Package: famexploreR (via r-universe)

October 29, 2024

Title Exploration of Threathened Flora field data

Version 1.0.0

Description A shiny app that explores data from Threathened Flora. It uses customized forms to explore the data and generate several plots, tables.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.0)

LazyData true

Imports dplyr, forcats, ggdist, gghalves, ggtern, ggplot2, glue, lubridate, purrr, readODS, readxl, rlang, sf, shiny, stats, stringr, tibble, tidyr, vegan, bookdown, bslib, bsicons, mapSpain, leaflet, formattable, kableExtra, flextable, readr,

Suggests rintrojs

Config/Needs/website rmarkdown

URL https://ajpelu.github.io/famexploreR/

Repository https://ajpelu.r-universe.dev

RemoteUrl https://github.com/ajpelu/famexploreR

RemoteRef HEAD

RemoteSha 2fe0599489ac73a50d7432a06beb024958827868

Contents

`biometryPlot` *Create Biometry Plot*

Description

This function generates a biometry plot based on the provided data frame. It visualizes the measurements of height, major diameter, and minor diameter for a specific species.

Usage

```
biometryPlot(x, base_size, axis_text_size = 24, ...)
```

Arguments

`x`	A data frame containing biometric measurements for a species.
`base_size`	The base_size for the plot
`axis_text_size`	The size of the axis text
`...`	others ggplot parameters

Details

The function takes a data frame `x` containing the following columns:

- `especie_code`: The species code.
- `id_individuo`: The individual identifier.
- `altura_cm`: Height in centimeters.
- `dmayor_cm`: Major diameter in centimeters.
- `dmenor_cm`: Minor diameter in centimeters.

The function creates a boxplot for each measurement type and uses custom colors. It also includes half-eye plots and half-point plots.

Value

A biometry plot visualizing height, major diameter, and minor diameter.

`biometryStat` *Calculate Biometry Statistics*

Description

This function calculates various statistics (mean, standard deviation, standard error, minimum, and maximum) for specified biometric variables in a data frame.

Usage

```
biometryStat(x, variables = c("altura_cm", "dmayor_cm", "dmenor_cm"))
```

Arguments

`x`	A data frame containing biometric data.
`variables`	A character vector specifying the names of the biometric variables for which statistics will be calculated. Default is c('altura_cm', 'dmayor_cm', 'dmenor_cm').

Value

A data frame with columns for each statistic (mean, sd, se, min, max) and a corresponding variable column.

`computeFlowering` *Compute Summary Statistics for Flowering Variables*

Description

This function computes summary statistics for specified flowering variables in a given data frame.

Usage

```
computeFlowering(x, var_interest)
```

Arguments

`x`	A data frame containing the variables of interest.
`var_interest`	A character vector of variable names to be summarized.

Details

This function calculates the following summary statistics for each specified variable:

- **variable**: The name of the variable.
- **n_ind**: The number of individuals with non-zero values.
- **pct_ind**: The percentage of individuals with non-zero values.
- **mean_count**: The mean value of the variable for individuals with non-zero values.
- **sd_count**: The standard deviation of the variable for individuals with non-zero values.
- **se_count**: The standard error of the mean for the variable.

Value

A data frame containing summary statistics for the specified variables.

Examples

```
## Not run:
data <- data.frame(
  n_flores = c(0, 2, 3, 0, 5),
  n_frutos = c(0, 0, 4, 0, 6)
)
count_columns <- c("n_flores", "n_frutos")
result_summary <- computeFlowering(data, count_columns)

## End(Not run)
```

---

diversityCommunity *Diversity Indices for Plant Communities*

Description

This function computes diversity indices for plant communities based on the specified method.

Usage

```
diversityCommunity(x)
```

Arguments

x	A list containing data frames

Details

The function first extracts the method used for data collection from the 'datos_generales' data frame. Based on the method, it selects the corresponding plant community data frame and calculates various diversity indices.

Value

A data frame with diversity indices for plant communities.

`herbivory` *Calculate Herbivory Metrics and Create a Plot*

Description

This function calculates herbivory metrics from a given dataset and creates a corresponding ggplot.

Usage

```
herbivory(
  data,
  bar_color = "blue",
  point_fill = "green",
  point_color = "black",
  point_alpha = 0.9
)
```

Arguments

`data`	A data frame containing the herbivory data with columns including 'comido_pct', 'id_individuo', and other relevant variables.
`bar_color`	The color for the bar in the ggplot. Default is blue.
`point_fill`	The fill color for points in the ggplot. Default is green.
`point_color`	The outline color for points in the ggplot. Default is black.
`point_alpha`	The alpha (transparency) for points in the ggplot. Default is 0.9.

Value

A list containing:

- 'damage': A tibble with herbivory metrics.
- 'plot_damage': A ggplot object displaying herbivory metrics.

hojas_validas *hojas_validas Dataset*

Description

This dataset contains multiple data frames related to various aspects of a study. It includes data on general information, soil, humidity and temperature, excrement observations, focal species, herbivory, neighborhood, vegetation coverage, vegetation contacts, taxonomy, treatment, fencing, state of the fencing, and coordinate reference system information.

Usage

```
hojas_validas
```

Format

A data frame containing the names of the sheets of the forms

launch_famexplorer *Run the famexploreR Shiny Application*

Description

Launches the famexploreR app

Usage

```
launch_famexplorer()
```

neighborAbundance_stats *Calculate Neighbor Abundance Statistics*

Description

This function calculates various neighbor abundance statistics based on input data.

Usage

```
neighborAbundance_stats(data, units = c("m2", "dm2", "cm2"), focal_sp)
```

Arguments

data	A data frame containing relevant columns, including "individuo" (individual ID), "diam_muestreo_vecindad_cm" (neighborhood sampling diameter in centimeters), "especie_vecina" (neighboring species), and "n_vecino" (number of neighbors).
units	A character vector specifying the units for area calculation. Options are "m2" (square meters), "dm2" (square decimeters), and "cm2" (square centimeters).
focal_sp	A character string specifying the focal species.

Value

A list containing two data frames:

output A data frame with individual-level neighbor abundance statistics.

output_summary A data frame with summarized neighbor abundance statistics.

---

neighborSpecies_stats *Summarizes neighbor species data*

Description

This function takes a data frame or tibble and groups it by neighbor species (`especie_vecina`). For each neighbor species, the function calculates the number of plots where each species has been recorded, as well as several statistics for the neighbor species: the mean, standard error, minimum, and maximum abundance of the neighbor species in the plots where it is present.

Usage

```
neighborSpecies_stats(data)
```

Arguments

data	A data frame or tibble containing the data to be summarized.

Value

A tibble with the following columns:

- especie_vecina: The grouping variable (neighbor species).
- ab_mean: The mean abundance of the neighbor species.
- ab_se: The standard error of the mean abundance.
- ab_min: The minimum abundance of the neighbor species.
- ab_max: The maximum abundance of the neighbor species.
- present_at: The count of plots where each neighbor species is recorded.
- present_at_per: The percentage of the sampled plots (`individuo`) where each neighbor species is recorded.

plotCommunity *Plot Plant Community Data*

Description

This function generates a bar plot to visualize plant community data based on the specified method.

Usage

```
plotCommunity(x, axis_text_size = 16, axis_title_size = 17, ...)
```

Arguments

x	A list containing data frames for different aspects of the study.
axis_text_size	The size of the axis text. Default value=16
axis_title_size	The size of the axis title. Default value=17
...	others ggplot parameters

Details

The function first extracts the method used for data collection from the 'datos_generales' data frame. Based on the method, it selects the corresponding plant community data frame and creates a bar plot to visualize the coverage of plant species.

Value

A bar plot visualizing plant community data.

plotFlowering *Create a bar plot with error bars for flowering/fructification data.*

Description

This function generates a bar plot with error bars for flowering data, allowing you to specify whether to use standard error ("se") or standard deviation ("sd") error bars.

Usage

```
plotFlowering(x, error = "se", bar_color = "blue", ...)
```

Arguments

x	A data frame containing the flowering data.
error	The type of error bars to use. Should be "se" (standard error) or "sd" (standard deviation). Default is "se".
bar_color	The color for the bars in the plot. Default is "blue".
...	others ggplot parameters

Value

A ggplot2 object representing the bar plot with error bars.

See Also

ggplot2, geom_bar, geom_errorbar

---

prepareGeo *Prepare Geo Spatial Data*

Description

This function takes a data frame containing information about geographic coordinates and prepares it for spatial analysis by converting it into a Simple Features (sf) object.

Usage

```
prepareGeo(x)
```

Arguments

x	A data frame containing at least the columns 'campo' and 'valor', where 'campo' specifies the type of information (e.g., 'crs', 'coord_x', 'coord_y', 'elevacion'), and 'valor' contains the corresponding values.

Details

This function creates an sf object with spatial coordinates and attributes, from an input data frame thata contain specific columns for 'campo' and 'valor'.

Value

A Simple Features (sf) object with spatial coordinates and attributes.

prepare Popup *preparaPopup function*

Description

This function takes a tibble `x` and a vector of field names `mdfields` as input and prepares an HTML popup content based on the specified fields.

Usage

```
prepare Popup(
  x,
  mdfields = c("especie focal", "localidad", "site", "reference", "poblacion",
    "tratamiento", "elevacion", "fecha")
)
```

Arguments

`x`	A tibble containing data with columns 'campo' and 'valor'.
`mdfields`	A vector of field names specifying which fields to include in the popup content.

Value

A character string containing HTML-formatted popup content.

readAllsheets *Read data from all sheets of an uploaded file.*

Description

This function reads data from all sheets of an uploaded file in ODS or XLSX format and returns them in a named list.

Usage

```
readAllsheets(upload_path, valid_sheets)
```

Arguments

`upload_path`	Character string specifying the path to the uploaded file.
`valid_sheets`	Character vector specifying the names of sheets that are expected to be in the uploaded file.

Value

A named list containing data from all sheets in the uploaded file, with sheet names as list names.

summarizeSoil *Summarize Soil Data*

Description

This function takes a tibble containing soil data, performs data summarization, and returns a formatted and summarized tibble.

Usage

```
summarizeSoil(x)
```

Arguments

x	A tibble containing soil data.

Details

This function performs the following steps:

1. Removes rows with missing values.
2. Excludes certain columns from the analysis.
3. Calculates mean, standard deviation, and standard error for numeric columns.
4. Pivots the data into a long format.
5. Renames columns for clarity.
6. Pivots the data back to a wider format.
7. Rounds numeric columns to three decimal places.

Value

A tibble with summarized soil data.

ternaryPlot *Generate a Ternary Plot with Customizable Axis Labels*

Description

This function generates a ternary plot using the ggtern package with customizable axis labels. The function allows specifying the variable names for the x, y, and z axes. It also capitalizes the first letter of each variable name for axis labels.

Usage

```
ternaryPlot(data, xvar, yvar, zvar, bsize, point_size, ...)
```

Arguments

data	A data frame containing the data to be plotted.
xvar	The name of the variable to be plotted in the x-axis of the ternary plot.
yvar	The name of the variable to be plotted in the y-axis of the ternary plot.
zvar	The name of the variable to be plotted in the z-axis of the ternary plot.
bsize	The base font size for the plot.
point_size	The size of the points.
...	others ggplot parameters

Details

This function creates a ternary plot, where data points are represented by points in a triangular coordinate system. The xvar, yvar, and zvar arguments allow you to specify which variables from the data argument should be used for each axis. The function also capitalizes the first letter of each variable name for use in the axis labels. You can adjust the base font size (bsize) for the plot to control the text size.

Value

A ggtern plot object displaying the ternary plot.

See Also

ggtern for more information on creating ternary plots using ggtern.

Examples

```
## Not run:
# Example usage of the custom function with customized variable names
data <- data.frame(
  arena = c(0.4, 0.3, 0.2, 0.1),
  arcilla = c(0.3, 0.4, 0.2, 0.1),
  limo = c(0.3, 0.3, 0.4, 0.1)
)

ternaryPlot(data, xvar = 'arena', yvar = 'arcilla',
zvar = 'limo')

## End(Not run)
```

`vecindadPlot` *Generate Vecindad Plot*

Description

This function generates a bar plot with error bars that visualizes the abundance of neighboring species.

Usage

```
vecindadPlot(x, axis_text_size = 16, axis_title_size = 17, ...)
```

Arguments

`x`	A data frame containing data for plotting. • `especie_vecina`: The neighboring species. • `ab_mean`: The mean abundance of the neighboring species. • `ab_se`: The standard error of the mean abundance.
`axis_text_size`	The size of the axis text. Default value=16
`axis_title_size`	The size of the axis title. Default value=17
`...`	others ggplot parameters

Value

A bar plot with error bars.

Index