Package ‘commecometrics’

July 22, 2025

Title Ecometric Models of Trait–Environment Relationships at the Community Level

Version 1.0.0

Description Provides a framework for modeling relationships between functional traits and both quantitative and qualitative environmental variables at the community level. It includes tools for trait binning, likelihood-based environmental estimation, model evaluation, fossil projection into modern ecometric space, and result visualization. For more details see Vermillion et al. (2018) <doi:10.1007/978-3-319-94265-0_17>, Polly et al. (2011) <doi:10.1098/rspb.2010.2233> and Polly and Head (2015) <doi:10.1017/S1089332600002953>.

BugReports `https://github.com/mariahm1995/commecometrics/issues`

License MIT + file LICENSE

Encoding UTF-8

Imports ggplot2, leaflet, dplyr, purrr, raster, rnaturalearth, sf, tibble, viridis

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Depends R (>= 3.5)

LazyData true

LazyDataCompression xz

Config/testthat/edition 3

URL `https://github.com/mariahm1995/commecometrics`

NeedsCompilation no

Author Maria A. Hurtado-Materon [cre, aut],
Leila Siciliano-Martina [aut],
A. Michelle Lawing [cph, aut]

Maintainer Maria A. Hurtado-Materon `<maria.h.m_1995@tamu.edu>`

Repository CRAN

Date/Publication 2025-07-22 12:00:07 UTC

Contents

`commecometrics-utils` *Internal utilities and global variables*

Description

Internal utilities and variable declarations to support NSE and ggplot2 piping.

`ecometric_model` *Run an ecometric model for quantitative environmental variables*

Description

Builds an ecometric trait space for quantitative environmental variables, estimating environmental values of each category at each trait bin combination. Also calculates anomalies based on observed values for each point.

Usage

```
ecometric_model(
  points_df,
  env_var = "env_var",
  transform_fun = function(x) x,
  inv_transform_fun = function(x) x,
  grid_bins_1 = NULL,
  grid_bins_2 = NULL,
  min_species = 3
)
```

Arguments

`points_df`	Output first element of the list from `summarize_traits_by_point()`. A data frame with columns: `summ_trait_1`, `summ_trait_2`, `count_trait`, and the environmental variable.
`env_var`	Name of the environmental variable (e.g., "precip").
`transform_fun`	Optional transformation function for environmental variable (e.g., `log(x + 1)`).
`inv_transform_fun`	Optional inverse transformation for environmental variable (e.g., `exp(x) - 1`).
`grid_bins_1`	Number of bins for the first trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`grid_bins_2`	Number of bins for the second trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`min_species`	Minimum number of species per point (default = 3).

Value

A list containing:

`points_df`	Filtered input data frame with the following added columns: **env_trans** Transformed environmental variable (if a transformation function is used). **bin_1** Bin assignment code for mean trait value. **bin_2** Bin assignment code for standard deviation of trait. **env_est** Predicted (maximum likelihood) environmental value on transformed scale. **env_anom** Difference between observed and predicted environmental values (transformed scale). **env_est_UN** Inverse-transformed predicted value (if `inv_transform_fun` is provided). **env_anom_UN** Inverse-transformed anomaly value (if `inv_transform_fun` is provided).
`eco_space`	Raster-format data frame representing trait space bins with estimated environmental values.
`model`	Linear model object (`lm`) relating predicted environmental values to observed environmental values (transformed scale when used).
`correlation`	Output from `cor.test`, reporting the Pearson correlation between predicted and observed environmental values (transformed scale when used).
`diagnostics`	Summary stats about bin usage and data coverage.
`settings`	Metadata including the modeled trait and transformation functions.
`prediction_accuracy`	Overall percentage of correct predictions.

Examples

```
# Load internal dataset
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")

# Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Fit an ecometric model using annual precipitation (BIO12)
modelResult <- ecometric_model(
  points_df = traitsByPoint$points,
  env_var = "precip",
  transform_fun = function(x) log(x + 1),
  inv_transform_fun = function(x) exp(x) - 1,
  min_species = 3
)

# View correlation between predicted and observed values
print(modelResult$correlation)

# View summary of the linear model fit
summary(modelResult$model)
```

`ecometric_model_qual` *Run an ecometric model for qualitative environmental variables*

Description

Builds an ecometric trait space for qualitative environmental variables, estimating the most probable category and the probability of each category at each trait bin combination. Also calculates prediction accuracy and anomalies for each point.

Usage

```
ecometric_model_qual(
  points_df,
  category_col,
  grid_bins_1 = NULL,
  grid_bins_2 = NULL,
```

```
  min_species = 3
)
```

Arguments

`points_df`	Output first element of the list from `summarize_traits_by_point()`. A data frame with columns: `summ_trait_1`, `summ_trait_2`, `count_trait`, and the environmental variable.
`category_col`	Name of the column containing the categorical trait.
`grid_bins_1`	Number of bins for the first trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`grid_bins_2`	Number of bins for the second trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`min_species`	Minimum number of species with trait data per point (default = 3).

Value

A list containing:

`points_df`	Filtered input data frame with the following added columns: **bin_1** Bin assignment code for first trait axis. **bin_2** Bin assignment code for second trait axis. **prob_category** Estimated probability of each environmental category per trait bin (e.g., `prob_1`, `prob_2`, etc.). **observed_probability** Probability assigned to the observed category for each point. **predicted_probability** Probability assigned to the predicted (most likely) category for each point. **predicted_category** Predicted environmental category for each point. **correct_prediction** Indicator for whether the predicted category matches the observed category (`"Yes"` or `"No"`). **env_anom** Difference between predicted and observed category probabilities.
`eco_space`	Raster-format data frame representing trait space bins with estimated environmental categories.
`diagnostics`	Summary stats about bin usage and data coverage.
`settings`	Metadata including the modeled trait.
`prediction_accuracy`	Overall percentage of correct predictions.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")
```

```
# Step 1: Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Step 2: Run ecometric model using land cover class as qualitative variable
modelResult <- ecometric_model_qual(
  points_df = traitsByPoint$points,
  category_col = "vegetation",
  min_species = 3
)

# View the percentage of correctly predicted categories
print(modelResult$prediction_accuracy)
```

`ecometric_space` *Plot ecometric space for quantitative environmental variables*

Description

Visualizes the ecometric space for quantitative environmental variables based on the output from `ecometric_model()`.

Usage

```
ecometric_space(
  model_out,
  env_name = "env_var",
  fossil_data = NULL,
  fossil_color = "#000000",
  modern_color = "#bc4749",
  palette = c("#bc6c25", "#fefae0", "#606c38"),
  x_label = "Summary metric 1",
  y_label = "Summary metric 2"
)
```

Arguments

`model_out`	Output from `ecometric_model()`, containing environmental estimates in trait space.
`env_name`	Name to display for the environmental variable (used in the legend title).

`fossil_data`	Optional. Output from `reconstruct_env()`.
`fossil_color`	Outline color for fossil data bins (default: `"#000000"`).
`modern_color`	Outline color for modern data bins (default: `"#bc4749"`).
`palette`	Vector of colors to use for the gradient scale representing environmental values.
`x_label`	Label for the x-axis in the output plots (default: "Summary metric 1").
`y_label`	Label for the y-axis in the output plots (default: "Summary metric 2").

Value

A ggplot2 object visualizing the ecometric trait-environment surface.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")
data("fossils", package = "commecometrics")

# Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Run ecometric model
ecoModel <- ecometric_model(
  points_df = traitsByPoint$points,
  env_var = "precip",
  transform_fun = function(x) log(x + 1),
  inv_transform_fun = function(x) exp(x) - 1,
  min_species = 3
)

# Reconstruct environments for fossil sites
recon <- reconstruct_env(
  fossildata = fossils,
  model_out = ecoModel,
  match_nearest = TRUE,
  fossil_lon = "Long",
  fossil_lat = "Lat",
  modern_id = "ID",
  modern_lon = "Longitude",
  modern_lat = "Latitude"
)
```

```
# Plot the ecometric trait-environment space
ecometricPlot <- ecometric_space(
  model_out = ecoModel,
  env_name = "Precipitation (log mm)",
  fossil_data = recon
)

# Display plot
print(ecometricPlot)
```

`ecometric_space_qual` *Plot ecometric space for qualitative environmental variables*

Description

Visualizes the predicted ecometric space (predicted category) and probability maps for each category based on the output from `ecometric_model_qualitative()`.

Usage

```
ecometric_space_qual(
  model_out,
  category_labels = NULL,
  palette = NULL,
  fossil_data = NULL,
  fossil_color = "#000000",
  modern_color = "#bc4749",
  x_label = "Summary metric 1",
  y_label = "Summary metric 2"
)
```

Arguments

`model_out`	Output from `ecometric_model_qualitative()`, containing environmental estimates in trait space.
`category_labels`	Optional named vector for category labels (used in the legend title). If `NULL`, the unique strings in the predicted category column (`env_est`) will be used as-is.
`palette`	Optional color vector for categories (must match number of categories).
`fossil_data`	Optional. Output from `reconstruct_env_qual()`.
`fossil_color`	Outline color for fossil data bins (default = "#000000").
`modern_color`	Outline color for modern data bins (default: `"#bc4749"`).
`x_label`	Label for the x-axis in the output plots (default: "Summary metric 1").
`y_label`	Label for the y-axis in the output plots (default: "Summary metric 2").

Value

A list containing:

`ecometric_space_plot`	ggplot showing the predicted category across trait space.
`probability_maps`	List of ggplots showing probability surfaces for each category.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")
data("fossils", package = "commecometrics")

# Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Run ecometric model for qualitative variable
modelResult <- ecometric_model_qual(
  points_df = traitsByPoint$points,
  category_col = "vegetation",
  min_species = 3
)

# Reconstruct fossil environmental categories
reconQual <- reconstruct_env_qual(
  fossildata = fossils,
  model_out = modelResult,
  match_nearest = TRUE,
  fossil_lon = "Long",
  fossil_lat = "Lat",
  modern_id = "ID",
  modern_lon = "Longitude",
  modern_lat = "Latitude"
)

# Plot qualitative ecometric space
ecoPlotQual <- ecometric_space_qual(
  model_out = modelResult,
  fossil_data = reconQual
)
```

```
# Display predicted category map
print(ecoPlotQual$ecometric_space_plot)

# Display one of the probability maps
print(ecoPlotQual$probability_maps[["1"]])
```

`fossils` *Fossil trait data for projection onto ecometric space*

Description

A dataset of fossil sites with estimated trait distribution and geographic coordinates, used to project extinct communities onto modern ecometric space.

Usage

```
fossils
```

Format

A data frame with the following columns:

Site	Unique identifier for the fossil sample
fossil_summ_trait_1	Estimated mean of relative blade length for the fossil site
fossil_summ_trait_2	Estimated sd of relative blade length for the fossil site
Long	Longitude coordinate (decimal degrees)
Lat	Latitude coordinate (decimal degrees)

Source

Siciliano-Martina et al. (2024). *Ecology and Evolution*, 14(10), e70214.

`geoPoints` *Example climate sampling points*

Description

A subset of 100 global sampling points with associated bioclimatic and vegetation variables. All points overlap with species ranges in the `spRanges` dataset.

Usage

```
geoPoints
```

Format

A data frame with the following columns:

ID	Unique identifier for each point
Longitude	Longitude coordinate (decimal degrees)
Latitude	Latitude coordinate (decimal degrees)
temp	Mean annual temperature (°C × 10)
precip	Annual precipitation (mm)
vegetation	Vegetation units (integer code)

Source

Derived from Siciliano-Martina et al. (2024), filtered for overlap with IUCN polygons.

`inspect_point_species` *Inspect overlapping species at sampling points*

Description

Creates an interactive map to verify species overlap at selected points.

Usage

```
inspect_point_species(
  traits_summary,
  point_ids = NULL,
  n_random = 10,
  lon_col = "Longitude",
  lat_col = "Latitude",
  ID_col = "ID",
  min_species_valid = 3,
  env_var = NULL
)
```

Arguments

`traits_summary`	A list output from `summarize_traits_by_point()`, containing summarized trait values (`$points`) and species overlaps (`$overlap`).
`point_ids`	Optional. A vector of specific point IDs to inspect. If NULL, selects `n_random` points at random.
`n_random`	Number of random points to inspect if `point_ids` not provided (default = 10).
`lon_col`	Name of the longitude column in `points` (default = "Longitude").
`lat_col`	Name of the latitude column in `points` (default = "Latitude").
`ID_col`	Name of the ID column in `points` (default = "ID").

`min_species_valid`	Minimum number of species with trait data to consider a point valid (default = 3).
`env_var`	Optional. Name of the environmental variable column in `points` to include in popup.

Value

An interactive leaflet map showing selected points with species list popups.

Examples

```
# Load sample data from the package
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")

# Summarize traits at points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Visualize a random sample of 10 points
inspect_point_species(
  traits_summary = traitsByPoint,
  n_random = 10,
  min_species_valid = 3
)
```

`optimal_bins` *Determine optimal number of bins using Scott's Rule*

Description

Calculates the optimal number of bins for a numeric vector based on Scott's rule.

Usage

```
optimal_bins(x)
```

Arguments

`x`	Numeric vector.

Value

Integer representing the optimal number of bins.

Examples

```
# Simple example
# Example with normally distributed data
optimal_bins(rnorm(100))
```

`reconstruct_env` *Reconstruct past environmental conditions using ecometric models*

Description

Uses fossil community trait summaries to reconstruct past environmental conditions by projecting them onto a binned ecometric trait space built from modern data. Optionally, it also assigns each fossil point to the nearest modern sampling site to retrieve observed environmental data.

Usage

```
reconstruct_env(
  fossildata,
  model_out,
  inv_transform = NULL,
  ci = 0.05,
  match_nearest = TRUE,
  fossil_lon = NULL,
  fossil_lat = NULL,
  modern_id = NULL,
  modern_lon = NULL,
  modern_lat = NULL,
  crs_proj = 4326
)
```

Arguments

`fossildata`	A data frame containing fossil trait summaries per fossil site. Must include columns corresponding to the same two summary metrics used for modern communities, using the column names specified by `fossil_summ_trait_1` and `fossil_summ_trait_2`.
`model_out`	Output list from `run_ecometric_model()`, containing modern data, diagnostics, and model settings.
`inv_transform`	A function to back-transform environmental estimates to the original scale. Default is `exp(x) - 1`. If `NULL`, the inverse transform stored in `model_out` is used if available.
`ci`	The width of the interval to calculate around the maximum likelihood estimate (default = 0.05).

`match_nearest`	Logical; if TRUE, the function matches each fossil to its nearest modern point based on coordinates (default = TRUE).
`fossil_lon`	Name of the longitude column in `fossildata`. Required if `match_nearest = TRUE`.
`fossil_lat`	Name of the latitude column in `fossildata`. Required if `match_nearest = TRUE`.
`modern_id`	Name of the unique ID column in modern points (e.g., "GlobalID").
`modern_lon`	Name of the longitude column in modern points. Required if `match_nearest = TRUE`.
`modern_lat`	Name of the latitude column in modern points. Required if `match_nearest = TRUE`.
`crs_proj`	Coordinate reference system to use when converting fossil and modern data to sf format (default = EPSG:4326).

Value

A data frame (`fossildata`) with reconstructed environmental values and optional nearest modern point data. Includes the following additional columns:

fossil_bin_1 Assigned bin number for the first trait axis (based on first summary metric of trait distribution of fossil communities).

fossil_bin_2 Assigned bin number for the second trait axis (based on second summary metric of trait distribution of fossil communities).

fossil_env_est Maximum likelihood estimate of the environmental variable (on transformed scale if applicable).

fossil_minlimit Lower bound of the confidence interval around the environmental estimate (transformed scale).

fossil_maxlimit Upper bound of the confidence interval around the environmental estimate (transformed scale).

fossil_env_est_UN (Optional) Inverse-transformed environmental estimate, on the original scale.

fossil_minlimit_UN (Optional) Inverse-transformed lower bound of the confidence interval.

fossil_maxlimit_UN (Optional) Inverse-transformed upper bound of the confidence interval.

nearest_modern_point (Optional) ID of the nearest modern sampling point (if `match_nearest = TRUE`).

... Additional columns from the matched modern site if `match_nearest = TRUE` (e.g., observed environmental values).

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")
data("fossils", package = "commecometrics")
```

```
# Step 1: Summarize modern trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Step 2: Run an ecometric model with BIO12 (precipitation)
ecoModel <- ecometric_model(
  points_df = traitsByPoint$points,
  env_var = "precip",
  transform_fun = function(x) log(x + 1),
  inv_transform_fun = function(x) exp(x) - 1,
  min_species = 3
)

# Step 3: Reconstruct fossil environments
recon <- reconstruct_env(
  fossildata = fossils,
  model_out = ecoModel,
  match_nearest = TRUE,
  fossil_lon = "Long",
  fossil_lat = "Lat",
  modern_id = "ID",
  modern_lon = "Longitude",
  modern_lat = "Latitude"
)
```

`reconstruct_env_qual` *Reconstruct past qualitative environmental categories using ecometric models*

Description

Uses fossil community trait summaries to reconstruct the most likely environmental category by projecting them onto a qualitative ecometric space built from modern data. Optionally, it assigns each fossil point to the nearest modern sampling point.

Usage

```
reconstruct_env_qual(
  fossildata,
  model_out,
  match_nearest = TRUE,
```

```
  fossil_lon = NULL,
  fossil_lat = NULL,
  modern_id = NULL,
  modern_lon = NULL,
  modern_lat = NULL,
  crs_proj = 4326
)
```

Arguments

`fossildata`	A data frame containing fossil trait summaries per fossil site. Must include columns corresponding to the same two summary metrics used for modern communities, using the column names specified by `fossil_summ_trait_1` and `fossil_summ_trait_2`.
`model_out`	Output list from `ecometric_model_qual()`, containing modern data, diagnostics, and model settings.
`match_nearest`	Logical; if TRUE, matches each fossil to the nearest modern point (default = TRUE).
`fossil_lon`	Name of the longitude column in `fossildata`. Required if `match_nearest` = `TRUE`.
`fossil_lat`	Name of the latitude column in `fossildata`. Required if `match_nearest` = `TRUE`.
`modern_id`	Name of the unique ID column in modern points (optional for metadata merging).
`modern_lon`	Name of the longitude column in modern points. Required if `match_nearest` = `TRUE`.
`modern_lat`	Name of the latitude column in modern points. Required if `match_nearest` = `TRUE`.
`crs_proj`	Coordinate reference system for sf operations (default = EPSG:4326).

Value

A data frame (`fossildata`) updated with:

fossil_bin_1 Assigned bin number for the first trait axis (based on first summary metric of trait distribution of fossil communities).

fossil_bin_2 Assigned bin number for the second trait axis (based on second summary metric of trait distribution of fossil communities).

fossil_env_est Predicted environmental category based on trait bin.

fossil_prob_* Probability of each environmental category for the assigned bin.

nearest_modern_point (Optional) ID of the nearest modern sampling point (if `match_nearest` = `TRUE`).

... Additional columns from the matched modern site if `match_nearest` = `TRUE`.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")
data("fossils", package = "commecometrics")

# Step 1: Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Step 2: Run a qualitative ecometric model (e.g., land cover class)
ecoModelQual <- ecometric_model_qual(
  points_df = traitsByPoint$points,
  category_col = "vegetation",
  min_species = 3
)

# Step 3: Reconstruct qualitative environments for fossil data
reconQual <- reconstruct_env_qual(
  fossildata = fossils,
  model_out = ecoModelQual,
  match_nearest = TRUE,
  fossil_lon = "Long",
  fossil_lat = "Lat",
  modern_id = "ID",
  modern_lon = "Longitude",
  modern_lat = "Latitude"
)
```

`sensitivity_analysis` *Perform sensitivity analysis on ecometric models (quantitative environmental variables)*

Description

Evaluates how varying sample sizes affect the performance of ecometric models, focusing on two aspects:

- **Sensitivity (internal consistency)**: How accurately the model predicts environmental conditions on the same data it was trained on.

- **Transferability (external applicability)**: How well the model performs on unseen data.

It tests different sample sizes by resampling the data multiple times (bootstrap iterations), training an ecometric model on each subset, and evaluating prediction error and correlation.

Usage

```
sensitivity_analysis(
  points_df,
  env_var,
  sample_sizes,
  iterations = 20,
  test_split = 0.2,
  grid_bins_1 = NULL,
  grid_bins_2 = NULL,
  transform_fun = NULL,
  parallel = TRUE,
  n_cores = parallel::detectCores() - 1
)
```

Arguments

`points_df`	Output first element of the list from `summarize_traits_by_point()`. A data frame with columns: `summ_trait_1`, `summ_trait_2`, `count_trait`, and the environmental variable.
`env_var`	Name of the environmental variable column in points_df (e.g., "precip").
`sample_sizes`	Numeric vector specifying the number of communities (sampling points) to evaluate in the sensitivity analysis. For each value, a random subset of the data of that size is drawn without replacement and then split into training and testing sets using the proportion defined by `test_split` (default is 80% training, 20% testing). All values in `sample_sizes` must be less than or equal to the number of rows in `points_df`, and large enough to allow splitting based on `test_split` (i.e., both the training and testing sets must contain at 30 communities).
`iterations`	Number of bootstrap iterations per sample size (default: 20).
`test_split`	Proportion of data to use for testing (default: 0.2).
`grid_bins_1`	Number of bins for the first trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`grid_bins_2`	Number of bins for the second trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`transform_fun`	Function to transform the environmental variable (default: NULL = no transformation).
`parallel`	Logical; whether to use parallel processing (default: TRUE).
`n_cores`	Number of cores to use for parallel processing (default: parallel::detectCores() - 1).

Details

Four base R plots are generated to visualize model performance as a function of sample size:

1. **Training correlation vs. Sample size:** Shows how well the model fits training data.
2. **Testing correlation vs. Sample size:** Shows generalizability to new data.
3. **Training mean anomaly vs. Sample size:** Shows average prediction error on training data.
4. **Testing mean anomaly vs. Sample size:** Shows average prediction error on test data.

Parallel processing is supported to speed up the analysis.

Value

A list containing:

`combined_results`	A data frame with mean absolute anomalies and correlations for each sample size and iteration.
`summary_results`	A data frame summarizing the mean anomalies and correlations across sample sizes.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")

# Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Run sensitivity analysis using annual precipitation
sensitivityResults <- sensitivity_analysis(
  points_df = traitsByPoint$points,
  env_var = "precip",
  sample_sizes = seq(40, 90, 10),
  iterations = 5,
  transform_fun = function(x) log(x + 1),
  parallel = FALSE  # Set to TRUE for faster performance on multicore machines
)

# View results
head(sensitivityResults$summary_results)
```

`sensitivity_analysis_qual` *Perform sensitivity analysis on ecometric models (qualitative environmental variables)*

Description

Evaluates how varying sample sizes affect the performance of ecometric models, focusing on two aspects:

- **Sensitivity (internal consistency)**: How accurately the model predicts environmental conditions on the same data it was trained on.
- **Transferability (external applicability)**: How well the model performs on unseen data.

It tests different sample sizes by resampling the data multiple times (bootstrap iterations), training an ecometric model on each subset, and evaluating prediction error and correlation.

Usage

```
sensitivity_analysis_qual(
  points_df,
  category_col,
  sample_sizes,
  iterations = 20,
  test_split = 0.2,
  grid_bins_1 = NULL,
  grid_bins_2 = NULL,
  parallel = TRUE,
  n_cores = parallel::detectCores() - 1
)
```

Arguments

`points_df`	Output first element of the list from `summarize_traits_by_point()`. A data frame with columns: `summ_trait_1`, `summ_trait_2`, `count_trait`, and the environmental variable.
`category_col`	Name of the column containing the categorical trait.
`sample_sizes`	Numeric vector specifying the number of communities (sampling points) to evaluate in the sensitivity analysis. For each value, a random subset of the data of that size is drawn without replacement and then split into training and testing sets using the proportion defined by `test_split` (default is 80% training, 20% testing). All values in `sample_sizes` must be less than or equal to the number of rows in `points_df`, and large enough to allow splitting based on `test_split` (i.e., both the training and testing sets must contain at 30 communities).
`iterations`	Number of bootstrap iterations per sample size (default = 20).
`test_split`	Proportion of data to use for testing (default = 0.2).

`grid_bins_1`	Number of bins for the first trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`grid_bins_2`	Number of bins for the second trait axis. If `NULL` (default), the number is calculated automatically using Scott's rule via `optimal_bins()`.
`parallel`	Logical; whether to run iterations in parallel (default = TRUE).
`n_cores`	Number of cores for parallelization (default = detectCores() - 1).

Details

Two plots are generated:

1. **Training Accuracy vs. Sample size:** Reflects internal model consistency.
2. **Testing Accuracy vs. Sample size:** Reflects external model performance.

Parallel processing is supported to speed up the analysis.

Value

A list containing:

`combined_results`	All raw iteration results.
`summary_results`	Mean accuracy per sample size.

Examples

```
# Load internal data
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")

# Summarize trait values at sampling points
traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

# Run sensitivity analysis for dominant land cover class
sensitivityQual <- sensitivity_analysis_qual(
  points_df = traitsByPoint$points,
  category_col = "vegetation",
  sample_sizes = seq(40, 90, 10),
  iterations = 5,
  parallel = FALSE
)
```

```
# View results
head(sensitivityQual$summary_results)
```

`spRanges` *Species distribution polygons for five Carnivora species*

Description

A spatial dataset of species range polygons matching the species in the `traits` dataset.

Usage

```
spRanges
```

Format

An `sf` object with the following columns:

TaxonName Species name (matching the `traits` table)

geometry Polygon geometry representing species distribution

Source

Download from the IUCN Red List webpage (IUCN, 2025).

`summarize_traits_by_point` *Summarize trait distributions at sampling points with optional continent assignment*

Description

For each spatial sampling point, this function calculates two metrics specified by the user of a trait across all overlapping species polygons, and calculates richness. Optionally, it assigns each point to a continent using Natural Earth data.

Usage

```
summarize_traits_by_point(
  points_df,
  trait_df,
  species_polygons,
  comm_metric_1 = function(x) mean(x, na.rm = TRUE),
  comm_metric_2 = function(x) sd(x, na.rm = TRUE),
  trait_column = "trait_name",
  species_name_col = "sci_name",
  continent = FALSE,
  lon_col = "Longitude",
  lat_col = "Latitude",
  parallel = TRUE,
  n_cores = parallel::detectCores() - 1
)
```

Arguments

`points_df`	A data frame containing sampling points with columns for longitude and latitude.
`trait_df`	A data frame of trait data. Must include a column for species names (’TaxonName’) and the trait of interest (default = "trait_name").
`species_polygons`	An `sf` object containing species distribution polygons. Must include a species name column.
`comm_metric_1`	A function used to summarize the trait values across overlapping species. Defaults to `mean(x, na.rm = TRUE)`. The function must take a numeric vector as input and return a single numeric value. Can be replaced by any user-defined function, such as `max`, `median`, or a custom function.
`comm_metric_2`	A second function used to summarize trait values. Defaults to `sd(x, na.rm = TRUE)`. Works the same way as `summary_trait_1`.
`trait_column`	The name of the trait column in `trait_df` to summarize.
`species_name_col`	The name of the column in `species_polygons` that contains species names (default = "sci_name").
`continent`	Logical. If `TRUE`, assigns each sampling point to a continent using the Natural Earth shapefile via `rnaturalearth::ne_countries()`. If `FALSE` (default), no continent assignment is performed.
`lon_col`	Name of the longitude column in `points_df`. Default is ’Longitude’.
`lat_col`	Name of the latitude column in `points_df`. Default is ’Latitude’.
`parallel`	Logical; whether to parallelize the summarization step (default TRUE).
`n_cores`	Number of cores to use if parallelizing (default: detectCores() - 1).

Value

A list with two elements:

- **points** A data frame identical to `points_df` but with additional columns:
  - **summ_trait_1** Result of applying `metric_1` to the trait values of overlapping species (e.g., mean, max, median).
  - **summ_trait_2** Result of applying `metric_2` to the trait values of overlapping species (e.g., standard deviation, range).
  - **richness** Number of species overlapping the point (regardless of trait availability).
  - **count_trait** Number of species with non-missing trait values at the point.
  - **continent** (Optional) Continent name assigned from Natural Earth data, if `continent = TRUE`.
- **overlap** A list of character vectors, each containing the names of species whose distribution polygons overlap a given sampling point.

Examples

```
# Load sample data from the package
data("geoPoints", package = "commecometrics")
data("traits", package = "commecometrics")
data("spRanges", package = "commecometrics")

traitsByPoint <- summarize_traits_by_point(
  points_df = geoPoints,
  trait_df = traits,
  species_polygons = spRanges,
  trait_column = "RBL",
  species_name_col = "sci_name",
  continent = FALSE,
  parallel = FALSE
)

head(traitsByPoint$points)
```

`traits` *Relative blade length trait data for Carnivora*

Description

A dataset of relative blade length (RBL) values for five species in the order Carnivora. These species match those in the `spRanges` dataset.

Usage

```
traits
```

Format

A data frame with the following columns:

TaxonName Species name (binomial)

RBL Relative blade length (unitless ratio)

Source

Siciliano-Martina et al. (2024). *Ecology and Evolution*, 14(10), e70214.

Index