Package: nicheR (via r-universe)

June 17, 2026

Type Package

Title Ellipsoid-Based Virtual Niches and Visualization

Version 0.1.0

Description Provides a robust set of tools for researchers and modelers to construct and define virtual ecological niches using ellipsoid geometries. It enables the identification and extraction of suitable environmental areas, simulation of species occurrence points with various sampling strategies, and visualization of niche boundaries and simulated occurrences in both environmental and geographic space. Inspired by methodologies in 'NicheA' and the 'virtualspecies' R package, 'nicheR' aims to streamline the process of niche conceptualization and data generation for ecological studies. Methodological and theoretical foundations are described in Peterson et al. (2011, ISBN:9780691136882), Etherington et al. (2009) <doi:10.1111/j.1365-2699.2008.02041.x>, Qiao et al. (2015) <doi:10.1111/ecog.01961>, Nunez-Penichet et al. (2021) <doi:10.21425/F5FBG52142>, Cobos and Peterson (2022) <doi:10.17161/bi.v17i.15985>, Alkishe et al. (2022) <doi:10.5194/we-22-33-2022>, and Leroy et al. (2015) <doi:10.1111/ecog.01388>.

URL https://github.com/castanedaM/nicheR, https://castanedam.github.io/nicheR/

BugReports https://github.com/castanedaM/nicheR/issues

Depends R (>= 3.5)

Imports terra

Suggests rgl (>= 1.3), knitr, rmarkdown

VignetteBuilder knitr

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libproj-dev libsqlite3-dev

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

Date/Publication 2026-06-08 16:02:48 UTC

RemoteUrl https://github.com/castanedam/nicher

RemoteRef HEAD

RemoteSha 5bc29818a0d71e3e2b13177c73d88271876232a9

Contents

add_data *Add occurrence points or other data to an existing E-space plot*

Description

Adds points to an existing environmental space plot created with `plot_ellipsoid()`. Points can be plotted with a single color or colored by a continuous variable (e.g., suitability) using a color palette.

Usage

```
add_data(data, x, y,
                pts_col = "#000000",pts_alpha  = 1,
                col_layer = NULL,
                pal = hcl.colors(100, palette = "Viridis"), rev_pal = FALSE,
                pch = 1, cex = 1, bg_sample = NULL, ...)
```

Arguments

`data`	A data frame containing the points to plot. Must include columns matching `x` and `y`, and `col_layer` if provided.
`x`	Character. Name of the column to use as the x-axis variable.
`y`	Character. Name of the column to use as the y-axis variable.
`pts_col`	Character. Color for all points when `col_layer` is `NULL`. Default is `"#000000"` (black).
`pts_alpha`	Numeric in `[0, 1]`. Transparency of points when `col_layer` is `NULL`. Default is `1` (fully opaque).
`col_layer`	Character or `NULL`. Name of a column in `data` to use for coloring points by a continuous variable. If `NULL` (default), all points are drawn with `pts_col`.
`pal`	A color palette function or character vector of colors used when `col_layer` is provided. Default is `hcl.colors(100, palette = "Viridis")`.
`rev_pal`	Logical. If `TRUE`, reverses the color palette before applying it. Default is `FALSE`.
`pch`	Integer or character. Point symbol. Default is `1`.
`cex`	Numeric. Size scaling for points. Default is `1`.
`bg_sample`	Integer or `NULL`. If provided and `nrow(data)` exceeds this value, a random subsample of this size is drawn before plotting. Useful for large data frames. Default is `NULL` (plot all).
`...`	Additional arguments passed to `points`.

Details

When `col_layer` is provided, points are colored by the values of that column mapped onto the palette. `NA`s in `col_layer` are removed before plotting; zeros are retained as valid values (e.g., truncated suitability predictions outside the ellipsoid boundary).

Value

Called for its side effect of adding points to the current plot. Returns NULL invisibly.

See Also

plot_ellipsoid, add_ellipsoid

Examples

```
data("ref_ellipse", package = "nicheR")
data("back_data", package = "nicheR")

# Open base plot then add centroid as a cross
plot_ellipsoid(ref_ellipse,
               background = back_data,
               col_ell = "#e10000", col_bg = "grey80",
               lwd = 2, pch = 20, cex_bg = 0.4,
               xlab = "Bio1", ylab = "Bio12")

# Add points colored by suitability on top of background
pred_df <- utils::read.csv(system.file("extdata", "predictions_virt.csv", package = "nicheR"))
plot_ellipsoid(ref_ellipse,
               background = back_data,
               col_ell = "#e10000", col_bg = "grey80",
               lwd = 2, pch = 20, cex_bg = 0.4,
               xlab = "Bio1", ylab = "Bio12")
add_data(pred_df,
         x = "bio_1", y = "bio_12",
         col_layer = "suitability",
         pch = 20, cex = 0.5)
```

---

add_data_3d *Add data to an existing 3D E-space plot*

Description

Add data to an existing 3D E-space plot

Usage

```
add_data_3d(data, dim = c(1, 2, 3), col_layer = NULL, alpha = 1, ...)
```

Arguments

data	A data frame or matrix containing the points.
dim	Integer vector of length 3. Indices of dimensions to plot.
col_layer	Character or NULL. Column for coloring.
alpha	Transparency for the points. Default is 1.
...	Additional arguments passed to rgl::points3d.

Value

adds data to an existing 3D e-space plot

Examples

```
# Building an ellipsoid
## Define ranges for three variables
range <- data.frame(bio_1  = c(22, 32),
                    bio_12 = c(800, 4200),
                    bio_15 = c(45, 115))

## Build the ellipsoid
ell5 <- build_ellipsoid(range = range)
ell5$cov_limits

ell5u <- update_ellipsoid_covariance(ell5, c("bio_1-bio_12" = 200,
                                             "bio_1-bio_15" = 0,
                                             "bio_12-bio_15" = -3000))

if(requireNamespace("rgl", quietly = TRUE)){

# Plot the ellipsoid in 3D
plot_ellipsoid_3d(ell5u)

# Add background points
add_data_3d(back_data[, c(3, 7, 10)], col = "#8A8A8A")

 }
```

---

add_ellipsoid *Add an ellipsoid boundary to an existing E-space plot*

Description

Draws the 2D boundary of a `nicheR_ellipsoid` object onto an existing environmental space plot created with `plot_ellipsoid()`. The boundary is computed as a cross-section of the ellipsoid at the chosen pair of dimensions.

Usage

```
add_ellipsoid(object,
                      dim = c(1, 2), lty = 1, lwd = 1,
                      col_ell = "#000000", alpha_ell = 1,
                      cex_ell = 1, ...)
```

Arguments

object	A nicheR_ellipsoid object.
dim	Integer vector of length 2. Indices of the two dimensions to plot. Default is c(1, 2).
lty	Integer. Line type. Default is 1 (solid).
lwd	Numeric. Line width. Default is 1.
col_ell	Character. Color of the ellipsoid boundary line. Default is "#000000" (black).
alpha_ell	Numeric in [0, 1]. Transparency of the ellipsoid boundary line. Default is 1 (fully opaque).
cex_ell	Numeric. Size scaling for the ellipsoid boundary. Default is 1.
...	Additional arguments passed to lines.

Value

Called for its side effect of adding lines to the current plot. Returns NULL invisibly.

See Also

plot_ellipsoid, add_data

Examples

```
data("example_sp_2", package = "nicheR")
data("example_sp_1", package = "nicheR")
data("back_data", package = "nicheR")

# Open a plot, then overlay the ellipsoid prominently
plot_ellipsoid(example_sp_2,
               background = back_data,
               col_ell = "grey70", col_bg = "grey80",
               lwd = 1, pch = 20, cex_bg = 0.3,
               xlab = "Bio1", ylab = "Bio12")

add_ellipsoid(example_sp_1, col_ell = "#e10000", lwd = 2)

# Compare two ellipsoids on the same plot
plot_ellipsoid(example_sp_1,
               background = back_data,
               col_ell = "#e10000", col_bg = "grey80",
               lwd = 2, pch = 20, cex_bg = 0.3,
               xlab = "Bio1", ylab = "Bio12",
               main = "Two ellipsoids")

add_ellipsoid(example_sp_2, col_ell = "#0004d5", lwd = 2)
```

add_ellipsoid_3d *Add an ellipsoid to an existing 3D E-space plot*

Description

Add an ellipsoid to an existing 3D E-space plot

Usage

```
add_ellipsoid_3d(object, dim = c(1, 2, 3), wire = FALSE, col_ell = "#800000",
                 alpha_ell = 1, ...)
```

Arguments

object	A nicheR_ellipsoid object.
dim	Integer vector of length 3.
wire	Logical. If TRUE, plots wireframe, otherwise plots a shaded volume. Default is FALSE.
col_ell	Color of the ellipsoid. Default is "#000000".
alpha_ell	Transparency of the ellipsoid. Default is 1. Not applied if wire = TRUE.
...	Additional arguments passed to rgl::wire3d or rgl::shade3d.

Value

adds a new ellipsoid object to the 3d rendering of the 3D plot

Examples

```
# Building an ellipsoid
## Define ranges for three variables
range <- data.frame(bio_1  = c(22, 32),
                    bio_12 = c(800, 4200),
                    bio_15 = c(45, 115))

## Build the ellipsoid
ell5 <- build_ellipsoid(range = range)
ell5$cov_limits

ell5u <- update_ellipsoid_covariance(ell5, c("bio_1-bio_12" = 200,
                                             "bio_1-bio_15" = 0,
                                             "bio_12-bio_15" = -3000))

if(requireNamespace("rgl", quietly = TRUE)){

# Plot the ellipsoid in 3D
plot_ellipsoid_3d(ell5u)
```

```
# Add the original ellipsoid as a wireframe
add_ellipsoid_3d(ell5, wire = TRUE, col = "#0e008b")

  }
```

`apply_bias` *Apply sampling bias to suitability surfaces*

Description

Applies a prepared composite sampling bias surface to a suitability raster by multiplication. The bias surface is aligned to the suitability grid when needed and the result is cropped and masked to the suitability domain. The output is a product of suitability and bias and is therefore no longer interpretable as a probability.

Usage

```
apply_bias(prepared_bias, prediction, prediction_layer = NULL,
                  effect_direction = "direct", verbose = TRUE)
```

Arguments

`prepared_bias`	A single-layer `SpatRaster` composite bias surface, or the list output from `prepare_bias` containing a `composite_surface` element.
`prediction`	A `SpatRaster` containing one or more suitability layers with values in `[0, 1]`.
`prediction_layer`	Character. Name of the layer to extract from `prediction` when it contains multiple layers. If `NULL` (default) and `prediction` has a single layer, that layer is used.
`effect_direction`	Character. How the bias surface is applied to the suitability layer. `"direct"` (default) multiplies suitability by the bias directly — higher bias increases sampling probability. `"inverse"` multiplies by $1 -$ bias — higher bias decreases sampling probability.
`verbose`	Logical. If `TRUE` (default), prints progress messages.

Details

The function performs the following steps:

1. Extracts the composite bias surface from `prepared_bias`.
2. Verifies both bias and suitability values are within `[0, 1]`.
3. Aligns the bias surface to the suitability grid if geometries differ, using `terra::resample()` with nearest-neighbor interpolation.
4. Multiplies suitability by the (possibly inverted) bias surface.
5. Crops and masks the output to the suitability domain.

Value

A named list of class "nicheR_biased_surface" containing:

- One SpatRaster per input suitability layer, named "<layer>_biased". The raster layer name includes the applied direction (e.g., "suitability_biased_direct").
- combination_formula: a character string describing the operation applied (e.g., "suitability * bias" or "suitability * (1-bias)").

See Also

prepare_bias to build the composite bias surface, sample_biased_data to sample occurrences from the output.

Examples

```
pred_rast <- terra::rast(system.file("extdata/predictions_rast.tif",
                                     package = "nicheR"))

bias_rast <- terra::rast(system.file("extdata/ma_biases.tif",
                                     package = "nicheR"))

# 1. Prepare and standardized bias layers
bias <- prepare_bias(bias_surface = bias_rast[[1]],
                     effect_direction = "direct")

# 2. Apply bias into suitability layer
biased_pred <- apply_bias(prepared_bias = bias,
                          prediction = pred_rast,
                          prediction_layer = "suitability")

terra::plot(biased_pred$suitability_biased)
```

---

back_data *Background environmental data for examples*

Description

A dataset containing geographic coordinates and two bioclimatic variables used as a background point cloud for generating and testing niche ellipsoids.

Usage

```
back_data
```

Format

A data frame with 12,396 rows and 4 variables:

x Longitude in decimal degrees (WGS84).

y Latitude in decimal degrees (WGS84).

bio_1 Annual Mean Temperature (°C).

bio_5 Max Temperature of Warmest Month (°C).

bio_6 Min Temperature of Coldest Month (°C).

bio_7 Temperature Annual Range (bio_5 - bio_6) (°C).

bio_12 Annual Precipitation (mm).

bio_13 Precipitation of Wettest Month (mm).

bio_14 Precipitation of Driest Month (mm).

bio_15 Precipitation Seasonality (Coefficient of Variation).

Details

This dataset represents an irregular point cloud typical of environmental background data used in ecological niche modeling (ENM). It is primarily used in the 'nicheR' package to provide the environmental space for functions like `conserved_ellipses`.

Source

https://www.worldclim.org

Examples

```
data(back_data)
head(back_data)
```

---

build_ellipsoid *Build a probabilistic ellipsoidal niche from ranges*

Description

Builds an ellipsoidal niche in multivariate environmental space using a multivariate normal (MVN) contour defined by a constant Mahalanobis distance. The ellipsoid is parameterized from user-provided variable ranges by deriving a centroid and marginal standard deviations, and assuming a diagonal covariance matrix.

Usage

```
build_ellipsoid(range, cl = 0.99,
                        verbose = TRUE)
```

Arguments

range	A 2-row matrix or data.frame of bounds, with variables as columns. Rows may be ordered as min/max or max/min. Column names are required and used as variable names.
cl	Numeric confidence level in (0, 1). Used to compute the chi-square cutoff defining the ellipsoid contour.
verbose	Logical; if TRUE, prints brief progress messages.

Details

range must be a 2-row matrix or data.frame with variables in columns. Rows represent lower and upper bounds for each variable (row order may be min/max or max/min). The centroid is computed as:

$$\mu_i = (m_i + M_i)/2.$$

Marginal standard deviations are derived assuming bounds represent approximately ± 3 standard deviations:

$$\sigma_i = (M_i - m_i)/6,$$

and a diagonal covariance matrix is assumed:

$$\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2).$$

The ellipsoid contour is defined using a chi-square cutoff $c^2 = \chi^2_n(\mathrm{cl})$, where n is the number of variables.

Value

An object of class "nicheR_ellipsoid" produced by ellipsoid_calculator (via new_nicheR_ellipsoid), containing ellipsoid geometry and associated quantities (e.g., centroid, covariance matrix, chi-square cutoff, semi-axis lengths, axis vertex coordinates, volume, and covariance limits).

Examples

```
# Two-dimensional ellipsoid from environmental ranges
range_df <- data.frame(bio_1  = c(22, 28),
                       bio_12 = c(1000, 3500))
ell2d <- build_ellipsoid(range = range_df)
ell2d

# Three-dimensional ellipsoid
range_3d <- data.frame(bio_1  = c(22, 28),
                       bio_12 = c(1000, 3500),
                       bio_15 = c(50, 70))
ell3d <- build_ellipsoid(range = range_3d)
ell3d
```

`conserved_ellipses` *Generate ellipses via multivariate normal biased sampling*

Description

Creates a set of ellipses with centroids sampled from a background, biased by their proximity to the centroid to a reference niche. Includes an option to thin the background to reduce centroid sampling bias due to point-density.

Usage

```
conserved_ellipses(object, background, n = 10, smallest_proportion = 0.1,
                   largest_proportion = 1.0, thin_background = FALSE,
                   resolution = 100, seed = 1)
```

Arguments

`object`	A nicheR_ellipsoid object used as the reference. This is will be considered the "largest" ellipse to be generated.
`background`	Matrix or Dataframe. The 2D point cloud (coordinates) used to select centroids for the ellipses.
`n`	Integer. Number of ellipses to generate. Default = 10.
`smallest_proportion`	Numeric scalar in `(0, 1)`. The scale of the smallest ellipse relative to the original. Default is `0.1`.
`largest_proportion`	Numeric. Maximum scaling factor for the variance relative to the reference. Default = 1.0. This controls how much larger the new ellipses can be compared to the reference. Values larger than 1 will result in ellipses that exceed the reference size.
`thin_background`	Logical. If TRUE, centroids are sampled more uniformly across the background using a grid-based thinning approach. Default = FALSE.
`resolution`	Integer. Number of cells per side in the grid to deal with point density variation across background. Default = 100.
`seed`	Integer. Random seed for reproducibility. Default = 1. Set to NULL for no seeding.

Details

Ellipses are generated to simulate a community of niches with varying degrees of similarity to the reference. The distribution of the generated ellipses is influenced by the proximity to the reference and the density of the background points.

Value

An object of class `nicheR_community` containing the generated ellipses, the reference object, and generation metadata.

Examples

```
# Loading data
## Reference niche
data("ref_ellipse", package = "nicheR")

## Background data
data("back_data", package = "nicheR")

# Generate conserved ellipses
conserved_comm <- conserved_ellipses(object = ref_ellipse,
                                     background = back_data[, c(3, 7)],
                                     n = 10)
```

`covariance_limits` *Calculate safe covariance ranges for positive definiteness*

Description

Identifies the maximum and minimum covariance values that maintain a positive definite (PD) variance-covariance matrix. For niches with more than two dimensions, it independently shrinks the positive and negative theoretical limits until the "all-maximum" and "all-minimum" covariance scenarios are globally valid.

Usage

```
covariance_limits(varcov_matrix, tol = 1e-6)
```

Arguments

`varcov_matrix`	A square numerical matrix with variances on diagonal.
`tol`	Small value to subtract from limits to ensure strict PD.

Details

The function begins by calculating the deterministic 2D limits for each pair of variables ($|cov_{ij}| < \sqrt{\sigma_i^2 \cdot \sigma_j^2}$).

In higher dimensions (> 2), satisfying all pairwise limits is necessary but not sufficient to ensure the entire matrix is PD. Specifically, contradictory negative correlations can lead to non-transitivity errors. To provide a "safe zone" for users, the function independently shrinks the maximum and minimum bounds by 1

It tests the "all-maximum" and "all-minimum" matrices using Cholesky decomposition. If a test fails, the respective bounds are reduced until a globally valid state is found. This independent

approach ensures that geometric constraints in the negative correlation space do not unnecessarily penalize the limits in the positive correlation space.

Value

A data frame with columns `min` and `max`, where each row represents a pair of dimensions named using the column names of the input matrix.

Examples

```
# Example 1: 2D Matrix (Standard Deterministic Limits)
v_mat_2d <- matrix(c(10, 0, 0, 5), nrow = 2)
colnames(v_mat_2d) <- c("temp", "precip")
covariance_limits(v_mat_2d)

# Example 2: 3D Matrix (Independent Shrinkage Exploration)
v_mat_3d <- matrix(c(10, 0, 0, 0, 5, 0, 0, 0, 7), nrow = 3)
colnames(v_mat_3d) <- c("temp", "precip", "hum")
covariance_limits(v_mat_3d)
```

`ellipsoid_calculator` *Calculate n-dimensional ellipsoid metrics*

Description

Computes geometric and probabilistic metrics for an n-dimensional ellipsoid defined by a centroid and covariance matrix, including semi-axis lengths, axis vertices, and hypervolume for a chi-square confidence contour.

Usage

```
ellipsoid_calculator(cov_matrix, centroid,
                           cl, verbose = TRUE)
```

Arguments

`cov_matrix`	A square, numeric covariance matrix Σ. Must be SPD. Row/column names (if provided) are used as variable names in the output.
`centroid`	Numeric vector giving the centroid μ. Must have length equal to `ncol(cov_matrix)`.
`cl`	Numeric confidence level in (0, 1). Used to compute the chi-square cutoff defining the ellipsoid contour.
`verbose`	Logical; if `TRUE`, prints progress messages.

Details

The ellipsoid boundary is defined by the constant Mahalanobis distance contour:

$$(x-\mu)^\top \Sigma^{-1}(x-\mu) = c^2,$$

where μ is the centroid, Σ is the covariance matrix, and $c^2 = \chi^2_n(\mathrm{cl})$ is the chi-square cutoff with n degrees of freedom.

The covariance matrix must be symmetric positive definite (SPD). The inverse covariance is computed via the Cholesky factorization. Semi-axis lengths are computed from covariance eigenvalues λ_i as:

$$a_i = \sqrt{\lambda_i c^2}.$$

Axis vertices are computed along each eigenvector direction as $\mu \pm a_i v_i$. Hypervolume is computed with `ellipsoid_volume`, and covariance-derived limits with `covariance_limits`.

Value

An object of class `"nicheR_ellipsoid"` created by `new_nicheR_ellipsoid`, containing ellipsoid geometry and associated quantities (e.g., centroid, covariance matrix, chi-square cutoff, semi-axis lengths, axis vertex coordinates, volume, and covariance limits).

Examples

```
cm <- matrix(c(11.11, 0,
               0, 17777.78),
             nrow = 2,
             byrow = TRUE)
colnames(cm) <- rownames(cm) <- c("var1", "var2")
ctr <- c(20, 600)
ell <- ellipsoid_calculator(cov_matrix = cm, centroid = ctr, cl = 0.95, verbose = FALSE)
```

`ellipsoid_volume` *Compute ellipsoid hypervolume*

Description

Computes the geometric volume (area in 2D, volume in 3D, hypervolume in higher dimensions) of a p-dimensional ellipsoid defined by its semi-axis lengths.

For semi-axes $a_1, \ldots, a_p$, the volume is:

$$V_p = \frac{\pi^{p/2}}{\Gamma(p/2+1)} \prod_{i=1}^{p} a_i$$

where $\pi^{p/2}/\Gamma(p/2+1)$ is the volume of the unit p-dimensional ball.

In probabilistic niche models, semi-axis lengths are typically derived from covariance eigenvalues and a chi-square cutoff.

Usage

```
ellipsoid_volume(n_dimensions, semi_axes_lengths)
```

Arguments

n_dimensions	Integer. Number of dimensions p.
semi_axes_lengths	Numeric vector of length n_dimensions containing the ellipsoid semi-axis lengths.

Value

Numeric. Geometric volume (or hypervolume) of the ellipsoid.

See Also

build_ellipsoid, ellipsoid_calculator

Examples

```
range_df <- data.frame(bio_1 = c(22, 28),
                       bio_12 = c(1000, 3500))
ell <- nicheR::build_ellipsoid(range = range_df)

ell$volume

# or recalculate
nicheR::ellipsoid_volume(n_dimensions = ell$dimensions, semi_axes_lengths = ell$semi_axes_lengths)
```

---

example_ellipsoids *Example niche ellipsoid objects for virtual communities*

Description

Pre-calculated nicheR_ellipsoid objects representing hypothetical species niches based on bioclimatic variables. These objects are primarily used in the package vignettes and examples to demonstrate ellipsoid creation, covariance adjustments, visual comparisons, and multidimensional niches.

Usage

```
example_sp_1

example_sp_2

example_sp_3

example_sp_4
```

Format

Objects of class `nicheR_ellipsoid` (which are `lists`) with 13 elements:

dimensions Integer. Number of dimensions (2 or 3).

var_names Character vector. Names of variables (e.g., `"bio_1"`, `"bio_12"`).

centroid Named numeric vector. The center of the niche (μ).

cov_matrix Matrix. The covariance matrix (Σ).

Sigma_inv Matrix. The precision matrix (inverse covariance).

chol_Sigma Matrix. Cholesky decomposition of the covariance.

eigen List. Eigenvectors and eigenvalues of the covariance.

cl Numeric. Confidence level used (e.g., 0.99).

chi2_cutoff Numeric. The chi-square quantile for the given `cl`.

semi_axes_lengths Numeric vector. Radii of the ellipsoid axes.

axes_coordinates List. Vertices (endpoints) for each ellipsoid axis.

volume Numeric. The hyper-volume of the ellipsoid.

cov_limits List. Axis-aligned minimum and maximum covariance limits.

An object of class `nicheR_ellipsoid` of length 13.

An object of class `nicheR_ellipsoid` of length 13.

An object of class `nicheR_ellipsoid` of length 14.

Details

These objects serve as templates for testing community simulation and projection functions. They were generated using `build_ellipsoid` and modified with `update_ellipsoid_covariance` to reflect distinct ecological strategies:

- `example_sp_1`: A 2D niche (`bio_1`, `bio_12`) with a broad, warm-climate preference and wide precipitation tolerance. It includes a positive covariance (750) between temperature and precipitation.
- `example_sp_2`: A 2D niche (`bio_1`, `bio_12`) shifted toward cooler and wetter environments compared to `example_sp_1`, with a positive covariance (500).
- `example_sp_3`: A 2D niche (`bio_1`, `bio_12`) representing a warm-adapted specialist restricted to dry environments. It has a much smaller niche volume and a slight positive covariance (120).
- `example_sp_4`: A 3D niche (`bio_1`, `bio_12`, `bio_15`) adapted to seasonally pulsed precipitation environments. It features a strong negative covariance (-5000) between annual precipitation and precipitation seasonality.

See Also

`build_ellipsoid`, `update_ellipsoid_covariance`

Examples

```
data(example_sp_1)
print(example_sp_1)

# Access the volume of the 2D broad warm-climate niche
example_sp_1$volume

# Access the covariance matrix of the 3D seasonal precipitation niche
data(example_sp_4)
example_sp_4$cov_matrix
```

`ma_bios` *Bioclimatic variables for part of the Americas*

Description

A `SpatRaster` containing 8 bioclimatic variables representing present-day climatic conditions for an area that covers parts of South and North America. Variables were obtained at a 10 arc-minute resolution. Sourced from WorldClim 2.1: https://worldclim.org/data/worldclim21.html

Format

A `SpatRaster` with 8 layers:

bio_1 Annual Mean Temperature

bio_5 Max Temperature of Warmest Month

bio_6 Min Temperature of Coldest Month

bio_7 Temperature Annual Range (bio_5 - bio_6)

bio_12 Annual Precipitation

bio_13 Precipitation of Wettest Month

bio_14 Precipitation of Driest Month

bio_15 Precipitation Seasonality (Coefficient of Variation)

Value

No return value. Used with function `rast` to load the GeoTIFF file from the package's `inst/extdata` folder.

Examples

```
ma_bios <- terra::rast(system.file("extdata", "ma_bios.tif",
                                   package = "nicheR"))

terra::plot(ma_bios[[1]])
```

`nested_ellipses` *Generate nested ellipses based on a reference ellipse*

Description

Creates a sequence of nested ellipses by scaling the covariance matrix of a reference ellipse. The distribution of the nested ellipses can be controlled using a bias exponent to cluster them toward the border or the centroid.

Usage

```
nested_ellipses(object, n = 10, smallest_proportion = 0.1, bias = 1)
```

Arguments

`object`	An object of class "nicheR_ellipsoid" describing an initial ellipse. Must contain `centroid`, `cov_matrix`, and `cl`.
`n`	Integer. Number of nested ellipses to generate. Default is 10.
`smallest_proportion`	Numeric scalar in (0, 1). The scale of the smallest ellipse relative to the original. Default is `0.1`.
`bias`	Numeric. An exponent controlling the spacing of the nested ellipses. • `bias = 1`: Linear spacing (default). • `0 < bias < 1`: Clusters ellipses toward the border (outer original ellipse). • `bias > 1`: Clusters ellipses toward the centroid (inner smallest ellipse).

Details

The largest ellipse corresponds to the original reference, and the smallest is that scaled by `smallest_proportion`.

The function generates a sequence of scale factors k using the formula: $k_i = \text{smallest\_proportion} + (1 - \text{smallest\_proportion}) \times t_i^{\text{bias}}$, where t_i is a linear sequence from 1 down to 0.

Value

An object of class `nicheR_community` containing the generated ellipses, the reference object, and generation metadata.

Examples

```
# Loading data
## Reference niche
data("ref_ellipse", package = "nicheR")

# Generate nested ellipses
nested_comm <- nested_ellipses(object = ref_ellipse, n = 10)
```

plot_community *Plot a nicheR Community of Ellipses*

Description

Visualizes a nicheR_community object by plotting the background environmental space and the community of ellipses.

Usage

```
plot_community(object, background = NULL, dim = c(1, 2), bg_sample = NULL,
               lty = 1, lwd = 1, col_comm = NULL, col_bg = "#8A8A8A",
               pch = 1, alpha_bg = 1, alpha_comm = 1, cex_bg = 1,
               cex_comm = 1, ...)
```

Arguments

object	A nicheR_community object generated by one of the community simulation functions (e.g., random_ellipses, nested_ellipses, conserved_ellipses).
background	Matrix or Dataframe. The environmental background points.
dim	Numeric vector of length 2. The indices of the variables to plot (default is c(1, 2)).
bg_sample	Integer. Number of background points to sample for plotting. The default, NULL, plots all background points.
lty, lwd	Numeric. Line type and width for community ellipses.
col_comm	Character vector. Colors for the community ellipses. Defaults to a rainbow palette.
col_bg	Character. Color for the background points.
pch	Integer or point character for background points.
alpha_bg, alpha_comm	Numeric. Transparency for background and ellipses.
cex_bg, cex_comm	Numeric. Character expansion for points/lines.
...	Additional arguments passed to the base plot function (e.g., xlim, ylim, xlab, ylab).

Value

A plot of the community of ellipses in a 2D plot

Examples

```
# Loading data
## Reference niche
data("ref_ellipse", package = "nicheR")

## Background data
data("back_data", package = "nicheR")

# Generate community of conserved ellipses
conserved_comm <- conserved_ellipses(object = ref_ellipse,
                                     background = back_data[, c(3, 7)],
                                     n = 10)

# Plot the community
plot_community(conserved_comm, background = back_data[, c(3, 7)])
```

`plot_ellipsoid` *Plot a nicheR ellipsoid in environmental space*

Description

Plots the 2D boundary of a `nicheR_ellipsoid` object in environmental space for a chosen pair of dimensions. Optionally overlays background points or a prediction surface colored by a continuous variable (e.g., suitability). Use `add_data` and `add_ellipsoid` to layer additional data onto the plot after calling this function.

Usage

```
plot_ellipsoid(object, background = NULL, prediction = NULL,
               dim = c(1, 2), col_layer = NULL,
               pal = hcl.colors(100, palette = "Viridis"), rev_pal = FALSE,
               bg_sample = NULL,
               lty = 1, lwd = 1, col_ell = "#000000",col_bg = "#8A8A8A",
               pch = 1, alpha_bg = 1, alpha_ell = 1,
               cex_ell = 1, cex_bg = 1,
               fixed_lims = NULL,...)
```

Arguments

`object`	A `nicheR_ellipsoid` object containing at least `centroid`, `cov_matrix`, `chi2_cutoff`, and `var_names`.
`background`	Optional data frame or matrix of background points to plot behind the ellipsoid. Rows are observations, columns are environmental variables. If provided, `prediction` is ignored.
`prediction`	Optional data frame or matrix of prediction values to plot. Used when `background` is `NULL`. Can be colored by a continuous variable using `col_layer`.

dim	Integer vector of length 2. Indices of the two dimensions to plot. Default is c(1, 2).
col_layer	Character or NULL. Name of a column in prediction to use for coloring points by a continuous variable. If NULL (default), all prediction points are drawn with col_bg.
pal	A color palette function or character vector used when col_layer is provided. Default is hcl.colors(100, palette = "Viridis").
rev_pal	Logical. If TRUE, reverses the color palette. Default is FALSE.
bg_sample	Integer or NULL. If provided and the number of background or prediction rows exceeds this value, a random subsample of this size is drawn before plotting. Useful for large data frames. Default is NULL (plot all points).
lty	Integer. Line type for the ellipsoid boundary. Default is 1 (solid).
lwd	Numeric. Line width for the ellipsoid boundary. Default is 1.
col_ell	Character. Color of the ellipsoid boundary line. Default is "#000000" (black).
col_bg	Character. Color of background or prediction points when col_layer is NULL. Default is "#8A8A8A" (grey).
pch	Integer or character. Point symbol for background or prediction points. Default is 1.
alpha_bg	Numeric in [0, 1]. Transparency of background or prediction points. Default is 1 (fully opaque).
alpha_ell	Numeric in [0, 1]. Transparency of the ellipsoid boundary line. Default is 1 (fully opaque).
cex_ell	Numeric. Size scaling for the ellipsoid boundary line. Default is 1.
cex_bg	Numeric. Size scaling for background or prediction points. Default is 1.
fixed_lims	A named list with elements xlim and ylim, each a numeric vector of length 2. When provided, overrides the limits computed by safe_lims(). Intended for use by plot_ellipsoid_pairs to enforce consistent axis scales across panels, but can also be set manually by the user. Default is NULL (limits computed from data).
...	Additional graphical parameters passed to plot.

Details

The function has three display modes depending on what is provided:

1. **Background only** (background is not NULL): plots background points in col_bg with the ellipsoid boundary overlaid.
2. **Prediction surface** (background is NULL, prediction is not NULL): plots prediction points, optionally colored by col_layer using values mapped onto pal. When col_layer is provided, points outside the ellipsoid (zero or NA in col_layer, as produced by truncated prediction types) are drawn in col_bg behind the colored interior points. Axis limits are computed from the full prediction extent so the view is never collapsed to the ellipsoid interior.
3. **Ellipsoid only** (both NULL): plots the ellipsoid boundary alone with no background.

Value

Called for its side effect of creating a plot. Returns `NULL` invisibly.

See Also

`add_data` to overlay occurrence points, `add_ellipsoid` to overlay additional ellipsoid boundaries, `plot_ellipsoid_pairs` for pairwise plots of all dimensions `vignette("plotting_vignette", package = "nicheR")`

Examples

```
data("ref_ellipse", package = "nicheR")
data("back_data", package = "nicheR")

# Mode 1: ellipsoid boundary only
plot_ellipsoid(ref_ellipse,
               col_ell = "#e10000", lwd = 2,
               xlab = "Bio1 (Mean Annual Temperature)",
               ylab = "Bio12 (Annual Precipitation)")

# Mode 2: with background points
plot_ellipsoid(ref_ellipse,
               background = back_data,
               col_ell = "#e10000", col_bg = "grey70",
               lwd = 2, pch = 20, cex_bg = 0.4,
               xlab = "Bio1", ylab = "Bio12")

# Mode 3: prediction colored by suitability
pred_df <- utils::read.csv(system.file("extdata", "predictions_virt.csv", package = "nicheR"))

plot_ellipsoid(ref_ellipse,
               prediction = pred_df,
               col_layer = "suitability",
               bg_sample = 1000,
               col_ell = "#e10000", lwd = 2,
               pch = 20, cex_bg = 0.4,
               xlab = "Bio1", ylab = "Bio12")

# Mode 3b: truncated suitability, outside points shown in grey
plot_ellipsoid(ref_ellipse,
               prediction = pred_df,
               col_layer = "suitability_trunc",
               col_bg  = "#d4d4d4",
               col_ell = "#e10000", lwd = 2, pch = 20, cex_bg = 0.4,
               xlab = "Bio1", ylab = "Bio12")
```

plot_ellipsoid_3d *Plot a nicheR ellipsoid in 3D environmental space*

Description

Creates an interactive 3D plot of a nicheR_ellipsoid object with support for background points or suitability surfaces.

Usage

```
plot_ellipsoid_3d(object, dim = c(1, 2, 3), wire = FALSE, aspect = TRUE,
                  background = NULL, prediction = NULL, col_layer = NULL,
                  pal = hcl.colors(100, palette = "Viridis"),
                  rev_pal = FALSE, bg_sample = NULL, col_ell = "#8b0000",
                  alpha_ell = 1, alpha_bg = 1, col_bg = "#8A8A8A",
                  fixed_lims = NULL, xlab = NULL, ylab = NULL,
                  zlab = NULL, ...)
```

Arguments

object	A nicheR_ellipsoid object constructed with at least 3 dimensions.
dim	Integer vector of length 3. Indices of dimensions to plot.
wire	Logical. If TRUE, plots wireframe. The default, FALSE, plots a shaded volume.
aspect	Logical. If TRUE, maintains aspect ratio (1:1:1).
background	Optional data frame/matrix of background points. This argument has priority over prediction if both are provided.
prediction	Optional data frame/matrix for prediction surfaces.
col_layer	Character or NULL. Column in prediction to use for coloring.
pal	Color palette function or character vector.
rev_pal	Logical. If TRUE, reverses the palette.
bg_sample	Integer or NULL. Subsample size for large data.
col_ell	Color of the ellipsoid. Default is "#000000".
alpha_ell	Transparency of the ellipsoid. Default is 1. Not applied to wireframe mode.
alpha_bg	Transparency of background points. Default is 1. Also applied to prediction points.
col_bg	Color for background points.
fixed_lims	Named list with xlim, ylim, and zlim.
xlab	x-axis label. The default, NULL, uses ellipsoid object variable names, if any found.
ylab	y-axis label. The default, NULL, uses ellipsoid object variable names, if any found.
zlab	z-axis label. The default, NULL, uses ellipsoid object variable names, if any found.
...	Additional graphical parameters.

Value

A 3d plot of the ellipsoid in E-space, shown in viewer.

Examples

```
# Building an ellipsoid
## Define ranges for three variables
range <- data.frame(bio_1  = c(22, 32),
                    bio_12 = c(800, 4200),
                    bio_15 = c(45, 115))

## Build the ellipsoid
ell5 <- build_ellipsoid(range = range)
ell5$cov_limits

ell5 <- update_ellipsoid_covariance(ell5, c("bio_1-bio_12" = 200,
                                            "bio_1-bio_15" = 0,
                                            "bio_12-bio_15" = -3000))

# Plot the ellipsoid in 3D

if(requireNamespace("rgl", quietly = TRUE)){
  plot_ellipsoid_3d(ell5)
 }
```

---

plot_ellipsoid_pairs *Plot all pairwise 2D ellipsoid projections*

Description

Plots all pairwise two-dimensional slices of a `nicheR_ellipsoid` in a multi-panel layout using `plot_ellipsoid()`. When `background` or `prediction` is supplied, axis limits are computed once from the global range of all variables and shared across every panel, so projections are directly comparable without distortion from per-panel rescaling.

Usage

```
plot_ellipsoid_pairs(object, background = NULL, prediction = NULL, ...)
```

Arguments

`object`	A `nicheR_ellipsoid` object.
`background`	Optional data frame or matrix of background points passed to each `plot_ellipsoid()` call. When provided, global axis limits are computed from the range of all variables in `background` combined with all pairwise ellipsoid boundaries.
`prediction`	Optional data frame or matrix of prediction values passed to each `plot_ellipsoid()` call. Used when `background` is `NULL`. Global limits are computed from the range of all variables in `prediction`.

... Additional graphical arguments passed to `plot_ellipsoid()`.

Details

Global limits are computed per variable across the full data and all ellipsoid boundary projections, then passed to each panel via the `fixed_lims` argument of `plot_ellipsoid()`. This prevents individual panels from rescaling to their own data extent, which would make niche widths appear identical across dimensions even when they differ. If neither `background` nor `prediction` is provided, each panel shows only the ellipsoid boundary and limits come from that boundary alone, which is the intended behavior for a boundary-only view.

Value

Invisibly returns `NULL`.

Examples

```
data("example_sp_4", package = "nicheR")
data("back_data", package = "nicheR")
ell3d <- example_sp_4

# Boundary only
nicheR::plot_ellipsoid_pairs(ell3d, col_ell = "#e10000", lwd = 2)

# With background: global limits shared across all panels
ma_bios <- terra::rast(system.file("extdata/ma_bios.tif", package = "nicheR"))
back_df <- as.data.frame(ma_bios, xy = TRUE)

plot_ellipsoid_pairs(ell3d,
                     background = back_df,
                     col_ell = "#e10000", col_bg = "grey70",
                     lwd = 2, pch = 20, cex_bg = 0.3)

# With truncated suitability predictions
pred_trunc <- predict(ell3d,
                      newdata = back_df[, ell3d$var_names],
                      include_suitability = FALSE,
                      include_mahalanobis = FALSE,
                      suitability_truncated = TRUE)

plot_ellipsoid_pairs(ell3d,
                     prediction = pred_trunc,
                     col_layer  = "suitability_trunc",
                     col_bg  = "#d4d4d4",
                     col_ell = "#e10000", lwd = 2, pch = 20, cex_bg = 0.3)

#'
```

predict *Predict suitability and Mahalanobis distance from a nicheR ellipsoid*

Description

Computes Mahalanobis distance and suitability values deriving from a `nicheR_ellipsoid` or `nicheR_community` object, for `newdata` provided as a `data.frame`, `matrix`, or `SpatRaster`.

Usage

```
## S3 method for class 'nicheR_ellipsoid'
predict(
  object,
  newdata,
  adjust_truncation_level = NULL,
  include_suitability = TRUE,
  suitability_truncated = FALSE,
  include_mahalanobis = TRUE,
  mahalanobis_truncated = FALSE,
  keep_data = NULL,
  verbose = TRUE,
  ...
)

## S3 method for class 'nicheR_community'
predict(object, newdata, prediction = "Mahalanobis", verbose = TRUE, ...)
```

Arguments

`object` An object of the classes `"nicheR_ellipsoid"` or `"nicheR_community"`.

`newdata` Environmental predictors. One of:

- A `SpatRaster` (or legacy `raster` classes, coerced automatically).
- A `data.frame` or `matrix` with columns named to match `object$var_names`.

`adjust_truncation_level` Optional numeric confidence level in `(0, 1)` to override `object$cl` when computing truncated outputs. Default is `NULL` (uses the level stored in `object`).

`include_suitability` Logical. If `TRUE` (default), returns suitability values ($\exp(-0.5D^2)$).

`suitability_truncated` Logical. If `TRUE`, returns a truncated suitability layer where values outside the chi-square contour are set to `0`. Default is `FALSE`.

`include_mahalanobis` Logical. If `TRUE` (default), returns Mahalanobis distance (D^2).

`mahalanobis_truncated` Logical. If `TRUE`, returns a truncated Mahalanobis layer where values outside the chi-square contour are set to `NA`. Default is `FALSE`.

keep_data	Logical or NULL. If TRUE, includes the original predictors in the output. Default is NULL: FALSE for SpatRaster input, TRUE for tabular input.
verbose	Logical. If TRUE (default), prints progress messages.
...	Additional arguments, not currently used.
prediction	Character. The type of prediction to return. One of: "Mahalanobis" (default), "suitability", "Mahalanobis_trunc", or "suitability_trunc".

Details

Suitability is computed as $\exp(-0.5D^2)$, where D^2 is the squared Mahalanobis distance from the niche centroid. Truncated outputs use a chi-square cutoff based on the ellipsoid confidence level (cl).

For tabular inputs, coordinate columns (e.g., x, y, lon, lat) are detected and retained when keep_data = TRUE. Extra non-predictor columns are ignored.

Value

For nicheR_ellipsoid objects, if newdata is a SpatRaster, returns a SpatRaster with the requested prediction as layers (and optionally the original predictor layers if keep_data = TRUE). If newdata is tabular, returns a data.frame with the requested predictions as columns (by default returns the original predictors as columns).

For nicheR_community objects, if newdata is a SpatRaster, returns a SpatRaster where each layer represents predictions for each ellipse. If newdata is a data.frame, returns a data.frame with the original data plus one prediction column per ellipse.

Examples

```
range_df <- data.frame(bio_1 = c(22, 28),
                       bio_12 = c(1000, 3500))
ell <- build_ellipsoid(range = range_df)


ma_bios <- terra::rast(
  system.file("extdata/ma_bios.tif", package = "nicheR"))
back_df <- as.data.frame(ma_bios, xy = TRUE)

# Default: Mahalanobis distance and suitability, data frame input
pred_df <- predict(ell,
                   newdata = back_df)
head(pred_df)

# All four outputs at once
pred_all <- predict(ell,
                    newdata = back_df,
                    include_mahalanobis = TRUE,
                    include_suitability = TRUE,
                    mahalanobis_truncated = TRUE,
                    suitability_truncated = TRUE)
colnames(pred_all)
```

```
nicheR::plot_ellipsoid(object = ell, prediction = pred_all)
#' nicheR::plot_ellipsoid(object = ell, prediction = pred_all, col_layer = "suitability")

# Raster input: returns a SpatRaster
pred_rast <- predict(ell,
                     newdata = ma_bios[[ell$var_names]],
                     include_suitability = TRUE,
                     suitability_truncated = TRUE)
pred_rast

terra::plot(pred_rast)

# Adjust truncation level without refitting
pred_80 <- predict(ell,
                   newdata = back_df,
                   suitability_truncated   = TRUE,
                   adjust_truncation_level = 0.80)
nicheR::plot_ellipsoid(object = ell, prediction = pred_80, col_layer = "suitability_trunc")
```

`prepare_bias` *Prepare sampling bias surfaces*

Description

Standardizes and combines one or more bias layers into a composite bias surface for use in biased occurrence sampling. Each layer is min-max normalized to `[0, 1]` and assigned a directional effect (`"direct"` or `"inverse"`) before being multiplied together into a single composite surface.

Usage

```
prepare_bias(bias_surface, effect_direction = c("direct", "inverse"),
                     template_layer = NULL, include_composite = TRUE,
                     include_processed_layers = FALSE, mask_na = FALSE,
                     verbose = TRUE)
```

Arguments

`bias_surface`	A `SpatRaster` (single or multi-layer) or a list of `SpatRaster` objects representing the raw bias layers.
`effect_direction`	Character vector. Direction of effect for each bias layer. Each element must be `"direct"` (higher values increase sampling probability) or `"inverse"` (higher values decrease sampling probability). Length 1 recycles to all layers. Must otherwise match the number of bias layers.
`template_layer`	Optional `SpatRaster`. Reference layer used to align all bias layers (resolution, extent, CRS). If `NULL` (default), the finest-resolution bias layer is used as the template.

include_composite
: Logical. If TRUE (default), includes the composite bias surface in the output.

include_processed_layers
: Logical. If TRUE, includes the standardized individual layers (after directional transformation) in the output. Default is FALSE.

mask_na
: Logical. Controls how NA values are handled when combining layers. If TRUE, uses the intersection of layer extents — any pixel with an NA in any layer becomes NA in the composite. If FALSE (default), uses the union of extents and ignores NAs where other layers have valid values.

verbose
: Logical. If TRUE (default), prints progress messages.

Details

Each bias layer is processed as follows:

1. Resampled and cropped to the template grid if needed.
2. Min-max standardized to [0, 1].
3. Transformed by direction: "direct" layers are kept as-is; "inverse" layers are replaced by $1 - x$.

The composite surface is the product of all transformed layers. If only one layer is provided, it is returned as-is.

If both include_composite and include_processed_layers are FALSE, the function defaults to include_composite = TRUE with a warning.

Value

A named list with some or all of the following elements depending on the include_composite and include_processed_layers arguments:

- composite_surface: A SpatRaster with the combined bias surface, named "standarized_composite_bias_surfac
- processed_layers: A multi-layer SpatRaster with the standardized and direction-transformed individual layers.
- combination_formula: A character string showing the formula used to combine layers (e.g., "sp_richness * (1-nighttime)").

See Also

apply_bias to apply the prepared bias surface to a suitability prediction, sample_biased_data to sample occurrences from the resulting bias-weighted surface.

Examples

```
pred_rast <- terra::rast(system.file("extdata/predictions_rast.tif",
                                     package = "nicheR"))

bias_rast <- terra::rast(system.file("extdata/ma_biases.tif",
                                     package = "nicheR"))
```

```
# Single layer, direct effect (higher values increase sampling probability)
bias_single <- prepare_bias(bias_surface = bias_rast[[1]],
                            effect_direction = "direct")
bias_single$composite_surface

# Two layers: first direct, second inverse
bias_two <- prepare_bias( bias_surface = bias_rast[[c(1, 2)]],
                          effect_direction = c("direct", "inverse"),
                          include_processed_layers = TRUE)

bias_two$combination_formula
terra::plot(bias_two$processed_layers)
terra::plot(bias_two$composite_surface)
```

`print` *Print method for nicheR objects*

Description

Provides a concise summary of `nicheR` objects.

Usage

```
## S3 method for class 'nicheR_ellipsoid'
print(x, digits = 3, ...)

## S3 method for class 'nicheR_community'
print(x, digits = 3, ...)
```

Arguments

`x`	An object of the classes `"nicheR_ellipsoid"` or `"nicheR_community"`.
`digits`	Integer. Number of decimal places used when printing numeric values. Default is 3.
`...`	Additional arguments.

Details

The function formats and rounds key quantities for readability but does not modify the underlying object.

Value

The input object `x`, returned invisibly.

See Also

`build_ellipsoid`

Examples

```
range_df <- data.frame(bio_1  = c(22, 28),
                       bio_12 = c(1000, 3500))
ell <- build_ellipsoid(range = range_df)
print(ell)
```

`random_ellipses` *Generate random ellipses constrained by a point cloud and a reference ellipse*

Description

Creates n random ellipses with centroids sampled from an irregular point cloud. Covariance matrices are built using random rotations and scaled eigenvalues restricted by user-defined limits.

Usage

```
random_ellipses(object, background, n = 10, smallest_proportion = 0.1,
                largest_proportion = 1.0, thin_background = FALSE,
                resolution = 50, seed = 1)
```

Arguments

`object`	A nicheR_ellipsoid object used as a reference ellipse (the biggest to be generated), and containing at least `covariance_matrix` and `cl`.
`background`	Matrix or Dataframe. The 2D point cloud (coordinates) used to select random centroids.
`n`	Integer. Number of ellipses to generate.
`smallest_proportion`	Numeric. Minimum scaling factor for the variance. Must be between 0 and 1. Default = 0.1. This controls how much smaller the new ellipses can be compared to the reference.
`largest_proportion`	Numeric. Maximum scaling factor for the variance relative to the reference. Default = 1.0. This controls how much larger the new ellipses can be compared to the reference. Values larger than 1 will result in ellipses that exceed the reference size.
`thin_background`	Logical. If TRUE, centroids are sampled more uniformly across the background using a grid-based thinning approach. Default = FALSE.
`resolution`	Integer. Number of cells per side in the grid to deal with point density variation across background. Default = 50.
`seed`	Integer. Random seed for reproducibility. Default = 1. Set to NULL for no seeding.

Value

An object of class `nicheR_community` containing the generated ellipses, the reference object, and generation metadata.

Examples

```
# Loading data
## Reference niche
data("ref_ellipse", package = "nicheR")

## Background data
data("back_data", package = "nicheR")

# Generate random ellipses
rand_comm <- random_ellipses(object = ref_ellipse,
                             background = back_data[, c(3, 7)],
                             n = 10)
```

`range_utilities` *Compute variable ranges from data or statistics with optional expansion*

Description

These functions compute the minimum and maximum values for variables, either directly from a dataset or based on normal distribution parameters, with the ability to expand the resulting ranges by a percentage.

Usage

```
ranges_from_data(data, expand_min = NULL, expand_max = NULL)
ranges_from_stats(mean, sd, cl = 0.95, expand_min = NULL, expand_max = NULL)

ranges_from_data(data, expand_min = NULL, expand_max = NULL)

ranges_from_stats(mean, sd, cl = 0.95, expand_min = NULL, expand_max = NULL)
```

Arguments

`data`	A data.frame of at least two columns. Each column should contain numeric values.
`expand_min`	A named vector or list of percentages (e.g., 10 for 10 defining how much to expand the minimum value of specific variables.
`expand_max`	A named vector or list of percentages defining how much to expand the maximum value of specific variables.
`mean`	A named numeric vector of mean values for each variable.

sd	A named numeric vector of standard deviation values for each variable. Names must match those in 'mean'.
cl	A numeric value indicating the confidence level (default 0.95).

Value

A data.frame with the (potentially expanded) minimum and maximum values of each variable.

Examples

```
# From data
df <- data.frame(var1 = c(0, 10), var2 = c(50, 100))
ranges_from_data(df, expand_min = list(var1 = 10),
                 expand_max = list(var2 = 20))

# From statistics
m <- c(var1 = 10, var2 = 100)
s <- c(var1 = 2, var2 = 15)
ranges_from_stats(mean = m, sd = s, cl = 0.95,
                  expand_min = list(var1 = 10))
```

---

read_nicheR *Read a nicheR object from disk*

Description

A wrapper around readRDS to load saved nicheR objects back into the R environment.

Usage

```
read_nicheR(file)
```

Arguments

file	Character. The path to the file to be read.

Value

The saved nicheR_ellipsoid or nicheR_community object.

See Also

save_nicheR

Examples

```
# Build and save an ellipsoid first
range_df <- data.frame(bio_1  = c(15, 25),
                       bio_12 = c(500, 1500))
ell <- build_ellipsoid(range = range_df)

tmp <- tempfile(fileext = ".rds")
save_nicheR(ell, file = tmp)

# Read it back
ell_loaded <- read_nicheR(tmp)
ell_loaded
```

`ref_ellipse` *Reference ellipse for virtual community examples*

Description

A pre-calculated `nicheR_ellipsoid` object representing a hypothetical species niche based on Annual Mean Temperature (bio_1) and Annual Precipitation (bio_12).

Usage

```
ref_ellipse
```

Format

An object of class `nicheR_ellipsoid` (which is a `list`) with 13 elements:

dimensions Integer. Number of dimensions (2).

var_names Character vector. Names of variables (`"bio_1"`, `"bio_12"`).

centroid Named numeric vector. The center of the niche (μ).

cov_matrix Matrix. The 2×2 covariance matrix (Σ).

Sigma_inv Matrix. The precision matrix (inverse covariance).

chol_Sigma Matrix. Cholesky decomposition of the covariance.

eigen List. Eigenvectors and eigenvalues of the covariance.

cl Numeric. Confidence level used (e.g., 0.99).

chi2_cutoff Numeric. The chi-square quantile for the given `cl`.

semi_axes_lengths Numeric vector. Radii of the ellipsoid axes.

axes_coordinates List. Vertices (endpoints) for each ellipsoid axis.

volume Numeric. The hyper-volume of the ellipsoid.

cov_limits List. Axis-aligned minimum and maximum limits.

Details

This object serves as a template for testing community simulation functions like `conserved_ellipses`. It was generated using `build_ellipsoid` with a centroid at (23.75, 1750) and specific covariance structures to reflect a typical temperature-precipitation relationship.

Examples

```
data(ref_ellipse)
print(ref_ellipse)

# Access the volume
ref_ellipse$volume
```

---

sample_biased_data *Sample occurrence data from a bias-weighted prediction surface*

Description

Samples `n_occ` virtual occurrence points using the bias-weighted prediction values directly as sampling probabilities. Unlike `sample_data()`, there is no sampling strategy argument — the prediction layer values themselves define where points are drawn from, making this function suited for simulating realistically biased occurrence records.

Usage

```
sample_biased_data(n_occ, prediction, prediction_layer = NULL,
                           sampling_mask = NULL, seed = 1, verbose = TRUE,
                           strict = NULL)
```

Arguments

`n_occ`	Integer. Number of occurrence points to sample.
`prediction`	A `SpatRaster` or data frame containing the bias-weighted prediction surface to sample from.
`prediction_layer`	Character. Name of the layer or column to use as sampling weights. Required when `prediction` contains multiple layers or columns.
`sampling_mask`	A `SpatRaster` or `SpatVector` used to restrict sampling to a geographic area. Only supported when `prediction` is a `SpatRaster`.
`seed`	Integer. Random seed for reproducibility. Default is `1`.
`verbose`	Logical. If `TRUE` (default), prints progress messages.
`strict`	Logical or `NULL`. If `TRUE`, removes `NA` and zero-valued cells before sampling. If `NULL` (default), auto-detected from the layer name and the proportion of zeros and `NA`s in the prediction values.

Details

Prediction values are used directly as sampling weights, so they must be non-negative. Higher values correspond to higher sampling probability, reflecting areas of greater bias (e.g., higher detectability or observer effort). This is in contrast to `sample_data()`, which transforms prediction values according to a `sampling` and `method` argument.

Auto-detection of `strict` follows the same logic as `sample_data()`: it is set to `TRUE` if the layer name contains `"trunc"` or if the proportion of zeros or `NA`s exceeds 25%.

Value

A data frame of sampled occurrence points with the same columns as the input `prediction` (minus the internal `pred` column). If `prediction` is a `SpatRaster`, the output includes `x` and `y` coordinate columns.

See Also

`sample_data` for unbiased sampling with explicit strategy and method control, `apply_bias` for generating the bias-weighted prediction surface used as input here.

Examples

```
biased_pred <- terra::rast(system.file("extdata/applied_bias_rast.tif",
                                       package = "nicheR"))

# Sample points form bias surface (not probability surface)
occ_biased <- sample_biased_data(n_occ = 100,
                                 prediction = biased_pred,
                                 prediction_layer = "suitability_biased_direct")

head(occ_biased)
```

---

sample_data *Sample occurrence data from a prediction surface*

Description

Samples `n_occ` virtual occurrence points from a suitability or Mahalanobis distance prediction surface. Supports centroid, edge, and random sampling strategies, and accepts both raster (`SpatRaster`) and data frame inputs.

Usage

```
sample_data(n_occ, prediction, prediction_layer = NULL,
                   sampling = "centroid", method = "suitability",
                   sampling_mask = NULL, seed = 1, strict = NULL,
                   verbose = TRUE)
```

Arguments

n_occ	Integer. Number of occurrence points to sample.
prediction	A SpatRaster or data frame containing the prediction surface to sample from.
prediction_layer	Character. Name of the layer or column to use as the prediction values. Required when prediction contains multiple layers or columns.
sampling	Character. Sampling strategy. One of "centroid" (default), "edge", or "random". Controls where within the niche points are preferentially drawn from.
method	Character. Weighting method. One of "suitability" (default) or "mahalanobis". Must match the type of values in prediction_layer: suitability values must be in [0, 1], Mahalanobis values must be non-negative.
sampling_mask	A SpatRaster or SpatVector used to restrict sampling to a geographic area. Only supported when prediction is a SpatRaster.
seed	Integer. Random seed for reproducibility. Default is 1.
strict	Logical or NULL. If TRUE, removes NA and zero-valued cells before sampling (recommended with truncated prediction layers). If NULL (default), auto-detected from the layer name and the proportion of zeros and NAs in the prediction values.
verbose	Logical. If TRUE (default), prints progress messages.

Details

The sampling and method arguments interact to define the probability weights used when drawing points:

- sampling = "centroid", method = "suitability": weights proportional to suitability — higher near the niche center.
- sampling = "edge", method = "suitability": weights proportional to $1 -$ suitability — higher near the niche boundary.
- sampling = "centroid", method = "mahalanobis": weights inversely proportional to Mahalanobis distance — higher near the centroid.
- sampling = "edge", method = "mahalanobis": weights proportional to Mahalanobis distance — higher near the boundary.
- sampling = "random": equal weights regardless of method.

When strict = NULL, the function auto-detects truncation by checking whether the layer name contains "trunc" or whether the proportion of zeros or NAs exceeds 25%.

Value

A data frame of sampled occurrence points with the same columns as the input prediction (minus the internal pred column). If prediction is a SpatRaster, the output includes x and y coordinate columns.

Examples

```
pred_df <- utils::read.csv(system.file("extdata/predictions_virt.csv",
                                       package = "nicheR"))

# Centroid strategy: samples cluster near the niche center
occ_centroid <- sample_data(n_occ = 100,
                            prediction = pred_df,
                            prediction_layer = "suitability_trunc",
                            sampling = "centroid",
                            method = "suitability",
                            strict = TRUE)
head(occ_centroid)

# Edge strategy: samples spread toward the niche boundary
occ_edge <- sample_data(n_occ = 100,
                        prediction = pred_df,
                        prediction_layer = "suitability_trunc",
                        sampling = "edge",
                        method = "mahalanobis",
                        strict = TRUE)

# Random strategy: samples distributed uniformly across suitable area
occ_random <- sample_data(n_occ = 100,
                          prediction = pred_df,
                          prediction_layer = "suitability_trunc",
                          sampling = "random")
```

`save_nicheR` *Save a nicheR object to disk*

Description

A wrapper around saveRDS to save `nicheR_ellipsoid` or `nicheR_community` objects to a file. Includes a safety check for overwriting existing files and ensures the file extension is .rds.

Usage

```
save_nicheR(object, file, overwrite = FALSE, ...)
```

Arguments

`object`	A `nicheR_ellipsoid` or `nicheR_community` object.
`file`	Character. The connection or name of the file where the object will be saved (usually ending in ".rds").
`overwrite`	Logical. If `TRUE`, an existing file at the specified path will be replaced. Default is `FALSE`.
`...`	Additional arguments passed to saveRDS.

Value

No return value. Saves the object to the specified path.

Examples

```
# Build a simple ellipsoid to save
range_df <- data.frame(bio_1  = c(15, 25),
                       bio_12 = c(500, 1500))
ell <- build_ellipsoid(range = range_df)

# Save to a temporary file
tmp <- tempfile(fileext = ".rds")
save_nicheR(ell, file = tmp)

# Overwrite the same file
save_nicheR(ell, file = tmp, overwrite = TRUE)
```

`update_covariance` *Update covariance values and calculate remaining safe limits*

Description

Updates a variance-covariance matrix with specific values and identifies the safe limits for all remaining zero-covariance combinations.

Usage

```
update_covariance(varcov_matrix, covariance = 0, tol = 1e-6)
```

Arguments

`varcov_matrix`	A square numerical matrix with variances on diagonal.
`covariance`	A single value to apply to all off-diagonals, or a named vector where names follow the "dim1-dim2" format.
`tol`	Small value to subtract from limits to ensure strict PD.

Value

A list containing `up_mat` (updated matrix) and `limits` (data frame of safe limits for remaining zero-cov pairs).

Examples

```
# Setup
v_mat_3d <- matrix(c(10, 0, 0, 0, 5, 0, 0, 0, 7), 3, 3)
colnames(v_mat_3d) <- c("temp", "precip", "hum")

# 1. 2D Update: Change only the single available covariance
update_covariance(v_mat_3d[1:2, 1:2], covariance = c("temp-precip" = 3))

# 2. 3D Full Update: Set all combinations to 2.0
update_covariance(v_mat_3d, covariance = 2.0)

# 3. 3D Partial: Define two, find limits for the remaining precip-hum
update_covariance(v_mat_3d, covariance = c("temp-precip" = -4, "temp-hum" = 5))
```

`update_ellipsoid_covariance` *Update covariances in a nicheR ellipsoid and recompute metrics*

Description

Updates one or more off-diagonal covariance values in a `nicheR_ellipsoid` object and recomputes all ellipsoid metrics (centroid, semi-axes, volume, etc.) from the new covariance matrix. This allows iterative niche shaping by adjusting the rotation and correlation structure of the ellipsoid without rebuilding it from scratch.

Usage

```
update_ellipsoid_covariance(object, covariance,
                                    tol = 1e-6,
                                    verbose = TRUE)
```

Arguments

`object`	A `nicheR_ellipsoid` object, typically created with `build_ellipsoid`.
`covariance`	Either a single numeric value applied to all off-diagonal elements, or a named numeric vector where names identify the variable pair in the format `"var1-var2"` (e.g., `c("bio_1-bio_12" = 0.3)`).
`tol`	Small positive number used as tolerance when computing safe covariance limits for positive definiteness. Default is `1e-6`.
`verbose`	Logical. If `TRUE` (default), prints progress messages.

Details

Covariance values control the orientation and correlation structure of the ellipsoid in environmental space. Setting a positive covariance between two variables tilts the ellipsoid so that high values of one variable tend to co-occur with high values of the other. Negative covariance tilts it in the opposite direction.

The updated covariance matrix must remain positive definite — if the requested value would violate this, use covariance_limits to find the safe range before calling this function.

Value

A nicheR_ellipsoid object with the updated covariance matrix and recomputed ellipsoid metrics. An additional element cov_limits_remaining is attached, giving the remaining safe covariance limits for any variable pairs that still have zero covariance.

See Also

build_ellipsoid to create the initial ellipsoid, covariance_limits to find valid covariance ranges before updating.

Examples

```
range_df <- data.frame(bio_1  = c(22, 28),
                       bio_12 = c(1000, 3500))
ell <- build_ellipsoid(range = range_df)

# Check covariance allowed covariance range
ell$cov_limits

# Introduce a negative correlation between bio_1 and bio_12
cov_limits <- c("bio_1-bio_12" = -100)
ell_corr <- update_ellipsoid_covariance(object = ell,
                                        covariance =  cov_limits)
ell_corr
```

---

virtual_data *Generate data based on a ellipsoidal niche*

Description

Simulates n random points from a multivariate normal distribution defined by the centroid and covariance matrix of a nicheR_ellipsoid object.

Usage

```
virtual_data(object, n = 100, truncate = FALSE, effect = "direct", seed = 1)
```

Arguments

object	A nicheR_ellipsoid object containing at least centroid and cov_matrix.
n	Integer. The number of virtual points to generate. Default = 100.
truncate	Logical. If TRUE (default), points are constrained within the confidence limit (cl) defined in the object.

effect	Character. The distribution pattern of points. "direct" (default) creates a concentration near the centroid. "inverse" creates higher density towards the edges. "uniform" distributes points evenly throughout the ellipsoid volume. Note: "inverse" and "uniform" require truncate = TRUE.
seed	Integer. Random seed for reproducibility. Default = 1. Set to NULL for no seeding.

Details

When truncate = FALSE, the function generates points from a standard multivariate normal distribution defined by the ellipsoid's centroid and covariance matrix, without any constraints on their location. The function uses eigen-decomposition to transform standard normal variables into the coordinate system defined by the ellipsoid's covariance structure.

When truncate = TRUE, the function generates candidate points uniformly distributed within a bounding box (hyper-cube) defined by the ellipsoid's axes_coordinates. Points falling outside the ellipsoid (where Mahalanobis distance $Md >$ chi2_cutoff) are removed.

From this filtered pool, n points are selected using weighted random sampling without replacement. The weights are determined by the effect argument:

- "direct": Weights are proportional to the multivariate normal density ($\exp(-0.5 \times Md)$), clustering points near the centroid.
- "inverse": Weights are proportional to the complement of the normal density ($1-\exp(-0.5\times Md)$), pushing points toward the edges.
- "uniform": All points within the ellipsoid have equal weight, resulting in a uniform spatial distribution.

Value

A matrix with n rows and columns corresponding to the environmental variables (dimensions) of the input object.

Examples

```
# Loading data
## Reference niche
data("ref_ellipse", package = "nicheR")

# Generate virtual data from the reference niche
vdata_direct <- virtual_data(ref_ellipse, n = 100, effect = "direct")
vdata_inverse <- virtual_data(ref_ellipse, n = 100,
                              effect = "inverse", truncate = TRUE)

# Check a sample of the generated data
head(vdata_direct)
head(vdata_inverse)
```

Index