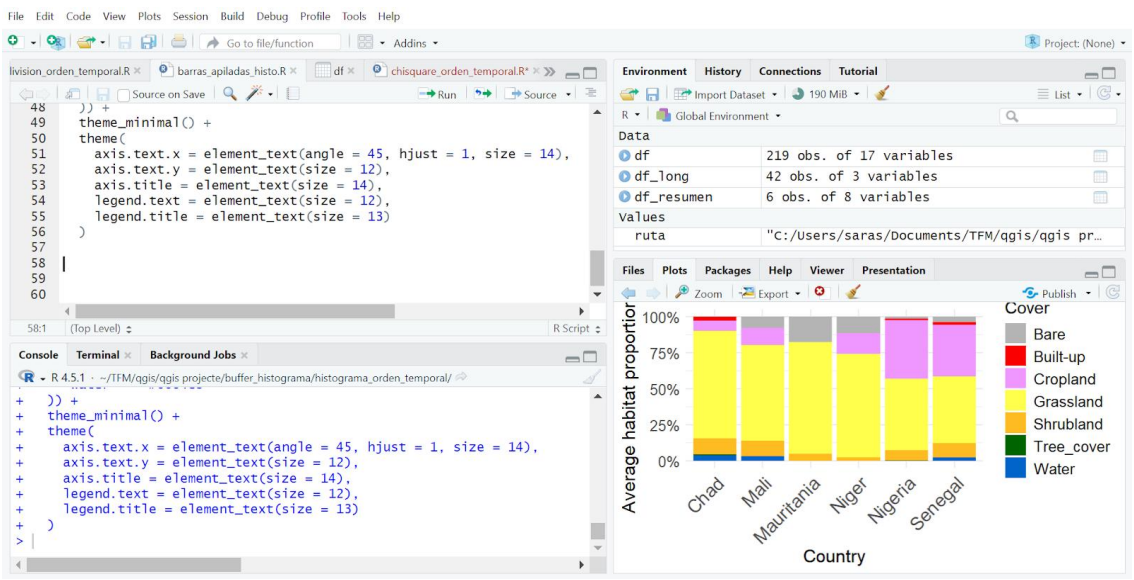
- Transform the summarised data from wide to long format to facilitate plotting with ggplot2.”

```
df_long <- df_resumen %>%
  pivot_longer(cols = starts_with("PROP_"),
               names_to = "habitat",
               values_to = "proportion") %>%
  mutate(habitat = recode(habitat,
                          "PROP_10" = "Tree_cover",
                          "PROP_20" = "Shrubland",
                          "PROP_30" = "Grassland",
                          "PROP_40" = "Cropland",
                          "PROP_50" = "Built-up",
                          "PROP_60" = "Bare",
                          "PROP_90" = "Water"))
```

- Create a stacked bar plot to visualise the average habitat proportions by country. Customise the colours, axis labels, legend, and overall theme to enhance clarity and visual quality.

```
ggplot(df_long, aes(x = Country, y = proportion, fill = habitat)) +
  geom_bar(stat = "identity") +
  labs(x = "Country",
       y = "Average habitat proportion (%)",
       fill = "Habitat type") +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  scale_fill_manual(values = c(
    "Tree_cover" = "#006400",
    "Shrubland"  = "#ffbb22",
    "Grassland"  = "#ffff4c",
    "Cropland"   = "#f096ff",
    "Built-up"   = "#fa0000",
    "Bare"       = "#b4b4b4",
    "Water"      = "#0064c8"
  )) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, size = 14),
    axis.text.y = element_text(size = 12),
    axis.title  = element_text(size = 14),
    legend.text = element_text(size = 12),
    legend.title = element_text(size = 13)
  )
```

- To generate similar plots for other stages or sheets (e.g., "roost_stage_histo_1", "roost_stage_histo_2", "roost_stage_histo_3"), simply replace the sheet name in the read_excel() function while keeping the rest of the code unchanged, assuming the data structure remains the same.

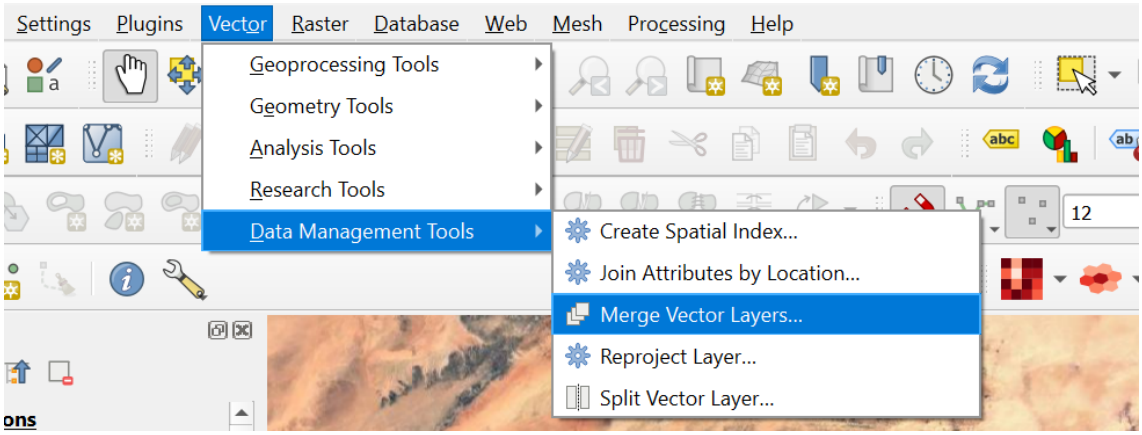

File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
livision_orden_temporal.R
barras_apiladas_histo.R
df
chisquare_orden_temporal.R*
Source on Save
Run
Source
48)) +
49 theme_minimal() +
50 theme(
51 axis.text.x = element_text(angle = 45, hjust = 1, size = 14),
52 axis.text.y = element_text(size = 12),
53 axis.title = element_text(size = 14),
54 legend.text = element_text(size = 12),
55 legend.title = element_text(size = 13)
56)
57
58
59
60
58:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.5.1 · ~/TFM/qgis/qgis projecte/buffer_histograma/histograma_orden_temporal/
+)) +
+ theme_minimal() +
+ theme(
+ axis.text.x = element_text(angle = 45, hjust = 1, size = 14),
+ axis.text.y = element_text(size = 12),
+ axis.title = element_text(size = 14),
+ legend.text = element_text(size = 12),
+ legend.title = element_text(size = 13)
+)
>
Environment
History
Connections
Tutorial
Import Dataset
190 MiB
List
R
Global Environment
Data
df 219 obs. of 17 variables
df_long 42 obs. of 3 variables
df_resumen 6 obs. of 8 variables
Values
ruta "C:/Users/saras/Documents/TFM/qgis/qgis pr…
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Average habitat proportion
100%
75%
50%
25%
0%
Chad
Mali
Mauritania
Niger
Nigeria
Senegal
Country
Cover
Bare
Built-up
Cropland
Grassland
Shrubland
Tree_cover
Water

5. Habitat diversity

5.1. Temporal separation of GPS points using R

5.1.1. Combine shapefile layers:

- Beginning with the marking campaigns, all recorded locations within the Sahel region were selected, including both diurnal and nocturnal records without differentiation. To facilitate data integration, all shapefile layers were merged using the *'Merge Vector Layers'* tool.



- The merged layer was subsequently exported as 'diurnal_nocturnal_locations.xlsx'. This consolidated dataset is used to create a polygon delineating the overall movement range of the individuals.

5.1.2. Load necessary libraries:

Readxl, dplyr, lubridate, and writexl.

5.1.3. Rename coordinate columns:

The columns 'location-l' and 'location_1' were renamed to 'latitude' and 'longitude', respectively, to facilitate subsequent data processing.

5.1.4. Convert dates and classify stages:

The timestamp column was converted to POSIXct datetime format with the UTC timezone. A new column, 'date', containing only the calendar date, was derived from the timestamp. Subsequently, the 'day' and 'month' columns were extracted.

5.1.5. Records are then classified into three temporal stages based on day and month:

Stage 1: from September 20 to October 19
Stage 2: from October 20 to November 19

Stage 3: from November 20 to April 25

```
data <- data %>%
  mutate(
    timestamp = as.POSIXct(timestamp, format = "%d/%m/%Y %H:%M", tz = "UTC"),
    date = as.Date(timestamp),
    day = day(date),
    month = month(date),
    stage = case_when(
      (month == 9 & day >= 20) | (month == 10 & day <= 19) ~ "Stage 1",
      (month == 10 & day >= 20) | (month == 11 & day <= 19) ~ "Stage 2",
      (month == 11 & day >= 20) | month %in% c(12, 1, 2, 3) | (month == 4 & day <= 25)
~ "Stage 3",
      TRUE ~ NA_character_
    )
  )
```

5.1.6. Save CSV files by stage:

Filter the data for each stage, selecting columns individual, latitude, longitude, timestamp, and stage.

Save each filtered dataset as a separate CSV file named:

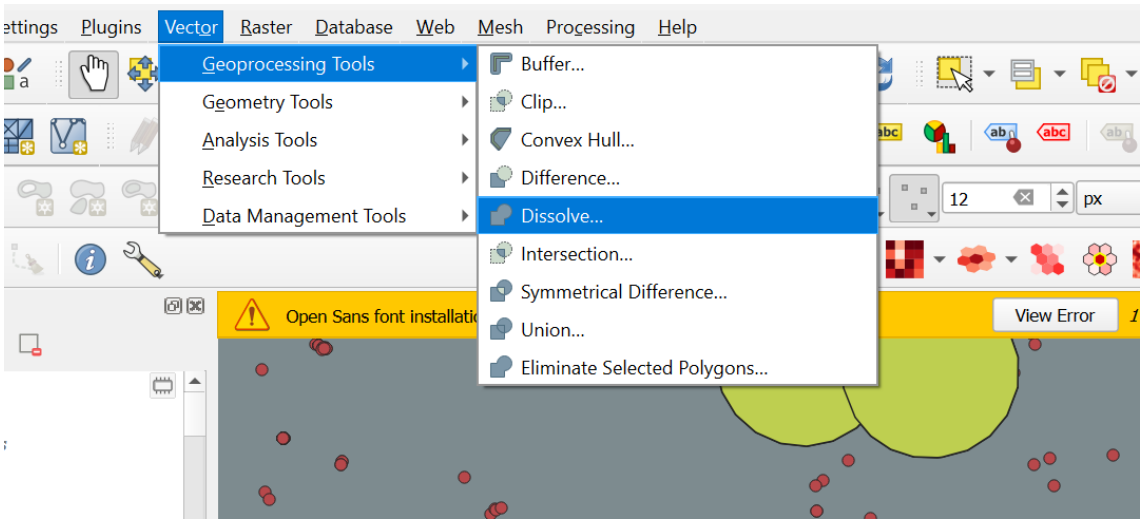
```
diurnal_nocturnal_locations_stage1.csv
diurnal_nocturnal_locations_stage2.csv
diurnal_nocturnal_locations_stage3.csv

write.csv2(
  filter(data, stage == "Stage 1")[, c("individual", "latitude", "longitude", "timestamp",
"stage")],
  file.path(output_path, "diurnal_nocturnal_locations_stage1.csv"),
  row.names = FALSE
)

write.csv2(
  filter(data, stage == "Stage 2")[, c("individual", "latitude", "longitude", "timestamp",
"stage")],
  file.path(output_path, "diurnal_nocturnal_locations_stage2.csv"),
  row.names = FALSE
)

write.csv2(
  filter(data, stage == "Stage 3")[, c("individual", "latitude", "longitude", "timestamp",
"stage")],
```

```
  file.path(output_path, "diurnal_nocturnal_locations_stage3.csv"),
  row.names = FALSE
)
```

5.2. Processing in QGIS for each stage

5.2.1. Dissolve the points

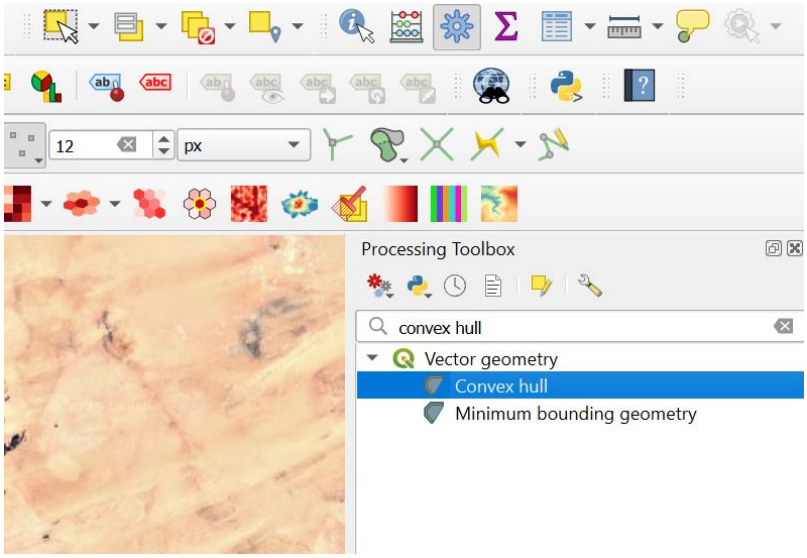
- Use the Dissolve tool (Vector > Geoprocessing Tools > Dissolve) without selecting any fields to merge all overlapping geometries into a single polygon, applied to the CSV layer (diurnal_nocturnal_locations_stage1, diurnal_nocturnal_locations_stage2, diurnal_nocturnal_locations_stage1).



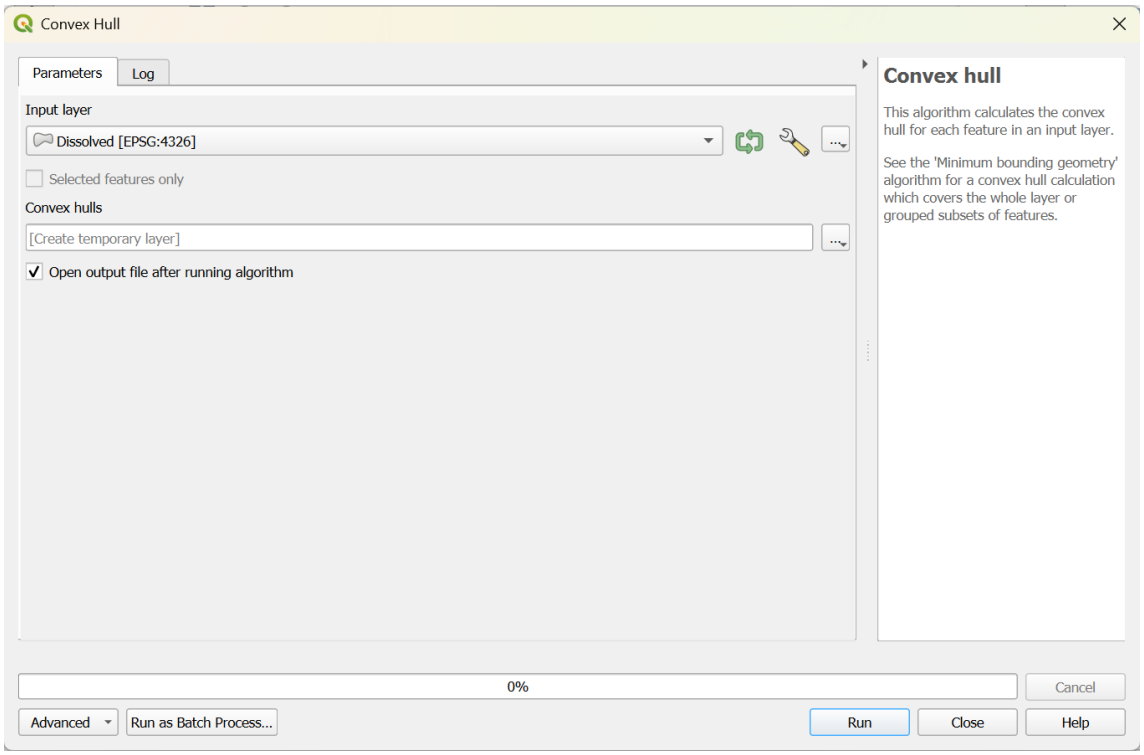
- Repeat this process for each stage.

5.2.2. Calculate the convex hull

- Menu: Vector > Geometry > Convex Hull



- Use the dissolved layer as input.



- This generates a polygon delineating the available area for the corresponding stage and population.

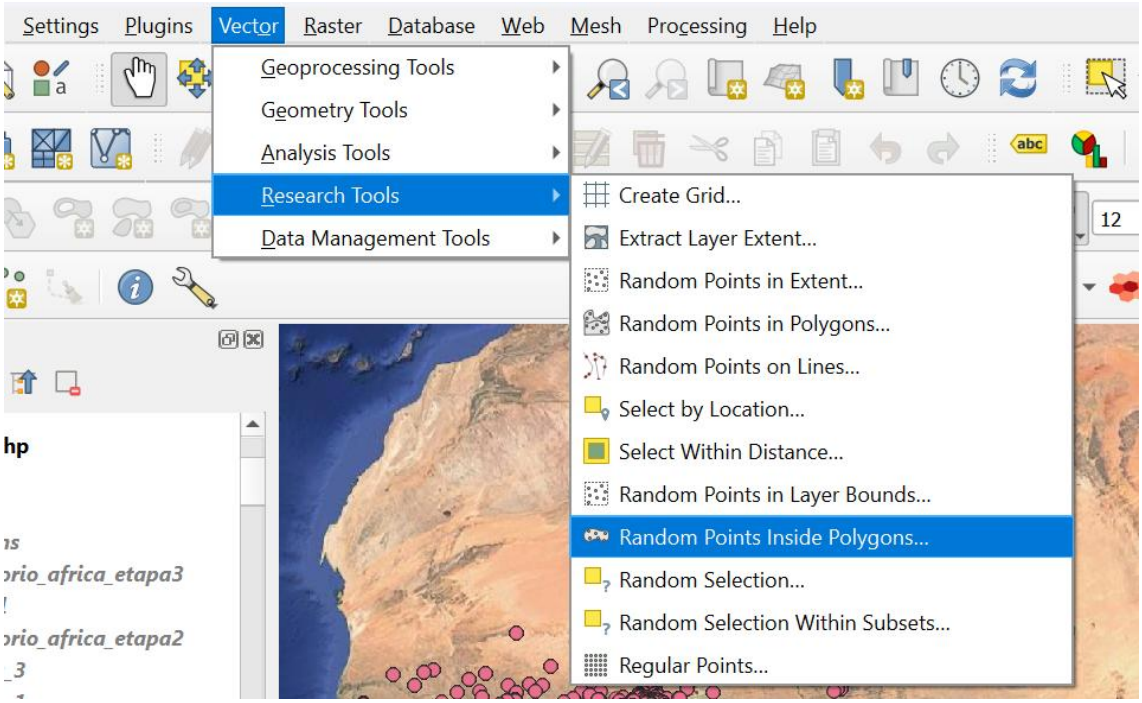
5.2.3. Rename and save the layer

- Save the generated polygon with the name:

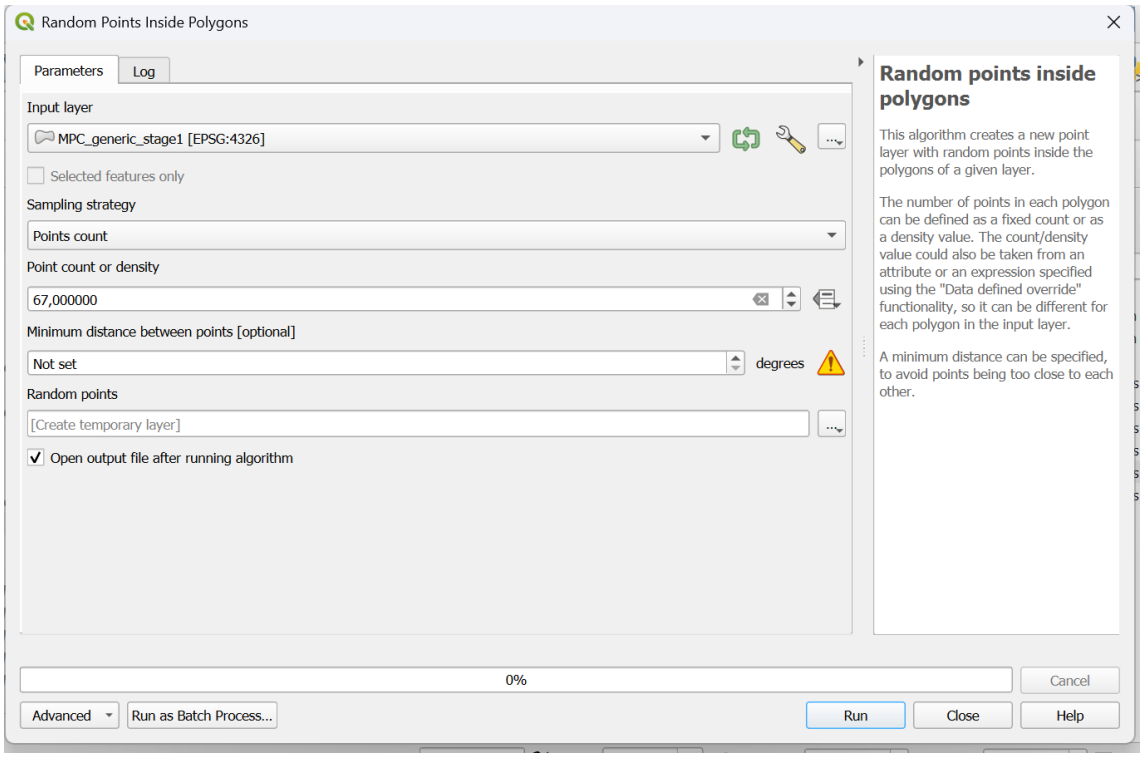
MPC_generic_stage1.shp
MPC_generic_stage2.shp
MPC_generic_stage3.shp

5.3. Create random points inside the polygon

- Generate random points inside the polygons using the “Random Points Inside Polygons”.



- Select the input layer MCP_generic_stage1.shp (or the corresponding stage) and set the number of random points to match the number of roosts for that stage.



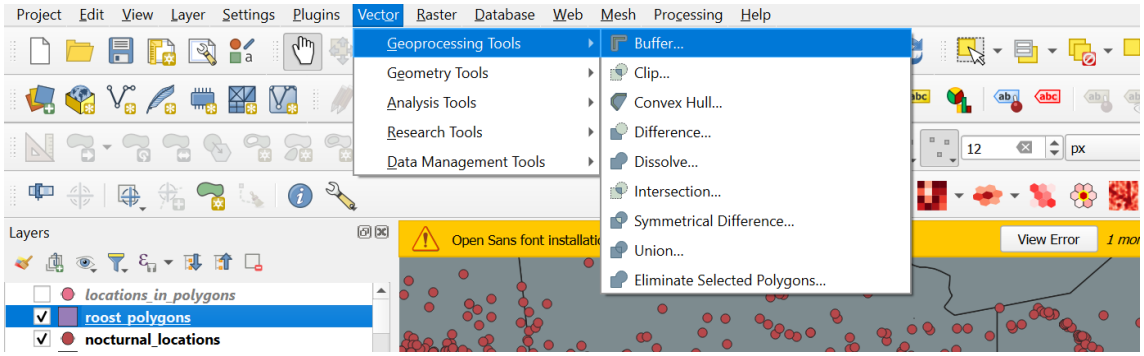
- Save the generated random points with the following names, corresponding to each stage:

random_points_stage1.shp
random_points_stage2.shp
random_points_stage3.shp

5.4. Create a 5 km buffer around the random points

- Generate 5 km buffers for the generated random points and save the layer with the following names:

MCP_random_buffer_stage1.shp
MCP_random_buffer_stage2.shp
MCP_random_buffer_stage3.shp



5.5. Generate a zonal histogram from the habitat raster

- Using the 'MCP_random_buffer' layers from all three stages, apply the Zonal Histogram tool to extract the number of pixels corresponding to each land cover category (see Document 3 for details).

5.6. Join attributes by location

- Join the output from histogram with the layer with the Africa countries layer (see Document 3 for details), and save the resulting layer as:

MCP_random_buffer_africa_stage1.xlsx
MCP_random_buffer_africa_stage2.xlsx
MCP_random_buffer_africa_stage3.xlsx

6. Shannon

6.1. Calculate Shannon diversity index for roost and random buffers.

- Using the files "roost_stage_histo_1.xlsx", "roost_stage_histo_2.xlsx" and "roost_stage_histo_3.xlsx", together with their corresponding random buffer files "MCP_random_buffer_africa_stage1.xlsx", "MCP_random_buffer_africa_stage2.xlsx" and "MCP_random_buffer_africa_stage3.xlsx", a new column named "shannon" was generated in each file.

- The Shannon index was calculated from the habitat proportion columns "PROP_10", "PROP_20", "PROP_30", "PROP_40", "PROP_50", "PROP_60", "PROP_80" and "PROP_90" using the following Excel formula:

=-SUMPRODUCT(IF([range of "PROP_10":"PROP_90"]>0, [range of "PROP_10":"PROP_90"]*LN([range of "PROP_10":"PROP_90"]), 0))

- This calculation was applied to both the roost and random buffer datasets to enable a direct comparison of habitat diversity between actual roost locations and randomly selected sites within the movement range.

6.2. Combine Shannon index results for all stages into a single dataset

- After calculating the Shannon index for each stage and each dataset type, all results were compiled into a single Excel file named "shannon.xlsx".
  This file contains the following columns:

"site_type" – indicates whether the buffer is "roost" or "random".

"country" – specifies the country where the site is located.

"stage_type" – indicates the temporal stage (Stage 1, Stage 2, Stage 3).

"shannon_value" – the Shannon diversity index calculated in the previous step.

- This dataset structure enables statistical testing and graphical representation of Shannon diversity in R.

6.3. Load required R libraries:

readxl, dplyr, ggplot2.

6.4. Create legend labels for the plot

Generate a "legend_label" column to make the plot legend clearer:

- "Random" entries remain as "Random".
- "Roost" entries are labelled as "Roost (Period X)", where X corresponds to the period value.

```
shannon_data <- shannon_data %>%
  mutate(
    legend_label = case_when(
      site_type == "Random" ~ "Random",
      site_type == "Roost"  ~ paste0("Roost (", period, ")"),
      TRUE ~ site_type
    )
  )
```

6.5. Define a custom colour palette

Assign distinct colours for "Roost (Period 1)" (green), "Roost (Period 2)" (orange), "Roost (Period 3)" (purple), and "Random" (grey).

```
legend_colors <- c(
  "Roost (Period 1)" = "#1b9e77",  # green
  "Roost (Period 2)" = "#d95f02",  # orange
  "Roost (Period 3)" = "#7570b3",  # purple
  "Random" = "#999999"             # gray
)
```

6.6. Create a boxplot visualisation

Using ggplot2, plot "shannon" values on the y-axis grouped by "period" on the x-axis, with fill colours determined by "legend_label".

- The x-axis represents the period,
- The y-axis represents the Shannon index values,
- Fill colours set according to "legend_label" to distinguish roost and random points across periods.

```
ggplot(shannon_data, aes(x = period, y = shannon, fill = legend_label)) +
  geom_boxplot(position = position_dodge(width = 0.8)) +
  labs(title = "Shannon diversity by site type and period",
       x = "Period", y = "Shannon index", fill = "Site type") +
  scale_fill_manual(values = legend_colors) +
```

```
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

6.7.Statistical comparison between Roost and Random sites per period

- Assess data normality with Shapiro-Wilk.
- For each group (Roost and Random in each period), a Shapiro-Wilk test is performed to check whether Shannon index values follow a normal distribution.
- The test statistic W and the exact p-value are calculated.
- If $p < 0.05$, the data do not meet normality, justifying the use of a non-parametric test such as the Mann-Whitney U test.
- If $p \geq 0.05$, the data meet normality, allowing the use of a two-sample t-test.

```
shapiro_results <- shannon_data %>%
  filter(site_type %in% c("Roost", "Random")) %>%
  group_by(period, site_type) %>%
  summarise(
    shapiro_test = list(shapiro.test(shannon)),
    .groups = "drop"
  ) %>%
  mutate(
    W = sapply(shapiro_test, function(x) x$statistic),
    p_value = sapply(shapiro_test, function(x) x$p.value)
  ) %>%
  select(-shapiro_test)

print(shapiro_results)
```

- Compute the median, minimum and maximum range, and sample size (n) for each period and site type. These statistics summarise the Shannon index distribution without assuming normality.

```
summary_stats <- shannon_data %>%
  filter(site_type %in% c("Roost", "Random")) %>%
  group_by(period, site_type) %>%
  summarise(
    median_shannon = median(shannon, na.rm = TRUE),
    min_shannon = min(shannon, na.rm = TRUE),
```

```
    max_shannon = max(shannon, na.rm = TRUE),
    n = n(),
    .groups = "drop"
  )

print(summary_stats)
```

- Perform Mann-Whitney U tests by period comparing Roost vs Random.
- For each period, a non-parametric Mann-Whitney U test (wilcox.test) is performed to compare Shannon index values between roost sites and random points.
- A clean summary of the results is obtained using broom::tidy.

```
wilcox_results <- shannon_data %>%
  filter(site_type %in% c("Roost", "Random")) %>%
  group_by(period) %>%
  summarise(
    wilcox_test = list(wilcox.test(shannon ~ site_type, exact = FALSE)),
    tidy = list(broom::tidy(wilcox.test(shannon ~ site_type, exact = FALSE))),
    .groups = "drop"
  ) %>%
  unnest(tidy) %>%
  select(period, estimate1, estimate2, statistic, p.value, method)

print(wilcox_results)
```

7. Tree species preference for nocturnal roosting

7.1. Prepare input data

Ensure the dataset contains two columns:

- "tree_species": the name of the tree species.
- "presence_absence": binary values 1 (presence) or 0 (absence).

7.2. Create a contingency table

- Use table() to cross-tabulate "tree_species" and "presence_absence".
- Store the result in the object tabla.
- Print the table to inspect frequency counts.

7.3. Run Fisher's exact test

- Apply fisher.test(tabla) to evaluate the statistical association between tree species and presence/absence.

7.4. Convert table to data frame

- Transform tabla into a data frame using as.data.frame(as.table(tabla)).
- Rename columns to "Species", "Presence", and "Frequency".

7.5. Create stacked bar plot (proportional)

- Use ggplot2 with:
  - x = Species (tree species names on the x-axis).
  - y = Frequency (counts of presence/absence).
  - fill = as.factor(Presence) to differentiate presence (1) and absence (0) by colour.
- Apply geom_bar(stat = "identity", position = "fill") to show proportions instead of raw counts.
- Customise axis labels (labs) and rotate x-axis text with theme(axis.text.x = element_text(angle = 90, hjust = 1)) for readability.

8. Interannual roost fidelity

Spatial fidelity was assessed for individuals with GPS records spanning at least two winters by integrating their GPS data with the previously generated roost polygon layer, 'locations_in_poligons'.

8.1. Data loading and preparation in R

- Load the dataset containing individual locations in roost polygons.
- Convert timestamps to date format and extract the year.
- Filter out records with missing values in individual, ID, or year.

```
ind_in_roosts <- read_excel("conjunto_id.xlsx") %>%
  mutate(timestamp = lubridate::dmy(timestamp),
         year = lubridate::year(timestamp)) %>%
  filter(!is.na(individual), !is.na(ID), !is.na(year))
```

8.2. Count the number of different years each individual uses each roost:

```
roost_repeats <- ind_in_roosts %>%
  group_by(individual, ID) %>%
  summarise(years_used = n_distinct(year), .groups = "drop")
```

8.3. Filter roosts reused in more than one year:

```
reused_roosts <- roost_repeats %>%
  filter(years_used > 1)
```

8.4. Summarise the number of reused roosts per individual:

```
reused_roosts_summary <- reused_roosts %>%
  group_by(individual) %>%
  summarise(n_reused_roosts = n(), .groups = "drop")
```

8.5. Calculate the percentage of individuals showing interannual fidelity

```
total_individuals <- ind_in_roosts %>% distinct(individual) %>% nrow()
individuals_with_fidelity <- reused_roosts_summary %>%
  filter(n_reused_roosts > 0) %>% nrow()
```

```
percent_fidelity <- (individuals_with_fidelity / total_individuals) * 100

cat(sprintf("Total individuals: %d\n", total_individuals))
cat(sprintf("Individuals with reused roosts (fidelity): %d (%.2f%%)\n",
individuals_with_fidelity, percent_fidelity))
```

8.6. Cross-check individuals with multi-year data between roosts and all GPS points

- Load both datasets, convert timestamps, and extract years.
- Identify individuals with data spanning more than one year in each dataset.
- Find common and unique individuals between datasets.

```
ind_in_roosts <- read_excel("conjunto_id.xlsx") %>%
  mutate(timestamp = dmy(timestamp), year = year(timestamp))

all_points <- read_excel("prueba del conjunto definitivo.xlsx") %>%
  mutate(timestamp = dmy(timestamp), year = year(timestamp))

multi_year_ind_roosts <- ind_in_roosts %>%
  group_by(individual) %>%
  summarise(n_years = n_distinct(year), .groups = "drop") %>%
  filter(n_years > 1)

multi_year_all_points <- all_points %>%
  group_by(individual) %>%
  summarise(n_years = n_distinct(year), .groups = "drop") %>%
  filter(n_years > 1)

common_individuals <- intersect(multi_year_ind_roosts$individual,
multi_year_all_points$individual)

cat("Individuals present in both datasets (multi-year):", length(common_individuals),
"\n")
cat("Individuals only in 'all_points':", length(setdiff(multi_year_all_points$individual,
common_individuals)), "\n")
cat("Individuals only in 'ind_in_roosts':",
length(setdiff(multi_year_ind_roosts$individual, common_individuals)), "\n")
```

8.7. Descriptive analysis of roost use per individual and year

- Calculate the number of distinct roosts used by each individual per year.

- Compute mean and standard deviation of roost use.
- Calculate percentages of individuals using only one roost or up to four roosts per season.

```
clean_data <- ind_in_roosts %>%
  filter(!is.na(ID), !is.na(individual), !is.na(year))

roosts_per_individual <- clean_data %>%
  group_by(individual, year) %>%
  summarise(n_roosts = n_distinct(ID), .groups = "drop")

mean_roosts <- mean(roosts_per_individual$n_roosts)
sd_roosts <- sd(roosts_per_individual$n_roosts)

one_roost <- sum(roosts_per_individual$n_roosts == 1)
four_roosts <- sum(roosts_per_individual$n_roosts == 4)
total_records <- nrow(roosts_per_individual)

percent_one <- (one_roost / total_records) * 100
percent_four <- (four_roosts / total_records) * 100

cat(sprintf("Mean ± SD: %.2f ± %.2f\n", mean_roosts, sd_roosts))
cat(sprintf("%d records (%.2f%%) used a single roost.\n", one_roost, percent_one))
cat(sprintf("%d records (%.2f%%) used up to four roosts.\n", four_roosts,
percent_four))
```

8.8.Seasonal roost use per individual:

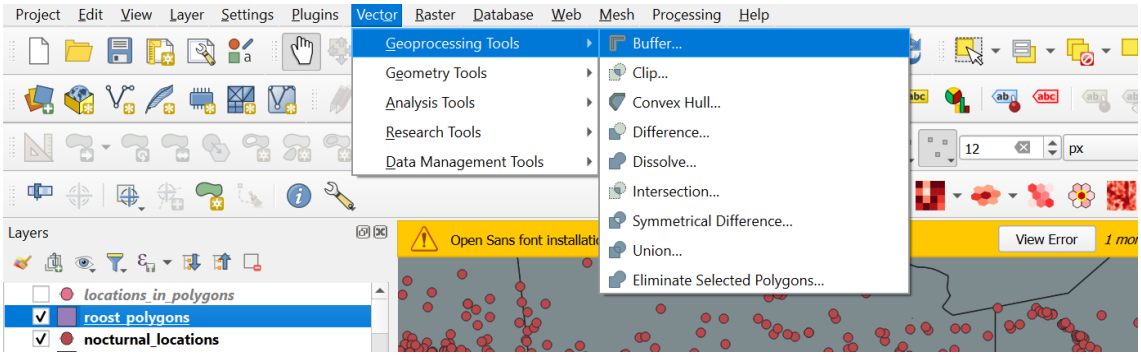
- Calculate the number of roosts used per individual in each wintering season.
- Compute the average and standard deviation of seasonal roost use.

```
seasonal_use <- clean_data %>%
  group_by(individual, year) %>%
  summarise(n_roosts_season = n_distinct(ID), .groups = "drop")

mean_seasonal <- mean(seasonal_use$n_roosts_season)
sd_seasonal <- sd(seasonal_use$n_roosts_season)

cat(sprintf("Average roosts used per individual per season: %.2f ± %.2f\n",
mean_seasonal, sd_seasonal))
```

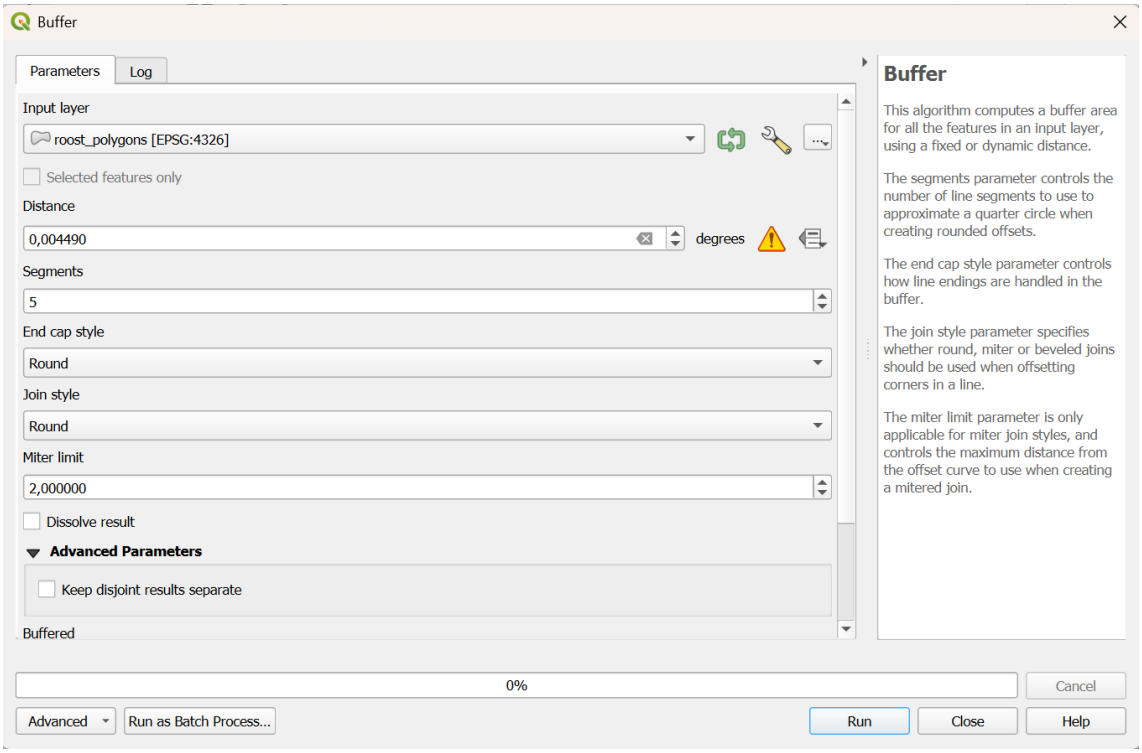
9. Field vs. Satellite Habitat Classification: A Comparative Analysis

9.1. Create a 500 m buffer around each roost centroid

- Start with the point layer representing the nocturnal roost centroids.
- In QGIS, go to: Vector > Geoprocessing Tools > Buffer.



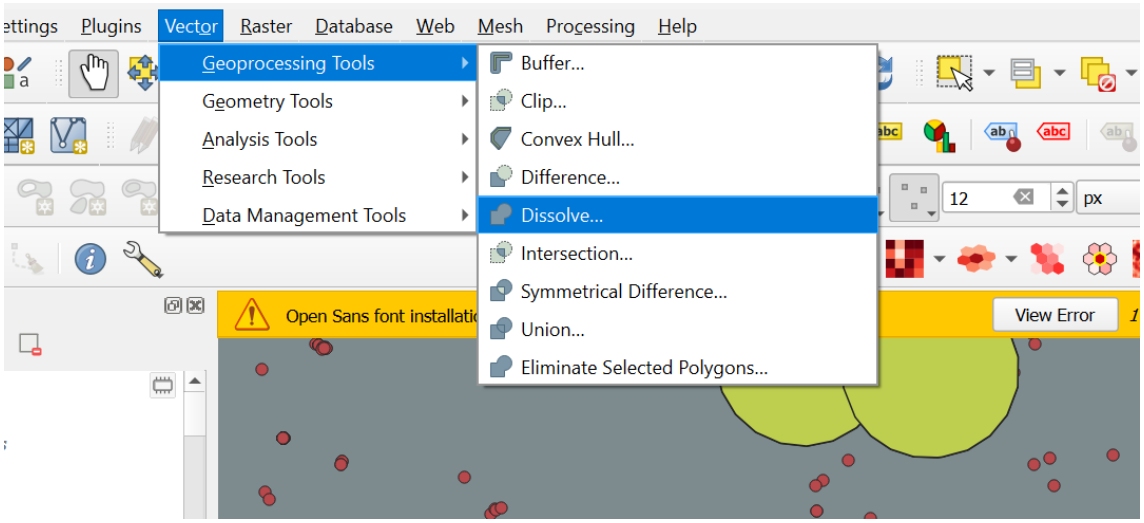
- Select the centroid layer as input and set the buffer distance to 0.00449 degrees (≈ 500 m at Sahel latitude).

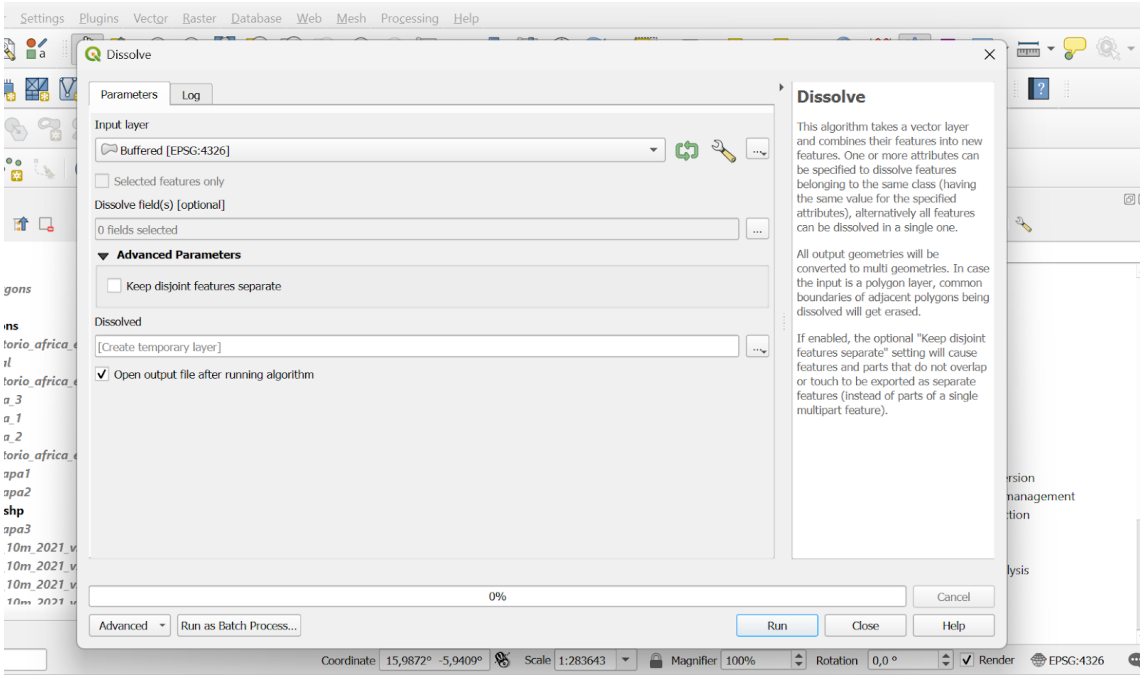


- Run the process to generate the buffer layer.

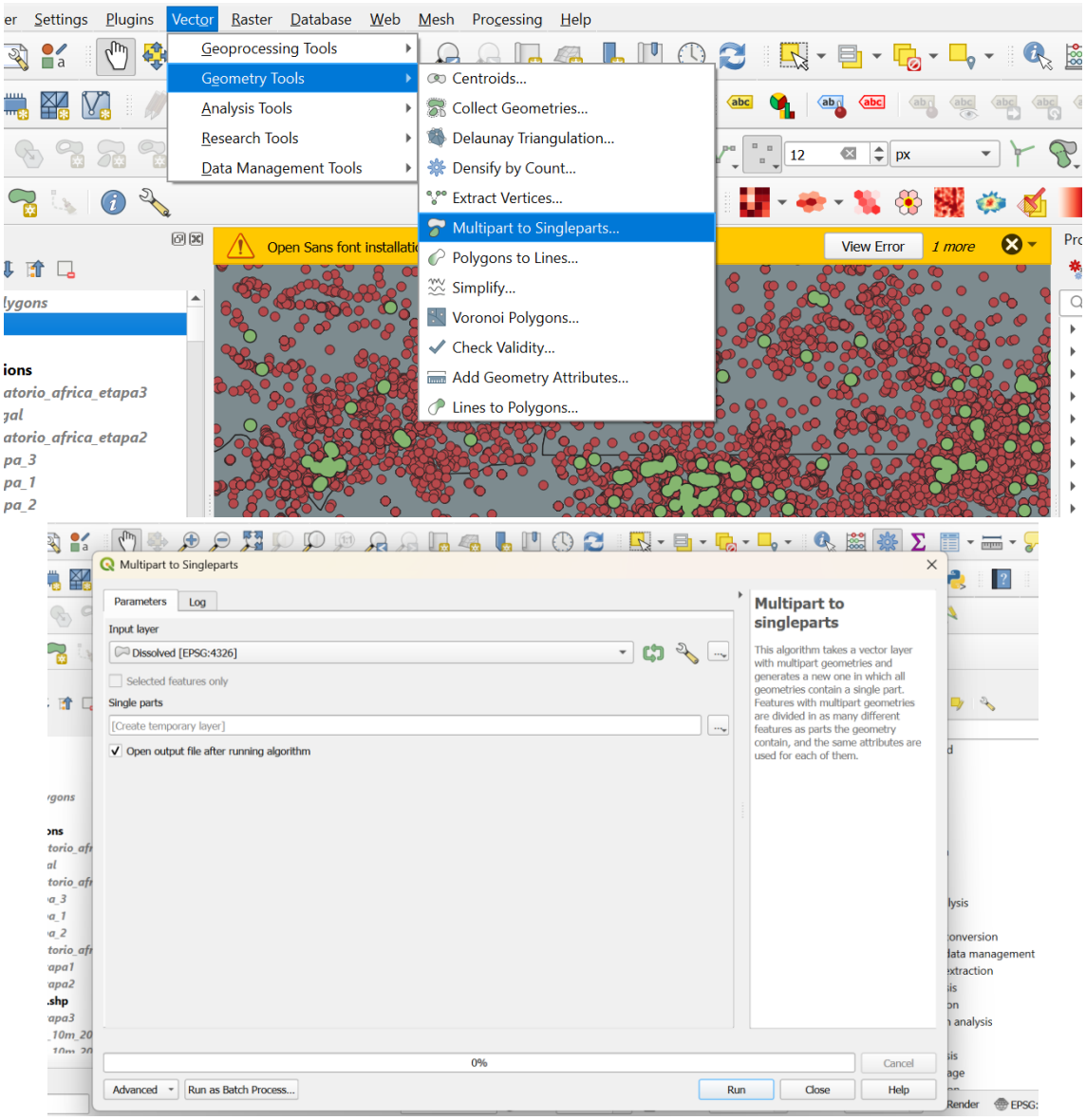
9.2. Dissolve and prepare buffer polygons

- Dissolve overlapping buffers:
  Use the Dissolve tool (Vector > Geoprocessing Tools > Dissolve) without selecting any fields to merge overlapping geometries into a single polygon.

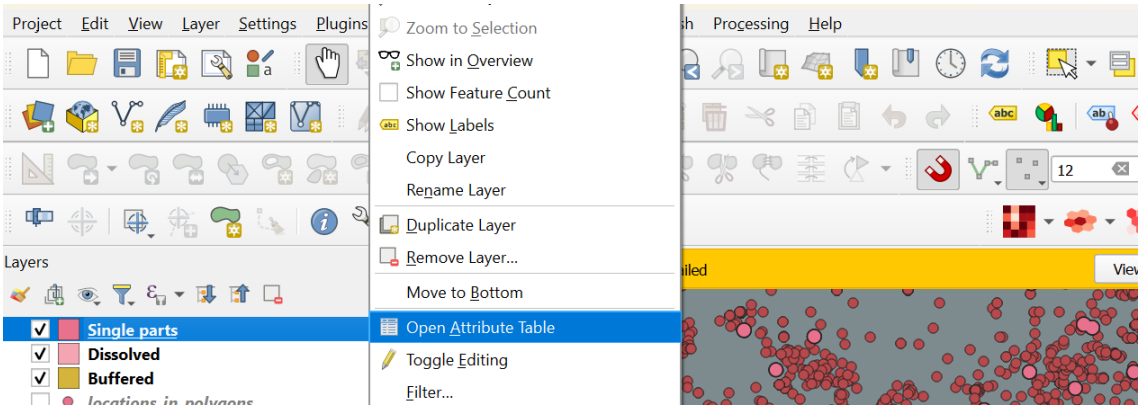


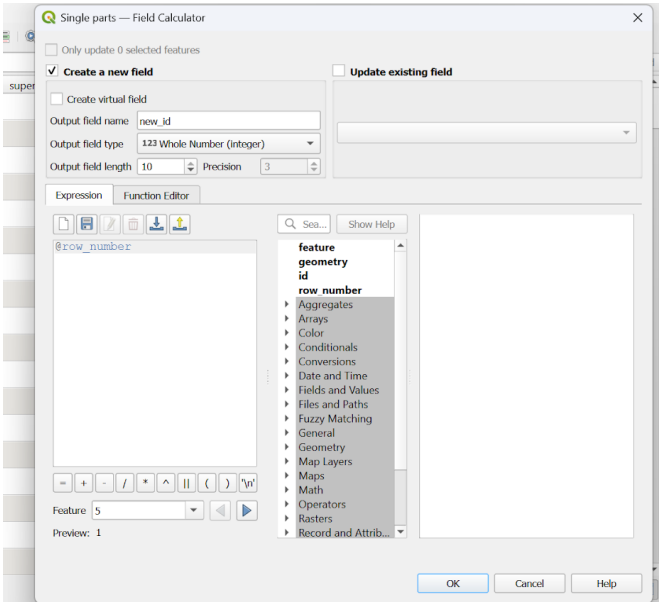


- Use the tool “Multipart to Singleparts” (Vector > Geometry Tools > Multipart to Singleparts) to split the resulting polygon into individual parts.



- Assign a unique ID to each polygon: open the attribute table of the buffer layer and use the Field Calculator to create a new field (e.g., buffer_ID). In the expression box, use @row_number to assign a unique ID to each polygon. This enables subsequent identification and grouping of overlapping buffers.





- Save the layer as “buffer_polygon_500m”.

9.3. Join with reference geographic data

- Use the “Join attributes by location” tool to join “buffer_polygon_500m” with the Africa countries layer, selecting the intersect geometric predicate (see Document 3 for details).
- Export results as:
  - africa_buffer_500m.xlsx
  - africa_buffer_500m.shp

9.4. Prepare raster layers

- Create a virtual raster combining all available land cover or vegetation classification rasters (ESA WorldCover 10 m) (see Document 3 for details).
- This raster will be used to extract habitat information from satellite imagery.

9.5. Extract land cover classes from satellite imagery

- From the “africa_buffer_500m” layer, use the Zonal Histogram tool to obtain the number of pixels of each land cover category within each buffer.
- Set the Output column prefix to HISTO_.
- Save the result as “histogram_500m_buffer.xlsx”.

9.6. Calculate habitat proportions and determine dominant habitat

9.6.1. Calculate habitat proportions

- Based on the HISTO_ columns from the zonal histogram, calculate the total number of pixels per buffer:
  - Create a new column total_pixel = sum of all HISTO_ columns in each row.
- For each land cover class, calculate its proportion:
  - PROP_X = HISTO_X / total_pixel (where X is the class code).
- Store all proportions in new columns prefixed with PROP_.

9.6.2. Determine the dominant habitat from proportions

- Create a new column named "habitat_dominant_satellite" using the Excel formula:

```
=INDEX(HISTO_10:HISTO_90, MATCH(MAX(HISTO_10:HISTO_90), HISTO_10:HISTO_90, 0))
```

9.6.3. Add field survey dominant habitat

- Create a second column named "habitat_dominant_real" to record the dominant habitat code obtained from field surveys for the same roost.
- Keep the same HISTO_ class code format for consistency with satellite-derived data.

9.7. Compare field and satellite habitat data

9.7.1. Load and clean data

- Use R (v4.5.1) with the libraries: tidyr, readxl, dplyr, caret, ggplot2, reshape2, stringr, RColorBrewer.
- Load the table containing:
  - habitat_dominant_sahel (satellite-derived)
  - habitat_dominant_satellite (field-derived)
- Standardise and clean the text in both columns by converting all characters to lowercase and removing any extra spaces.

9.7.2. Data Preparation

The names of the dominant habitat variables were standardised by removing whitespace and converting all text to lowercase, ensuring consistency between field observations (dominant_habitat_field) and satellite-derived data (dominant_habitat_satellite).

```
data <- data %>%
  mutate(
    dominant_habitat_field = tolower(trimws(dominant_habitat_field)),
```

```
  dominant_habitat_satellite = tolower(trimws(dominant_habitat_satellite))
 )
```

9.7.3. A dictionary was created to translate numerical habitat codes (HISTO) into their corresponding English descriptions.

```
habitat_codes <- c(
  "histo_10" = "Tree cover",
  "histo_20" = "Shrubland",
  "histo_30" = "Grassland",
  "histo_40" = "Cropland",
  "histo_50" = "Built-up area",
  "histo_60" = "Bare vegetation",
  "histo_80" = "Permanent water",
  "histo_90" = "Herbaceous wetland",
  "histo_95" = "Mangrove",
  "histo_100" = "Moss & lichen"
)
```

9.7.4. Harmonising Factor Levels

Prior to recoding, both columns were converted to factors. Existing levels were examined and restricted to the common levels to prevent inconsistencies. Subsequently, the levels were recoded using the corresponding English names defined in the dictionary.

```
data$dominant_habitat_field <- as.factor(data$dominant_habitat_field)
data$dominant_habitat_satellite <- as.factor(data$dominant_habitat_satellite)

levels_field <- levels(data$dominant_habitat_field)
levels_satellite <- levels(data$dominant_habitat_satellite)
common_levels <- intersect(levels_field, levels_satellite)

data$dominant_habitat_field <- factor(data$dominant_habitat_field, levels =
common_levels)
data$dominant_habitat_satellite <- factor(data$dominant_habitat_satellite, levels =
common_levels)

levels(data$dominant_habitat_field) <-
habitat_codes[levels(data$dominant_habitat_field)]
levels(data$dominant_habitat_satellite) <-
habitat_codes[levels(data$dominant_habitat_satellite)]
```

9.7.5. Calculation of Confusion Matrix and Class-wise Metrics

The function caret::confusionMatrix was used to calculate the confusion matrix and extract specific metrics per class, such as *Sensitivity, Specificity, User Accuracy, Producer Accuracy*, and *Balanced Accuracy.*

```
conf_mat <- confusionMatrix(data$dominant_habitat_satellite,
data$dominant_habitat_field)

summary_df <- data.frame(
  Class = rownames(conf_mat$byClass),
  Sensitivity = round(conf_mat$byClass[, "Sensitivity"], 2),
  Specificity = round(conf_mat$byClass[, "Specificity"], 2),
  User_Accuracy = round(conf_mat$byClass[, "Pos Pred Value"], 2),
  Producer_Accuracy = round(conf_mat$byClass[, "Sensitivity"], 2),
  Balanced_Accuracy = round(conf_mat$byClass[, "Balanced Accuracy"], 2)
)
summary_df$Class <- gsub("^Class: ", "", summary_df$Class)
```

9.7.6. Visualization of Results

Bar plots were generated to illustrate *User and Producer Accuracy* metrics for each class, alongside a heatmap of the confusion matrix to visually represent the distribution of correct and incorrect classifications.

```
# Bar plot
df_plot <- summary_df %>%
  select(Class, User_Accuracy, Producer_Accuracy) %>%
  pivot_longer(cols = c(User_Accuracy, Producer_Accuracy),
               names_to = "Metric",
               values_to = "Accuracy")

ggplot(df_plot, aes(x = Class, y = Accuracy, fill = Metric)) +
  geom_bar(stat = "identity", position = position_dodge(width = 0.8), color = "black") +
  ylim(0, 1) +
  labs(title = "Accuracy Metrics per Land Cover Class",
       x = "Land Cover Class",
       y = "Accuracy",
       fill = "Metric") +
  theme_minimal() +
  theme(text = element_text(size = 14),
        axis.title = element_text(face = "bold"),
        legend.position = "top")
```

```
# Confusion matrix heatmap
cm_table <- conf_mat$table
cm_df <- as.data.frame(cm_table)
colnames(cm_df) <- c("Prediction", "Reference", "Freq")

ggplot(cm_df, aes(x = Reference, y = Prediction, fill = Freq)) +
  geom_tile(color = "black") +
  geom_text(aes(label = Freq), color = "white", size = 6) +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Confusion Matrix Heatmap",
       x = "Reference (Real Habitat)",
       y = "Prediction (Satellite Habitat)") +
  theme_minimal() +
  theme(text = element_text(size = 14))
```

10. Comparing Field Counts and GPS Data of Lesser Kestrels

10.1. Prepare the input data:

- The dataset must include two columns:

```
number_of_kestrels_field
number_of_kestrels_gps
```

10.2. Calculate the Pearson correlation coefficient between GPS detections and field observations:

```
cor_result <- cor.test(
  data$number_of_kestrels_gps,
  data$number_of_kestrels_field,
  method = "pearson"
)
print(cor_result)
```

10.3. Fit a linear regression model to quantify the relationship:

```
model <- lm(number_of_kestrels_field ~ number_of_kestrels_gps, data = data)
summary(model)
```

10.4. Plot the relationship between GPS detections and field counts:

```
plot(
  data$number_of_kestrels_gps,
  data$number_of_kestrels_field,
  main = "Relationship between GPS data and field counts",
  xlab = "Kestrels detected by GPS",
  ylab = "Kestrels observed in field",
  pch = 19,
  col = "darkblue",
  cex.lab = 1.1,
  cex.main = 1.3,
  cex.axis = 1
)
```

10.5. Add the regression line in red:

```
abline(model, col = "red", lwd = 2)
```