# Package 'xnicher'

September 15, 2026

**Type** Package

**Title** Estimates Ecological Niche Models Using Ellipses

**Version** 1.0.0

**Description** Ecological niche model estimation using ellipsoidal geometry under an M hypothesis. Fits nine likelihood families (presence-only, weighted, inverse-probability-weighted, skew-normal, skew-normal-weighted, skew-t, skew-t-weighted, ncst, and ncst-weighted) via multi-start optimisation with Sobol sequences. Methods for the optimisation of ellipses parameters are as described in Jimenez et al. (2022) <doi:10.1016/j.ecolmodel.2021.109823>.

**URL** https://github.com/alrobles/xnicher

**BugReports** https://github.com/alrobles/xnicher/issues

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Depends** R (>= 3.5)

**Imports** checkmate, Rcpp (>= 1.1.0), RcppParallel, stats, terra, ucminfcpp, utils

**Suggests** ggplot2 (>= 3.0.0), knitr, pomp, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppEigen, RcppParallel

**SystemRequirements** GNU make

**Config/roxygen2/version** 8.0.0

**RoxygenNote** 7.1.2

**NeedsCompilation** yes

**Author** Angel Robles [aut, cre], Laura Jimenez [aut, ctb] (ORCID: <https://orcid.org/0000-0002-4674-4270>)

**Maintainer** Angel Robles <a.l.robles.fernandez@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-09-15 11:20:02 UTC

# Contents

## AIC.xnicher — *Akaike information criterion for a xnicher fit*

### Description

Computes $-2 \log L + 2k$, where $\log L$ is the un-penalised log-likelihood (see `logLik.xnicher`) and $k$ is the number of free parameters.

### Usage

```
## S3 method for class 'xnicher'
AIC(object, ..., k = 2)
```

### Arguments

| | |
|---|---|
| `object` | a fitted model object for which there exists a `logLik` method to extract the corresponding log-likelihood, or an object inheriting from class `logLik`. |
| `...` | Additional fitted model objects. |
| `k` | Numeric, the penalty per parameter; default `2` (AIC). Pass `k = log(nobs(fit))` to recover BIC. |

### Value

Numeric scalar, or a `data.frame` when multiple objects are passed.

### Examples

```
fit <- optimize_niche(env_occ = example_env_occ_2d,
                      env_m   = example_env_m_2d,
                      num_starts = 5L, likelihood = "weighted")
AIC(fit)
```

## assess — *Assess acceptance criteria for an optimization result*

### Description

Generic function for evaluating whether an optimization result meets acceptance criteria for niche model quality.

### Usage

```
assess(x, ...)
```

### Arguments

| | |
|---|---|
| x | An object for which an assess method is defined. |
| ... | Additional arguments passed to methods. |

### Value

A list of diagnostics and acceptance flags. See assess.xnicher for details.

---

## assess.xnicher *Assess acceptance criteria for a xnicher result*

---

### Description

Evaluates the quality of a multi-start optimization by comparing converged solutions. Returns one of four diagnostic flags:

"accepted_global" Multiple starts agree closely on the same high log-likelihood value — the global optimum is likely found.

"accepted_noise" Converged solutions are close but show minor numerical noise — likely a good solution.

"suggest_average" Converged solutions spread across distinct values — consider averaging theta or increasing num_starts.

"needs_more_starts" Too few converged solutions to draw conclusions — increase num_starts.

### Usage

```
## S3 method for class 'xnicher'
assess(x, tol_gap = 0.01, tol_dist = 0.05, min_converged = 2L, ...)
```

### Arguments

| | |
|---|---|
| x | A "xnicher" object returned by optimize_niche. |
| tol_gap | Numeric. Relative tolerance used to decide between "accepted_global" and "accepted_noise". Default 0.01. |
| tol_dist | Numeric. Relative tolerance used to decide between "accepted_noise" and "suggest_average". Default 0.05. |
| min_converged | Integer. Minimum number of converged solutions required to avoid "needs_more_starts". Default 2. |
| ... | Ignored. |

## Value

A named list with:

`flag` Character scalar, one of the four flags above.

`recommendation` Human-readable action recommendation.

`gap` Absolute log-likelihood gap between the best and second-best converged solutions.

`rel_gap` Relative gap (`gap / |best_loglik|`).

`n_converged` Number of converged solutions.

`best_loglik` Best log-likelihood value.

## Examples

```
result <- optimize_niche(
  env_occ    = example_env_occ_2d,
  env_m      = example_env_m_2d,
  num_starts = 5L,
  likelihood = "ip_weighted"
)
diag <- assess(result)
cat("Flag          :", diag$flag, "\n")
cat("Recommendation:", diag$recommendation, "\n")
cat("Best log-lik  :", round(diag$best_loglik, 4L), "\n")
```

---

# BIC.xnicher *Bayesian information criterion for a xnicher fit*

---

## Description

Computes $-2 \log L + k \log n$, where $n$ is `nobs(object)`. Calls `BIC`.

## Usage

```
## S3 method for class 'xnicher'
BIC(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | a fitted model object for which there exists a `logLik` method to extract the corresponding log-likelihood, or an object inheriting from class `logLik`. |
| `...` | Additional fitted model objects. |

## Value

Numeric scalar, or a `data.frame` when multiple objects are passed.

## Examples

```
fit <- optimize_niche(env_occ = example_env_occ_2d,
                      env_m   = example_env_m_2d,
                      num_starts = 5L, likelihood = "weighted")
BIC(fit)
```

---

`compare_xnicher` *Compare xnicher fits side-by-side via AIC and BIC*

---

## Description

Builds a one-row-per-model summary table for a list of `"xnicher"` fits, including log-likelihood, parameter count, AIC, BIC, deltas relative to the best model, and Akaike weights.

## Usage

```
compare_xnicher(
  ...,
  sort_by = c("AIC", "BIC", "loglik"),
  comparison_basis = c("unpenalised", "penalised")
)
```

## Arguments

| | |
|---|---|
| `...` | Two or more `"xnicher"` objects, optionally named, OR a single named list of such objects. |
| `sort_by` | Character. One of `"AIC"` (default), `"BIC"`, or `"loglik"`: which metric to order rows by. Best model is always at the top. |
| `comparison_basis` | Character, one of `"unpenalised"` (default) or `"penalised"`. Determines which log-likelihood is used as the basis for AIC and BIC. |

- `"unpenalised"`: the bare data log-likelihood $\ell(\hat{\theta})$ returned by `logLik.xnicher`. Standard frequentist convention, and the right basis when comparing **different likelihood families** at fixed penalty knobs.
- `"penalised"`: the optimiser's actual objective, $\ell(\hat{\theta}) - \mathrm{pen}(\hat{\theta})$, i.e. `x$best$loglik`. The right basis when comparing **the same family at different penalty strengths**, since the unpenalised log-likelihood is monotone in $(\lambda_\mu, \lambda_{\log\sigma}, \lambda_\alpha)$ and would always favour $\lambda = 0$. Note: this score is NOT a true frequentist IC – it is the penalised-objective AIC, an effective-fit score.

## Details

All fits MUST have been produced from the same `env_occ`; any mismatch in `nobs` or in the variable-name ordering raises an error, since IC values are not comparable across heterogeneous data.

## Value

A `data.frame` with one row per fit and columns:

`model` Name (or call index) of the fit.

`likelihood` Likelihood family (`x$likelihood`).

`loglik` Un-penalised log-likelihood at convergence.

`df` Number of free parameters.

`nobs` Sample size (occurrences).

`AIC`, `BIC` Information criteria.

`dAIC`, `dBIC` Deltas vs. the minimum.

`weight_AIC` Akaike weights, $\exp(-\Delta_i/2)/\sum_j \exp(-\Delta_j/2)$.

`convergence` ucminfcpp convergence code.

## Caveats

AIC and BIC compare *likelihoods* on the SAME observed data. When comparing weighted vs. presence-only families, both families evaluate $\log L$ on $n_{occ}$ occurrences; the weighted family additionally normalises by an `env_m` integral, so its likelihood is on a different scale. In this implementation, `logLik(weighted)` returns the full weighted-likelihood value (occurrence sum minus log-denominator), which is the right thing for **within-family** comparison (e.g., `weighted` vs. `skew_normal_weighted`) but should be interpreted with care when contrasting weighted with presence-only.

## Examples

```
fit_w   <- optimize_niche(example_env_occ_2d, example_env_m_2d,
                          num_starts = 5L, likelihood = "weighted")
fit_skn <- optimize_niche(example_env_occ_2d, example_env_m_2d,
                          num_starts = 5L, likelihood = "skew_normal_weighted")
compare_xnicher(weighted = fit_w, skew_normal = fit_skn)
```

---

`cvine_cholesky` *Build Cholesky factor of a correlation matrix from C-vine partial correlations*

---

## Description

Constructs the lower-triangular Cholesky factor $\mathbf{L}$ of a $d \times d$ correlation matrix using the C-vine method described in Lewandowski, Kurowicka & Joe (2009). The factor satisfies $\mathbf{R} = \mathbf{L}\mathbf{L}^\top$ with $\mathrm{diag}(\mathbf{R}) = 1$.

## Usage

```
cvine_cholesky(v, d, eta = 1)
```

## Arguments

| | |
|---|---|
| v | Numeric vector of unconstrained reals, one for each C-vine edge. The length must be $d(d-1)/2$. For $d = 2$, a single value is required. The order follows a **level-major** sequence: first all edges of level 1 (i.e., between variable 1 and each later variable), then edges of level 2 (between variable 2 and later variables, conditioned on variable 1), and so on. |
| d | Integer, dimension of the target correlation matrix ($d \geq 1$). |
| eta | Positive numeric shape parameter for the LKJ-C-vine prior (default 1). $\eta = 1$ gives a uniform distribution over correlation matrices; larger values concentrate mass near the identity. |

## Details

The algorithm proceeds in three steps:

1. Each element of v is mapped to the unit interval via the logistic (sigmoid) function, then transformed to a partial correlation on $(-1, 1)$ using the quantile function of a symmetric Beta distribution with shape $\phi_k = \eta + (d - k - 2)/2$ (for level $k$, 0-indexed).
2. The table of partial correlations is converted to unconditional correlations using the Yule–Kendall recursion (vine recursion).
3. For each new row $j$ (starting from $j = 2$), the algorithm solves a triangular system to obtain the first $j - 1$ entries of the row, and sets the diagonal entry to maintain unit row norm.

The resulting $\mathbf{L}$ can be used directly to construct the correlation matrix (`tcrossprod(L)`), or to build a covariance matrix by scaling with standard deviations.

## Value

A $d \times d$ lower-triangular matrix $\mathbf{L}$ with positive diagonal entries such that $\mathbf{L}\mathbf{L}^\top$ is a valid correlation matrix (unit diagonal, positive definite). For $d = 1$, returns a $1 \times 1$ matrix with entry 1.

## References

Lewandowski, D., Kurowicka, D., & Joe, H. (2009). Generating random correlation matrices based on vines and extended onion method. *Journal of Multivariate Analysis*, 100(9), 1989–2001. doi:10.1016/j.jmva.2009.04.008

## Examples

```
# For a 2x2 correlation matrix, we need one parameter
v <- 0.5
L <- cvine_cholesky(v, d = 2, eta = 1)
R <- tcrossprod(L)
print(R)

# For 3 dimensions, we need 3 parameters (d*(d-1)/2 = 3)
v <- c(0.1, -0.2, 0.8)
L <- cvine_cholesky(v, d = 3)
R <- tcrossprod(L)
```

```
all.equal(diag(R), rep(1, 3)) # Should be TRUE
```

---

cv_xnicher *k-fold (or leave-one-out) cross-validation for a xnicher fit*

---

## Description

Refits the same likelihood family + same penalty knobs as the supplied `xnicher` object on each train fold of `env_occ`, warm-started from the full-data $\hat{\theta}$, then scores the held-out fold by evaluating the un-penalised log-likelihood at the fold's converged $\hat{\theta}_{train}$. Returns the **summed** held-out log-likelihood across folds, which is the standard out-of-sample log-likelihood used to tune ridge regularisation.

## Usage

```
cv_xnicher(
  fit,
  env_occ,
  env_m = NULL,
  type = c("kfold", "loo"),
  k = 5L,
  seed = NULL,
  num_starts_cv = 1L,
  verbose = FALSE,
  ucminf_control = list(maxeval = 500L)
)
```

## Arguments

| | |
|---|---|
| `fit` | A `"xnicher"` object returned by `optimize_niche`. |
| `env_occ` | A `matrix` or `data.frame` of occurrences, identical (up to row order) to the one passed to the original `optimize_niche()` call. Validated against `fit$best$env_occ_fingerprint`. |
| `env_m` | A `matrix` / `data.frame` of background points, or `NULL` for presence-only families. Validated against `fit$best$env_m_fingerprint`. |
| `type` | Character. `"kfold"` (default) or `"loo"`. |
| `k` | Integer. Number of folds when `type = "kfold"`; ignored for `"loo"`. Default `5L`. |
| `seed` | Integer or `NULL`. Random seed for fold assignment (`type = "kfold"` only). Default `NULL` (no seed). |
| `num_starts_cv` | Integer. Number of starts per fold. Default `1L` (warm-start from `fit$best$theta`). Set higher for robustness in `"kfold"` mode; for `"loo"` this should essentially always be `1L` since you are running n_occ refits. |
| `verbose` | Logical. Print per-fold progress. |
| `ucminf_control` | Optional list passed to `ucminfcpp::ucminf` for each fold's optimiser; defaults to `list(maxeval = 500L)`. |

## Details

For penalised fits the un-penalised log-likelihood at the converged $\hat{\theta}$ (what logLik.xnicher returns) is monotone in the penalty strengths – the in-sample loglik is always best at $\lambda = 0$. Cross-validation breaks that monotonicity by scoring on data the optimiser did not see, so a properly tuned penalty achieves higher out-of-sample loglik than the unpenalised MLE.

## Value

A list with class "xnicher_cv":

cv_loglik Summed held-out log-likelihood across folds.

cv_loglik_mean Per-occurrence average: cv_loglik / nobs.

per_fold data.frame(fold, n_test, loglik_test, convergence).

type "kfold" or "loo".

k Number of folds (n_occ for LOO).

seed Seed used (or NA_integer_).

## Caveats

* Random k-fold CV assumes the occurrences are exchangeable. For spatially autocorrelated data this can be optimistic; spatial-block CV is left to a future release. * LOO with n_occ large (e.g. Vicugna's 545) is feasible but non-trivial: 545 ucminf refits at ~0.1-1 s each = ~1-10 minutes. Consider type = "kfold", k = 10L as a faster proxy. * Each fold's refit uses ucminfcpp::ucminf starting from fit$best$theta. If the per-fold likelihood is not unimodal (rare on dense data, common with skew families on sparse data), bump num_starts_cv or use the original optimize_niche() multistart machinery.

## See Also

compare_xnicher for AIC/BIC-based comparisons.

## Examples

```
fit_w <- optimize_niche(
  env_occ = example_env_occ_2d, env_m = example_env_m_2d,
  num_starts = 5L, breadth = 0.45, likelihood = "weighted",
  prior_mu_lambda = 1.0
)
cv_xnicher(fit_w, example_env_occ_2d, example_env_m_2d,
          type = "kfold", k = 5L, seed = 42L)
```

example_env_m_2d *Samples of environmental data from M hypothesis to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with two variables containing points contains 2 bioclimatic variables*

## Description

Samples of environmental data from M hypothesis to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with two variables containing points contains 2 bioclimatic variables

## Usage

```
example_env_m_2d
```

## Format

A data frame with 10000 rows and 2 variables:

**bio1WH** temperature, in C

**bio12WH** precipitation, in mm

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
head(example_env_m_2d)
```

example_env_m_3d *Samples points from M hypothesis to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with three columns containing extracted information from 3 bioclimatic variables*

## Description

Samples points from M hypothesis to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with three columns containing extracted information from 3 bioclimatic variables

## Usage

```
example_env_m_3d
```

## Format

A data frame with 10000 rows and 3 variables:

**bio1WH** Temperature, in C

**bio12WH** Precipitation, in mm

**bio19WH** Precipitation of Coldest Quarter, in mm

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
head(example_env_m_3d)
```

---

`example_env_occ_2d` *Species occurrence points to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with two variables containing points contains 2 bioclimatic variables*

---

## Description

Species occurrence points to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with two variables containing points contains 2 bioclimatic variables

## Usage

```
example_env_occ_2d
```

## Format

A data frame with 73 rows and 2 variables:

**bio1WH** temperature, in C

**bio12WH** precipitation, in mm

## Source

doi:10.1016/j.ecolmodel.2021.109823

## Examples

```
head(example_env_occ_2d)
```

`example_env_occ_3d` *Species occurrence points to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with three variables containing points contains 3 bioclimatic variables*

## Description

Species occurrence points to estimate negative log likelihood from Abeillia abeillei presence points. This is a hummingbird example. A dataset with three variables containing points contains 3 bioclimatic variables

## Usage

```
example_env_occ_3d
```

## Format

A data frame with 73 rows and 3 variables:

**bio1WH** temperature, in C

**bio12WH** precipitation, in mm

**bio19WH** Precipitation of Coldest Quarter, in mm

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
head(example_env_occ_3d)
```

`example_mu_vec` *Example of a vector of the center of an ellipsoid from two environmental variables. vector of length 2 corresponding to the centroid of an ellipsoid*

## Description

Example of a vector of the center of an ellipsoid from two environmental variables. vector of length 2 corresponding to the centroid of an ellipsoid

## Usage

```
example_mu_vec
```

## Format

An object of class `numeric` of length 2.

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
print(example_mu_vec)
```

---

`example_occ_df` *Species occurrence points from Abeillia abeillei presence points after download and clean from GBIF. This is a hummingbird example. A dataset with three variables. Contains scientific name, longitude and latitude.*

---

## Description

Species occurrence points from Abeillia abeillei presence points after download and clean from GBIF. This is a hummingbird example. A dataset with three variables. Contains scientific name, longitude and latitude.

## Usage

```
example_occ_df
```

## Format

A data frame with 73 rows and 3 variables:

**species** Species name

**lon** Decimal longitude geographical coordinate, in degrees

**lat** Decimal latitude geographical coordinate, in degrees

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
head(example_occ_df)
```

`example_s_mat` *Example of a covariance matrix of an ellipsoid from two environmental variables. The 2 x 2 matrix corresponded to a positive semi-definite matrix. In two dimensions encodes the rotation (orientation) and scaling of an ellipse.*

## Description

Example of a covariance matrix of an ellipsoid from two environmental variables. The 2 x 2 matrix corresponded to a positive semi-definite matrix. In two dimensions encodes the rotation (orientation) and scaling of an ellipse.

## Usage

```
example_s_mat
```

## Format

An object of class `matrix` (inherits from `array`) with 2 rows and 2 columns.

## Source

<doi:10.1016/j.ecolmodel.2021.109823>

## Examples

```
print(example_s_mat)
```

`example_vicugna` *Samples of environmental data from M hypothesis to estimate negative log likelihood from Vicugna vicugna.*

## Description

Samples of environmental data from M hypothesis to estimate negative log likelihood from Vicugna vicugna.

## Usage

```
example_vicugna
```

## Format

A list with 4 elements:

**env_m** Data frame with two columns and 10 000 rows each representing bio1 and bio12 of the accessibility area of the species

**env_occ** Data frame with two columns and 545 rows each representing environmental information bio1 and bio12 associated with the occurrence the species

**coords_m** Data frame with geographical coordinates of the accesitibility area (M).

**coords_occ** Data frame with geographical coordinates of the occurrence points.

## Examples

```
head(example_vicugna$env_occ)
```

---

# geom_xnicher_background

*Background environment layer*

---

## Description

A self-contained `ggplot2` layer that draws the environmental background `env_m` as a faint point cloud in 2-D environmental space. Designed to be composed with `geom_xnicher_occ` and `geom_xnicher_ellipse` via the `+` operator.

## Usage

```
geom_xnicher_background(
  env_m,
  var_names = NULL,
  size = 0.3,
  alpha = 0.25,
  colour = "grey60",
  ...
)
```

## Arguments

| | |
|---|---|
| `env_m` | Matrix or data frame with at least two columns. Only the first two columns are used (positional indexing). |
| `var_names` | Optional character vector of length 2 used to name the columns of the layer's internal data frame. Defaults to `c("x1", "x2")`. |
| `size, alpha, colour` | Aesthetic parameters; defaults are tuned for a discreet background layer. |
| `...` | Additional fixed parameters passed to the underlying `ggplot2::geom_point`. |

## Value

A ggplot2 layer.

## Examples

```
library(ggplot2)
ggplot() + geom_xnicher_background(example_env_m_2d)
```

---

geom_xnicher_ellipse *Niche-ellipse layer derived from a fitted* xnicher *model*

---

## Description

Builds a self-contained ggplot2 layer that traces one or more iso-suitability ellipses implied by a fitted 2-D xnicher model. The ellipse for level $\alpha$ is the contour $S(x) = \alpha$, equivalently $(x - \mu)^\top \Sigma^{-1}(x - \mu) = -2 \log \alpha$ when level_type = "suitability", or the chi-squared confidence ellipse $(x - \mu)^\top \Sigma^{-1}(x - \mu) =$ qchisq$(\alpha, df = 2)$ when level_type = "chisq".

## Usage

```
geom_xnicher_ellipse(
  model,
  level = c(0.95, 0.5, 0.05),
  level_type = c("suitability", "chisq"),
  n = 200L,
  linewidth = 0.6,
  colour = "firebrick",
  ...
)
```

## Arguments

| | |
|---|---|
| model | A xnicher object with length(model$var_names) == 2 (or, for legacy fits without var_names, a 2-D fit). |
| level | Numeric vector of contour levels in $(0, 1]$. Default c(0.95, 0.5, 0.05) (paper-aligned: core / common / edge iso-suitability contours). |
| level_type | Either "suitability" (the default) or "chisq". See Details. |
| n | Integer number of points around each ellipse. Default 200. |
| linewidth | Path linewidth (or size on ggplot2 < 3.4.0). |
| colour | Path colour. |
| ... | Additional fixed parameters passed to geom_path. |

### Details

For `length(level) > 1` the layer carries one column `level` grouping the closed ellipse paths so a single `geom_path` draws them all. Map e.g. `linetype = factor(level)` downstream to distinguish them visually.

### Value

A `ggplot2` layer.

### Examples

```
library(ggplot2)
fit <- optimize_niche(
  env_occ = example_env_occ_2d,
  env_m   = example_env_m_2d,
  num_starts = 5L
)
ggplot() +
  geom_xnicher_background(example_env_m_2d) +
  geom_xnicher_occ(example_env_occ_2d) +
  geom_xnicher_ellipse(fit)
```

---

## `geom_xnicher_isosuitability` *Iso-suitability contour layer derived from a fitted* `xnicher` *model*

---

### Description

Builds a self-contained `ggplot2` layer that draws iso-suitability contours, i.e. level sets $S(x) = c$ of the fitted niche suitability function over a 2-D environmental grid. Unlike `geom_xnicher_ellipse`, which traces analytical ellipses and is therefore tied to the Gaussian (multivariate normal) niche geometry, this layer evaluates the model's *actual* suitability function on a grid and contours the result. It works correctly for **any** likelihood family supported by `optimize_niche()`, including the skew-normal families (`"skew_normal"` and `"skew_normal_weighted"`) where the iso-suitability sets are *not* ellipses.

### Usage

```
geom_xnicher_isosuitability(
  model,
  level = c(0.95, 0.5, 0.05),
  n = 121L,
  expand = 0.1,
  linewidth = 0.6,
  colour = "firebrick",
  ...
)
```

## Arguments

| | |
|---|---|
| `model` | A `xnicher` object with `length(model$var_names) == 2` (or, for legacy fits without `var_names`, a 2-D fit). |
| `level` | Numeric vector of contour levels in $(0, 1]$. Default `c(0.95, 0.5, 0.05)`. |
| `n` | Integer grid resolution per axis. Default 121 (i.e. a 121 x 121 grid). Higher gives smoother contours at proportionally higher cost. |
| `expand` | Numeric scalar in $(0, 1)$. The grid spans $\mu \pm (1 + \texttt{expand}) \cdot k \cdot \sigma_{\text{eff}}$ along each axis, where $k = 4$ matches the typical 4-sigma support of a Gaussian niche and $\sigma_{\text{eff}}$ is the marginal standard deviation. Default `0.1` (10% padding). |
| `linewidth` | Path linewidth (or `size` on ggplot2 < 3.4.0). |
| `colour` | Path colour. |
| `...` | Additional fixed parameters passed to `geom_contour`. |

## Value

A `ggplot2` layer.

## Why "iso-suitability" and not just "ellipse"

For a Gaussian niche, the suitability function $S(x) \propto \exp(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu))$ has elliptical level sets; the family of contours $S(x) = c$ traces nested concentric ellipses around $\mu$. For a skew-normal niche $S(x) \propto \phi_2(x; \mu, \Sigma)\, \Phi(\alpha^\top \omega^{-1}(x-\mu))$ the $\Phi(\cdot)$ factor breaks the ellipse symmetry: contours bunch on the side that $\alpha$ pulls suitability toward, and stretch on the opposite side. The contour at level $c$ is still a closed curve enclosing high-suitability environments, but it is no longer an ellipse and has no closed-form parameterisation. The only honest visualisation is to evaluate $S(x)$ on a dense 2-D grid and contour the result – which is exactly what this layer does.

## Suitability normalisation

Suitability is always normalised so that $S(\mu^*) = 1$ at the modal centre: for the Gaussian families that is the centre $\mu$; for the skew-normal it is the location parameter $\mu$ from the SN $(\mu, \Sigma, \alpha)$ parameterisation (Azzalini & Capitanio 1999), which is generally *not* the global suitability maximum but is a well-defined reference point. The contour level `level = 0.5` therefore always means "regions where the niche is at least 50% as suitable as the reference centre".

## References

Azzalini, A. & Capitanio, A. (1999). Statistical applications of the multivariate skew normal distribution. *Journal of the Royal Statistical Society, Series B*, **61**(3), 579–602.

## Examples

```
library(ggplot2)
fit <- optimize_niche(
  env_occ    = example_env_occ_2d,
  env_m      = example_env_m_2d,
  num_starts = 5L,
  likelihood = "skew_normal_weighted"
```

```
)
ggplot() +
  geom_xnicher_background(example_env_m_2d) +
  geom_xnicher_occ(example_env_occ_2d) +
  geom_xnicher_isosuitability(fit)
```

---

`geom_xnicher_occ` *Occurrence-points layer*

---

## Description

A self-contained `ggplot2` layer that draws an occurrence cloud `env_occ` in 2-D environmental space.

## Usage

```
geom_xnicher_occ(
  env_occ,
  var_names = NULL,
  size = 1.2,
  alpha = 1,
  colour = "black",
  shape = 16,
  ...
)
```

## Arguments

| | |
|---|---|
| `env_occ` | Matrix or data frame with at least two columns. Only the first two columns are used. |
| `var_names` | Optional character vector of length 2 used to name the columns of the layer's internal data frame. |
| `size`, `alpha`, `colour`, `shape` | Aesthetic parameters. |
| `...` | Additional fixed parameters passed to `geom_point`. |

## Value

A `ggplot2` layer.

## Examples

```
library(ggplot2)
ggplot() +
  geom_xnicher_background(example_env_m_2d) +
  geom_xnicher_occ(example_env_occ_2d)
```

## get_ellipsoid_pars *Get ellipsoid parameters. A function to compute average and the inverse of covariance matrix from environmental data*

### Description

Get ellipsoid parameters. A function to compute average and the inverse of covariance matrix from environmental data

### Usage

```
get_ellipsoid_pars(env)
```

### Arguments

| | |
|---|---|
| env | A data frame containing environmental variables |

### Value

A list with computed average of environmental variables and the covariance matrix

### Examples

```
get_ellipsoid_pars(example_env_occ_2d)
```

## habitat_suitability *Tiled habitat-suitability map from an environmental* **terra** *stack*

### Description

Evaluates the standardized multivariate-normal density of Jimenez et al. (2022, Eq. 2) at every cell of an environmental SpatRaster, returning a one-layer raster of suitability values in $(0, 1]$ (or, optionally, log-suitability in $(-\infty, 0]$).

### Usage

```
habitat_suitability(
  param,
  env,
  output = "",
  overwrite = FALSE,
  return_log = FALSE,
  threads = RcppParallel::defaultNumThreads(),
  wopt = list()
)
```

## Arguments

| | |
|---|---|
| param | Named list with components<br>mu Numeric vector of length p — niche centroid.<br>Sigma Symmetric positive-definite p x p matrix.<br>The Cholesky factor of Sigma is computed once before the block loop. If chol() fails (rank-deficient Sigma), the function emits a warning and returns a raster of NA. |
| env | A multi-layer SpatRaster: one layer per environmental variable, in the same order as param$mu. Layer names are preserved on the input but are not used by this function — variable matching is positional. Use predict.xnicher when you need name-based matching. |
| output | Character. File path for the output GeoTIFF. The empty string "" (default) returns an in-memory SpatRaster. |
| overwrite | Logical. If TRUE and output exists, it is overwritten. Default FALSE. |
| return_log | Logical. If FALSE (default) the output is suitability $S(x) \in (0, 1]$. If TRUE the output is $\log S(x) \leq 0$. |
| threads | Integer. Number of parallel threads handed to the inner C++ kernel. Default RcppParallel::defaultNumThreads(). |
| wopt | List. Additional write options forwarded to writeStart. |

## Details

Memory is bounded by the largest block selected by terra's memory manager: continental-scale rasters are processed without ever materialising the full grid in R. The per-cell computation is run by niche_suitability_cpp, an **RcppParallel** kernel that takes raw pointers into terra's column-major buffer.

The function follows the streaming I/O pattern from *xsdm-devel*'s recipe 03:

1. terra::readStart on env, paired with an on.exit(terra::readStop);
2. terra::writeStart on the output, paired with an on.exit(terra::writeStop);
3. for each block i: read each variable's tile via terra::readValues(..., mat = TRUE), pack into a flat column-major numeric vector (n_tile x p), mask NA cells, compact, call niche_suitability_cpp, scatter the result back, and terra::writeValues.

## Value

A one-layer SpatRaster named "suitability" (or "log_suitability" when return_log = TRUE). Returned invisibly when output is non-empty.

## See Also

niche_suitability_cpp, predict.xnicher.

## Examples

```
## Hand-built (mu, Sigma):
r1 <- terra::rast(nrows = 10, ncols = 10)
r2 <- terra::rast(nrows = 10, ncols = 10)
terra::values(r1) <- rnorm(100)
terra::values(r2) <- rnorm(100)
ex <- c(r1, r2)
names(ex) <- c("bio1", "bio12")
habitat_suitability(
  param = list(mu = c(0, 0), Sigma = diag(2)),
  env   = ex
)
```

---

kde_gaussian *Multivariate Gaussian KDE with fixed Scott bandwidth*

---

## Description

Computes a multivariate Gaussian kernel density estimate at the rows of ‘x‘, using ‘data‘ as the reference sample. Bandwidths are set internally via Scott's rule-of-thumb (diagonal). For 2D, a manually optimized version is used; for higher dimensions, an Eigen-based vectorized version is used.

## Usage

```
kde_gaussian(x, data)
```

## Arguments

| | |
|---|---|
| x | Numeric matrix ‘n_eval x p‘ (evaluation points). |
| data | Numeric matrix ‘n_data x p‘ (reference sample for KDE). |

## Value

Numeric vector of length ‘n_eval‘ with the density estimates.

## Examples

```
x    <- as.matrix(example_env_occ_2d)
data <- as.matrix(example_env_m_2d)
dens <- kde_gaussian(x, data)
head(dens)
```

`logLik.xnicher` *Log-likelihood of a fitted xnicher model*

## Description

Returns the **un-penalised** log-likelihood of a `"xnicher"` fit at the converged $\theta$. This is the appropriate quantity for likelihood-based model comparison via `AIC` or `BIC`: even when `optimize_niche` minimises a ridge-penalised objective $-\log L(\theta) + \text{penalty}(\theta)$, the comparison metric is the bare $\log L(\theta)$ evaluated at the optimised $\theta$.

## Usage

```
## S3 method for class 'xnicher'
logLik(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | A `"xnicher"` object returned by `optimize_niche`. |
| `...` | Ignored. |

## Details

For penalised fits this means the value returned is **not** the same as `x$best$loglik`: `best$loglik` is the negative of the penalised objective at the optimum (and is monotone in fit quality but not directly comparable across penalty strengths), while `logLik(x)` is the bare data log-likelihood. `logLik(x)` is what AIC/BIC require.

Stored on `x$best$loglik_unpenalised` by `optimize_niche`; older fits without that field are detected and an informative error is raised.

## Value

An object of class `"logLik"` with attributes `df` (number of free parameters) and `nobs` (number of occurrence rows used in the fit).

## Effective degrees of freedom

`df` is reported naively as `length(theta)`: 2p + p(p-1)/2 for the Gaussian families, 3p + p(p-1)/2 for the skew-normal families, 3p + p(p-1)/2 + 1 for the skew-t families. This **ignores** the shrinkage induced by the ridge penalties (`prior_mu_lambda`, `prior_log_sigma_lambda`, `prior_alpha_lambda`). For weak penalties ($\lambda \leq 1$) the bias is small; for stronger penalties the effective df is lower than `length(theta)` so the penalty ($2k$ or $k \log n$) is conservative — IC values will under-favour the more-regularised model. Quantifying effective df via the trace of the influence matrix is left to a future release.

## Examples

```
fit <- optimize_niche(env_occ = example_env_occ_2d,
                      env_m   = example_env_m_2d,
                      num_starts = 5L, likelihood = "weighted")
logLik(fit)
AIC(fit)
BIC(fit)
```

---

`loglik_niche` *Negative log likelihood of an ellipsoid corrected with environmental combinations which come from the area of study (M)*

---

## Description

Negative log likelihood of an ellipsoid corrected with environmental combinations which come from the area of study (M)

## Usage

```
loglik_niche(mu, s_mat, env_occ, env_m, neg = TRUE)
```

## Arguments

| | |
|---|---|
| `mu` | A vector mu of parameters |
| `s_mat` | The covariance matrix from environmental data frame |
| `env_occ` | A data.frame containing the original sample of environmental combinations that correspond to presences |
| `env_m` | A data.frame containing a second random sample of environmental |
| `neg` | Logical. Default TRUE, returns the negative of the likelihood |

## Value

A negative log likelihood value

## Examples

```
loglik_niche(
  mu = example_mu_vec,
  s_mat = example_s_mat,
  env_occ = example_env_occ_2d,
  env_m = example_env_m_2d
)
# Example with log of the likelihood
loglik_niche(
  mu = example_mu_vec,
  s_mat = example_s_mat,
```

```
  env_occ = example_env_occ_2d,
  env_m = example_env_m_2d,
  neg = FALSE
)
```

---

`loglik_niche_math_cpp` *Negative log-likelihood (M-restricted, math scale, Cholesky version)*

---

## Description

Computes the negative log-likelihood of the multivariate normal niche model restricted to the set of existing environments $\mathbf{E}(t; G)$ (Jimenez et al. 2019, Eq. 3–5). The density at an occurrence point is normalised by the sum of the density over all background points in M:

## Usage

```
loglik_niche_math_cpp(theta, env_occ, env_m, eta = 1, neg = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `theta` | Numeric vector of unconstrained parameters (math scale): $[\mu, \log\sigma, v]$ of length $2p + p(p-1)/2$. |
| `env_occ` | Data frame or matrix ($n \times p$) of environmental values at presence points. |
| `env_m` | Data frame or matrix ($n_m \times p$) of background environmental values from the accessible area M. |
| `eta` | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| `neg` | Logical. If `TRUE` (default), returns the negative log-likelihood (suitable for minimisation). |
| `...` | Additional arguments (ignored; for compatibility). |

## Details

$$-\log\mathcal{L} = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i-\mu)^\top\Sigma^{-1}(\mathbf{x}_i-\mu) + n\cdot\log\left(\sum_{\mathbf{y}\in M}\exp\left[-\frac{1}{2}(\mathbf{y}-\mu)^\top\Sigma^{-1}(\mathbf{y}-\mu)\right]\right)$$

The $|\Sigma|^{-1/2}$ factor cancels between numerator and denominator (Eq. 3), so it does not appear in the objective. The logsumexp is computed with the max-shift trick for numerical stability.

## Value

A scalar numeric value: the (negative) log-likelihood.

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

## See Also

[optimize_niche()] for multi-start fitting, [loglik_niche_math_presence_only()] for the unconstrained (no M) version.

## Examples

```
theta <- start_theta(example_env_occ_2d)
ll <- loglik_niche_math_cpp(theta,
  env_occ = example_env_occ_2d,
  env_m = example_env_m_2d,
  eta = 1, neg = TRUE
)
print(ll)
```

---

# loglik_niche_math_ip_weighted

*Negative log-likelihood (inverse-probability-weighted normal, math scale)*

---

## Description

Computes the negative log-likelihood of the inverse-probability-weighted (IPW) normal niche model. This model uses $w = 1/\hat{g}$ (the inverse of the background KDE) as importance-sampling weights, applying a Horvitz–Thompson correction so that the denominator approximates the Lebesgue integral $\int f(\mathbf{x})\, d\mathbf{x}$ rather than the $g$-weighted integral $\int f\, g\, d\mathbf{x}$:

## Usage

```
loglik_niche_math_ip_weighted(
  theta,
  env_occ,
  env_m,
  eta = 1,
  neg = TRUE,
  m_subsample = NULL,
  m_kde_subsample = NULL,
  seed = NULL,
  den_idx = NULL,
  kde_idx = NULL,
  precomp_w_den = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `theta` | Numeric vector of unconstrained parameters (math scale): $[\mu, \log\sigma, v]$ of length $2p + p(p-1)/2$. |
| `env_occ` | Data frame or matrix ($n \times p$) of environmental values at presence points. |
| `env_m` | Data frame or matrix ($n_m \times p$) of background environmental values from the accessible area M. |
| `eta` | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| `neg` | Logical. If `TRUE` (default), returns negative log-likelihood. |
| `m_subsample` | Optional integer or fraction. If `den_idx` is not given, this defines the number (or fraction) of rows of `env_m` used for the denominator subsample. |
| `m_kde_subsample` | Optional integer or fraction. If `kde_idx` is not given, this defines the KDE reference subsample size (or fraction). |
| `seed` | Optional integer seed for reproducible subsampling. |
| `den_idx` | Optional integer vector of 1-based row indices for the denominator subset. If provided, no new denominator indices are generated. |
| `kde_idx` | Optional integer vector of 1-based row indices for the KDE reference subset. If provided, no new KDE indices are generated. |
| `precomp_w_den` | Optional numeric vector of precomputed denominator KDE weights. Must match the size of `den_idx`. If provided, KDE for the denominator is not recomputed. |
| `...` | Additional arguments (ignored; provided for compatibility). |

## Details

$$-\log\mathcal{L} = \frac{1}{2}\sum_i q_i + \sum_i \log\hat{g}(\mathbf{x}_i) + n \cdot \mathrm{logsumexp}_j\left[-\frac{1}{2}q_j - \log\hat{g}(\mathbf{y}_j)\right]$$

where $q_i$ is the Mahalanobis distance at occurrence point $\mathbf{x}_i$, and the sum over $j$ runs over background points $\mathbf{y}_j \in M$.

## Value

A scalar numeric value containing the (negative) log-likelihood.

## Data-generating process and motivation

The IPW model is designed for the data-generating process (DGP) of Jimenez et al. (2019), which assumes that presence records are drawn from the fundamental niche density $f(\mathbf{x}; \mu, \Sigma)$ restricted to the environments that actually exist in the accessible area $M$:

$$p(\mathbf{x}_i \mid \text{presence in } M) = \frac{f(\mathbf{x}_i)}{\int_M f(\mathbf{x})\, d\mathbf{x}}.$$

The computational challenge is approximating the denominator $\int_M f\, d\mathbf{x}$. The background sample $\{\mathbf{y}_j\}$ is drawn from geographic grid cells whose density in environmental space is $g(\mathbf{x})$. A naive sum $\frac{1}{K}\sum_j f(\mathbf{y}_j)$ therefore estimates $\int f\, g\, d\mathbf{x}$, not $\int f\, d\mathbf{x}$. Dividing each term by $\hat{g}(\mathbf{y}_j)$ corrects the sampling bias via importance sampling:

$$\int_M f(\mathbf{x})\, d\mathbf{x} = \int_M \frac{f(\mathbf{x})}{g(\mathbf{x})}\, g(\mathbf{x})\, d\mathbf{x} \approx \frac{1}{K}\sum_{j=1}^{K} \frac{f(\mathbf{y}_j)}{\hat{g}(\mathbf{y}_j)}.$$

This is the classical Horvitz–Thompson estimator (Horvitz & Thompson 1952) applied to niche modelling. The resulting likelihood estimates the fundamental niche on uniform environmental space (Lebesgue measure), removing the distortion that arises when common environments in $M$ dominate the denominator.

### Comparison with the `"weighted"` model

The `"weighted"` model (Jimenez & Soberon 2022, Eq. 5/8) uses $w = \hat{g}$, which is the correct likelihood under an alternative DGP where presence intensity is proportional to $f \cdot g$ (the use-availability or resource-selection function model; Warton & Shepherd 2010). The two models answer different ecological questions:

| Model | DGP assumption | Estimates |
|---|---|---|
| `"weighted"` | $\lambda \propto f \cdot g$ | Niche under M-biased observation process |
| `"ip_weighted"` | $f$ restricted to $M$ | Fundamental niche on uniform E-space |

Neither model is universally superior. `"weighted"` is appropriate when observation probability correlates with environmental density (e.g. opportunistic citizen-science records). `"ip_weighted"` is appropriate when sampling effort is approximately uniform across $M$ and the goal is to recover the species' intrinsic tolerances independent of which environments happen to be common.

### Known limitations

1. **Importance-sampling variance.** When $\hat{g}(\mathbf{y}_j) \approx 0$ (e.g. between disconnected environmental patches in a multimodal $M$), the weights $1/\hat{g}$ can be very large and a handful of background points may dominate the denominator. The effective sample size (ESS) should be monitored; low ESS indicates unreliable estimates.
2. **KDE bandwidth dependence.** $\hat{g}$ is computed once using Scott's rule bandwidth. Over- or under-smoothing can distort the weights. An adaptive or cross-validated bandwidth would improve robustness.
3. **Self-regularisation.** The IPW denominator diverges when $\sigma \to \infty$ because rare-environment cells contribute $1/\hat{g} \gg 1$. This acts as a built-in penalty against unrealistically broad niches (no explicit ridge prior is needed), but the effective constraint depends on the tail behaviour of $\hat{g}$ and can be noisy.

Accepts explicit subsampling indices and precomputed KDE weights for high-performance workflows. Internally delegates to `loglik_niche_math_ip_weighted_integrated`.

This function was formerly called `loglik_niche_math_kde_bias_corrected`.

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

Jimenez, L., & Soberon, J. (2022). Weighted-normal model for the fundamental niche. *Ecological Modelling*, 438, 109982.

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *J. Amer. Statist. Assoc.*, 47(260), 663–685.

Warton, D. I., & Shepherd, L. C. (2010). Poisson point process models solve the "pseudo-absence problem" for presence-only data in ecology. *Ann. Appl. Stat.*, 4(3), 1383–1402.

## See Also

[optimize_niche()] for multi-start fitting, [loglik_niche_math_cpp()] for the M-restricted Gaussian model.

## Examples

```
den_idx <- sample.int(nrow(example_env_m_2d), 2000)
kde_idx <- sample.int(nrow(example_env_m_2d), 5000)
pre_w <- kde_gaussian(
  example_env_m_2d[den_idx, ],
  example_env_m_2d[kde_idx, ]
)

loglik_niche_math_ip_weighted(
  theta   = start_theta(example_env_occ_2d),
  env_occ = example_env_occ_2d,
  env_m = example_env_m_2d,
  den_idx = den_idx,
  kde_idx = kde_idx,
  precomp_w_den = pre_w
)
```

---

# loglik_niche_math_ip_weighted_integrated

*Negative log-likelihood (inverse-probability-weighted normal, integrated C++)*

---

## Description

Low-level C++ bridge for the inverse-probability-weighted (IPW) normal niche model. Computes KDE weights and Mahalanobis distances entirely in C++ to minimise R overhead. This is the workhorse called by `loglik_niche_math_ip_weighted`.

## Usage

```
loglik_niche_math_ip_weighted_integrated(
  theta,
  env_occ,
  env_m,
  eta = 1,
  neg = TRUE,
  den_idx = NULL,
  kde_idx = NULL,
  precomp_w_den = NULL
)
```

## Arguments

| | |
|---|---|
| `theta` | Numeric vector of unconstrained parameters (math scale): $[\mu, \log\sigma, v]$ of length $2p + p(p-1)/2$. |
| `env_occ` | Data frame or matrix ($n \times p$) of environmental values at presence points. |
| `env_m` | Data frame or matrix ($n_m \times p$) of background environmental values from the accessible area M. |
| `eta` | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| `neg` | Logical. If `TRUE` (default), returns the negative log-likelihood (suitable for minimisation). |
| `den_idx` | Integer vector of 1-based row indices for the denominator subsample. If `NULL` (default), uses all rows of `env_m`. |
| `kde_idx` | Integer vector of 1-based row indices for the KDE reference subsample. If `NULL`, uses all rows of `env_m`. |
| `precomp_w_den` | Numeric vector of precomputed KDE weights for the denominator points (must match `den_idx` in length). If provided, KDE for the denominator is not recomputed. |

## Details

The objective function is described in detail in `loglik_niche_math_ip_weighted`.

This function was formerly called `loglik_niche_math_kde_bias_corrected_integrated`.

## Value

Scalar numeric: the (negative) log-likelihood.

## See Also

[loglik_niche_math_ip_weighted()] for the user-facing wrapper with automatic subsampling.

## Examples

```
theta <- start_theta(example_env_occ_2d)
ll <- loglik_niche_math_ip_weighted_integrated(
  theta   = theta,
  env_occ = example_env_occ_2d,
  env_m   = example_env_m_2d,
  eta     = 1,
  neg     = TRUE
)
print(ll)
```

---

`loglik_niche_math_presence_only`
*Negative log-likelihood (presence-only, math scale)*

---

## Description

Computes the negative log-likelihood of a multivariate normal niche model using only presence records, with no background correction (Jimenez et al. 2019, Eq. 2 without the M-restricted denominator).

## Usage

```
loglik_niche_math_presence_only(theta, env_occ, eta = 1, neg = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `theta` | Numeric vector of unconstrained parameters (math scale): $[\mu, \log\sigma, v]$ of length $2p + p(p-1)/2$. |
| `env_occ` | Data frame or matrix ($n \times p$) of environmental values at presence points. |
| `eta` | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| `neg` | Logical. If `TRUE` (default), returns the negative log-likelihood (suitable for minimisation). |
| `...` | Additional arguments (ignored; for compatibility). |

## Details

The minimised objective is:

$$-\log\mathcal{L} = \frac{n}{2}\log|\Sigma| + \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)^\top\Sigma^{-1}(\mathbf{x}_i - \mu)$$

The $(2\pi)$ normalisation constant is dropped (does not affect the optimum).

Internally, $\Sigma = LL^\top$ is reconstructed from `theta` via `cvine_cholesky`, and the Mahalanobis distances are computed via a triangular solve $L^{-1}(\mathbf{x}_i - \mu)$.

## Value

Scalar numeric: the (negative) log-likelihood.

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

## See Also

[optimize_niche()] for multi-start fitting, [loglik_niche_math_cpp()] for the M-restricted model.

## Examples

```
theta <- start_theta(example_env_occ_2d)
ll <- loglik_niche_math_presence_only(theta, example_env_occ_2d)
```

---

# niche_ip_weighted *Fit inverse-probability-weighted (IPW) normal niche model*

---

## Description

Fits the inverse-probability-weighted (IPW) normal niche model using a compiled C++ backend via an external pointer for fast evaluation.

## Usage

```
niche_ip_weighted(
  occ,
  M,
  den_idx,
  kde_idx,
  precomp_w_den,
  eta = 1,
  start = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| occ | Numeric matrix ($n \times p$) of environmental values at presence points. |
| M | Numeric matrix ($n_m \times p$) of environmental values from the accessible area $M$. Must have the same columns as occ. |
| den_idx | Integer vector of 1-based row indices selecting the denominator subset. Must have the same length as precomp_w_den. |
| kde_idx | Integer vector of 1-based row indices selecting the KDE reference subset. |

| | |
|---|---|
| `precomp_w_den` | Numeric vector of precomputed KDE weights matching `den_idx` in length. |
| `eta` | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| `start` | A numeric vector (single-start) or list of numeric vectors (multi-start). Use [start_theta()] or [start_theta_multiple()] to generate starting values. |
| `...` | Ignored. Present for wrapper compatibility. |

## Details

This model assumes the data-generating process of Jimenez et al. (2019): presence records are drawn from the fundamental niche density $f(\mathbf{x}; \mu, \Sigma)$ restricted to the environments available in $M$. Because background grid cells are not uniformly distributed in environmental space (their density is $g(\mathbf{x})$), a naive sum over the background would estimate $\int f\, g\, d\mathbf{x}$ rather than the required $\int f\, d\mathbf{x}$. The IPW correction divides each background term by $\hat{g}$ (a kernel density estimate of the environmental density of $M$), yielding a Horvitz–Thompson estimator (Horvitz & Thompson 1952) of the Lebesgue integral.

The objective is the negative log-likelihood:

$$-\log \mathcal{L} = \frac{1}{2}\sum_i q_i + \sum_i \log \hat{g}(\mathbf{x}_i) + n \cdot \mathrm{logsumexp}_j \left[ -\frac{1}{2} q_j - \log \hat{g}(\mathbf{y}_j) \right]$$

The `"weighted"` model (Jimenez & Soberon 2022, Eq. 5/8) uses $w = \hat{g}$ instead, which is the correct likelihood under the alternative use-availability DGP where presence intensity is proportional to $f \cdot g$. See `loglik_niche_math_ip_weighted` for a detailed comparison of both DGP assumptions and known limitations.

This function was formerly called `niche_kde_bias_corrected`.

KDE weights must be precomputed by the user and passed via `precomp_w_den`. No KDE is recomputed inside the optimiser.

Supports both single-start (`start` is a numeric vector) and multi-start (`start` is a list of numeric vectors) optimisation.

## Value

A list with components:

`theta` Best parameter vector.

`value` Negative log-likelihood at the optimum.

`conv` Convergence code (0 = success).

`all_results` Data frame of all starts (multi-start only).

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

Jimenez, L., & Soberon, J. (2022). Weighted-normal model for the fundamental niche. *Ecological Modelling*, 438, 109982.

Horvitz, D. G., & Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *J. Amer. Statist. Assoc.*, 47(260), 663–685.

## See Also

[optimize_niche()] for the unified fitting interface, [loglik_niche_math_ip_weighted()] for the R-level log-likelihood (includes full DGP derivation and known limitations).

## Examples

```
occ <- as.matrix(example_env_occ_3d)
M   <- as.matrix(example_env_m_3d)
set.seed(1)
den_idx <- sample.int(nrow(M), 300L)
kde_idx <- sample.int(nrow(M), 600L)
w_den   <- kde_gaussian(M[den_idx, ], M[kde_idx, ])
theta0  <- start_theta(example_env_occ_3d)
res <- niche_ip_weighted(occ, M, den_idx, kde_idx, w_den, start = theta0)
res$value
```

---

# niche_presence_only *Fit presence-only Gaussian niche model*

---

## Description

Fits a multivariate normal niche model using only presence records, with no background correction (Jimenez et al. 2019, Eq. 2). Uses the compiled C++ backend via an external pointer for fast evaluation.

## Usage

```
niche_presence_only(occ, eta = 1, start = NULL, ...)
```

## Arguments

| | |
|---|---|
| occ | Numeric matrix ($n \times p$) of environmental values at presence points. |
| eta | Numeric scalar, shape parameter for the LKJ C-vine prior on the correlation matrix (default 1 = uniform over correlations). |
| start | A numeric vector (single-start) or list of numeric vectors (multi-start). Use [start_theta()] or [start_theta_multiple()] to generate starting values. |
| ... | Ignored. Present for wrapper compatibility. |

## Details

The minimised objective is the negative log-likelihood of the unconstrained multivariate normal:

$$-\log \mathcal{L} = \frac{n}{2} \log |\Sigma| + \frac{1}{2} \sum_{i=1}^{n} q_i$$

where $q_i = (\mathbf{x}_i - \mu)^\top \Sigma^{-1} (\mathbf{x}_i - \mu)$.

Supports both single-start (`start` is a numeric vector) and multi-start (`start` is a list of numeric vectors) optimisation.

## Value

A list with components:

`theta` Best parameter vector.

`value` Negative log-likelihood at the optimum.

`conv` Convergence code (0 = success).

`all_results` Data frame of all starts (multi-start only).

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

## See Also

[optimize_niche()] for the unified fitting interface, [loglik_niche_math_presence_only()] for the R-level log-likelihood.

## Examples

```
occ    <- as.matrix(example_env_occ_3d)
theta0 <- start_theta(example_env_occ_3d)
res    <- niche_presence_only(occ = occ, start = theta0)
res$value
```

---

`nobs.xnicher` *Number of observations used to fit a xnicher model*

---

## Description

Returns the number of presence rows (`nrow(env_occ)`) that were passed to `optimize_niche`. This is the sample size used by `BIC` and is the conventional choice for the presence-only and weighted families: even though the weighted likelihood includes a denominator over background points, the observed quantities being modeled are the $n_{occ}$ occurrences.

## Usage

```
## S3 method for class 'xnicher'
nobs(object, ...)
```

### Arguments

| | |
|---|---|
| object | A "xnicher" object. |
| ... | Ignored. |

### Value

Integer scalar.

### Examples

```
fit <- optimize_niche(env_occ = example_env_occ_2d,
                      env_m   = example_env_m_2d,
                      num_starts = 5L, likelihood = "weighted")
nobs(fit)
```

---

## optimize_niche *Optimize niche model log-likelihood with multi-start Sobol design*

---

### Description

Runs multi-start optimization over a Sobol low-discrepancy sequence (via **pomp**) of starting points covering the parameter space implied by env_occ and breadth.

### Usage

```
optimize_niche(
  env_occ,
  env_m,
  num_starts = 100L,
  breadth = 0.1,
  likelihood = c("weighted", "ip_weighted", "presence_only", "skew_normal",
   "skew_normal_weighted", "skew_t", "skew_t_weighted", "ncst", "ncst_weighted"),
  grad = c("auto", "analytic", "central", "forward"),
  m_subsample = NULL,
  m_kde_subsample = NULL,
  seed = NULL,
  warm_start = TRUE,
  prior_log_sigma_lambda = 1,
  prior_log_sigma_center = NULL,
  prior_mu_lambda = 1,
  prior_mu_center = NULL,
  prior_alpha_lambda = 0.1,
  control = list(),
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| env_occ | Data frame of environmental values at presence points. |
| env_m | Data frame of background environmental values. Required for likelihood in "ip_weighted", "weighted", "skew_normal_weighted", or "skew_t_weighted"; ignored for "presence_only", "skew_normal", and "skew_t". |
| num_starts | Integer. Number of Sobol starting points. |
| breadth | Numeric in (0, 0.5). Controls the quantile range used to define starting bounds for mu parameters. Default 0.1. |
| likelihood | One of "weighted" (default; paper Eq. 5 + ridge), "ip_weighted" (legacy KDE-bias-corrected formula), "presence_only", "skew_normal" (presence-only multivariate skew-normal), "skew_normal_weighted" (paper Eq. 5 with skew-normal density + ridge prior on log sigma), "skew_t" (presence-only multivariate non-central skew-t via 32-node Gauss-Laguerre quadrature), or "skew_t_weighted" (paper Eq. 5 with NCST density + ridge prior). |
| grad | Gradient strategy: "auto" (default) selects "analytic" for the Gaussian weighted models and "central" otherwise. Force one of c("analytic", "central", "forward") to override. |
| m_subsample, m_kde_subsample | Optional integer or fraction in (0, 1]. Resolved to min(nrow(env_m), 10000) when NULL (default). |
| seed | Optional integer to make subsampling deterministic. |
| warm_start | Logical. When likelihood is "ip_weighted", "weighted", or "skew_normal_weighted", run a quick presence-only fit first and prepend its theta (padded with $\alpha = 0$ for the skew variants) as one extra starting point for the weighted multi-start. Cheap insurance against bad multistart luck on rough or multimodal weighted likelihoods (e.g. when env_m contains regions far from the occurrence cloud); does not replace the Sobol starts. Ignored for likelihood = "presence_only" and "skew_normal". Default TRUE. |
| prior_log_sigma_lambda | Numeric scalar (>= 0), strength of the ridge penalty on $\log \sigma$ used by likelihood = "weighted", "skew_normal_weighted", and "skew_t_weighted". The penalty is $\lambda \sum_k (\log \sigma_k - \log \hat{\sigma}_k)^2$. The centre $\log \hat{\sigma}_k$ defaults to the presence-only fit's $\log \sigma_k$ (shrinks the weighted fit toward the PO fit); see prior_log_sigma_center. Default 1.0 (weak). Larger values keep $\sigma$ closer to the PO scale; 0 reduces the model to pure ML weighted normal (Eq. 5) and is not recommended (subject to Patil & Ord 1976 drift). |
| prior_log_sigma_center | Optional numeric vector of length ncol(env_occ). NULL (default) anchors the centre at the presence-only fit's $\log \sigma$ when warm_start = TRUE (the PO fit is already run to seed the weighted multi-start); falls back to log(apply(env_occ, 2, sd)) when warm_start = FALSE. |
| prior_mu_lambda | Numeric scalar (>= 0), strength of a unit-free ridge penalty on $\mu$ used by the weighted families ("weighted", "skew_normal_weighted", "skew_t_weighted"). The penalty is $\lambda_\mu \sum_k \left((\mu_k - \hat{\mu}_k)/\hat{\sigma}_k\right)^2$ with anchors $\hat{\mu}_k$ and $\hat{\sigma}_k$ from the PO fit |

(see prior_mu_center). Because the penalty is divided by the PO $\sigma_k$, it is scale-free across heterogeneous variables (e.g. bio1 in $^\circ C$ vs bio12 in mm): prior_mu_lambda = 1 allows $\mu$ to drift ~1 PO standard deviation before the penalty pushes back. Default 1.0. Set to 0 for unpenalized maximum likelihood.

prior_mu_center

Optional numeric vector of length ncol(env_occ). NULL (default) anchors the centre at the presence-only fit's $\mu$ when warm_start = TRUE; if warm_start = FALSE and prior_mu_lambda > 0, it must be supplied explicitly.

prior_alpha_lambda

Numeric scalar (>= 0), strength of a ridge penalty $\lambda_\alpha \sum_k \alpha_k^2$ on the skew vector $\alpha$, shrinking toward the Gaussian sub-model. Applies to both presence-only and weighted skew families ("skew_normal", "skew_normal_weighted", "skew_t", "skew_t_weighted"). Fixes the well-known unbounded-MLE pathology of the skew-normal direct parameterization (Azzalini 1985, Pewsey 2000). Default 0.1 (mild). Set to 0 for unpenalized maximum likelihood.

control — Named list of control parameters for ucminfcpp::ucminf_xptr(). Recognized entries:

grad "central" (default)

gradstep c(1e-6, 1e-8)

grtol 1e-4

xtol 1e-8

stepmax 5

maxeval 2000

verbose — Logical. If TRUE, print per-start progress.

... — Additional arguments forwarded to the objective function (e.g. eta).

## Details

The supported likelihood models are:

- "weighted" (default): paper-faithful weighted-normal model. Implements Eq. 5 of Jiménez & Soberón (2022, Ecological Modelling 438:109982) – pure ML estimation of a weighted normal density on M – plus a weakly-informative ridge prior on $\log \sigma$ that prevents the well-known Patil & Ord (1976) $\sigma \to \infty$ drift in the un-regularised MLE. The ridge strength is controlled by prior_log_sigma_lambda (default 1.0, weak); set it to 0 for pure paper Eq. 5 (not recommended on multimodal M).
- "ip_weighted": inverse-probability-weighted (IPW) normal model. Assumes the Jimenez et al. (2019) DGP in which presences are drawn from $f$ restricted to $M$. Because background grid cells sample environmental space with density $g(\mathbf{x})$, a naive denominator sum estimates $\int f\, g$ rather than $\int f$. The IPW correction uses $w = 1/\hat{g}$ (the inverse of the background KDE) as importance-sampling weights (Eq. 8), yielding a Horvitz–Thompson estimator of the Lebesgue integral and recovering the fundamental niche on uniform environmental space. Empirically stable, with built-in self-regularisation against $\sigma \to \infty$ drift (no ridge prior required). The "weighted" model is preferred when observation intensity correlates with environmental density (e.g. opportunistic records); "ip_weighted" is preferred when sampling effort is approximately uniform across $M$ and the goal is to estimate intrinsic tolerances. See

loglik_niche_math_ip_weighted for a detailed comparison and known limitations. Formerly called "kde_bias_corrected"; the old name is accepted as a deprecated alias.

- "presence_only": model using only presence points, no background correction. Unimodal likelihood, useful as a sanity check or to seed the weighted multistart (see warm_start).
- "skew_normal": presence-only fit of a multivariate skew-normal niche (Azzalini & Capitanio 1999, J. R. Stat. Soc. Ser. B 61(3): 579-602):

$$S(x) \propto \phi_p(x - \mu; \Sigma)\, \Phi\left(\sum_k \alpha_k\, (x_k - \mu_k)/\sigma_k\right).$$

  Adds a length-p skewness vector $\alpha$ to the existing $(\mu, \Sigma)$ parameters. $\alpha = 0$ recovers the symmetric Gaussian niche exactly. Sobol-start machinery samples $\alpha_k$ in $[-3, 3]$ (Azzalini & Capitanio 1999, Sec. 5).
- "skew_normal_weighted": paper Eq. 5 with the SN density in place of the Gaussian, plus the same ridge prior on $\log \sigma$ used by "weighted". Defaults inherit from "weighted".
- "skew_t": presence-only fit of the multivariate non-central skew-t (NCST) density (Branco & Dey 2001, J. Multivariate Analysis 79(1):99-113):

$$T = \mu + X\sqrt{r/Y}, \quad X \sim SN_p(0, \Sigma, \alpha), Y \sim \chi^2_r.$$

  Adds a single degrees-of-freedom parameter log_r on top of the skew-normal layout. Smaller $r$ = heavier tails; $r \to \infty$ recovers "skew_normal". The marginal density has no closed form, so it is computed by 32-node Gauss-Laguerre quadrature on the $\chi^2_r$ mixing variable. Sobol-start machinery samples log_r in $[\log 2, \log 100]$.
- "skew_t_weighted": paper Eq. 5 with the NCST density in place of the Gaussian, plus the same ridge prior on $\log \sigma$ used by "weighted". Defaults inherit from "weighted".

For multimodal or long-tailed env_m, consider z-scoring env_occ and env_m so all variables have comparable spread (e.g. z-score or quantile-rank); the ridge prior is then uniform across axes.

## Value

An object of class "xnicher" (see new_xnicher).

## Optimization backend

optimize_niche() optimizes via ucminfcpp::ucminf_xptr() with a compiled C++ objective built by create_niche_obj_ptr(). The full (theta -> mu, log sigma, v) unpacking, cvine_cholesky, log-likelihood, and gradient are evaluated in pure C++ with no R-callback overhead. For the weighted likelihood, gradient mode "analytic" uses closed-form derivatives over mu and log sigma and central finite differences over the C-vine partial-correlation block.

## KDE sampling (weighted model)

The KDE that weights the M-background depends only on the environmental data (not on theta), so the weights are computed once before optimization. By default, the KDE reference (m_kde_subsample) and the denominator subset (m_subsample) are both capped at 10\,000 distinct background combinations – the maximum the model can usefully exploit even for very large rasters. A floor of max(500, 50 * 2^p) rows is used as a heuristic minimum representative sample (Silverman 1986; Wand & Jones 1995); below this floor a warning() is emitted.

## References

Jimenez, L., Soberon, J., Christen, J. A., & Soto, D. (2019). On the problem of modeling a fundamental niche from occurrence data. *Ecological Modelling*, 397, 109823.

Jimenez, L., & Soberon, J. (2022). Weighted-normal model for the fundamental niche. *Ecological Modelling*, 438, 109982.

Azzalini, A., & Capitanio, A. (1999). Statistical applications of the multivariate skew normal distribution. *J. R. Stat. Soc. B*, 61(3), 579–602.

Branco, M. D., & Dey, D. K. (2001). A general class of multivariate skew-elliptical distributions. *J. Multivariate Anal.*, 79(1), 99–113.

## See Also

[loglik_niche_math_cpp()], [loglik_niche_math_presence_only()], [loglik_niche_math_ip_weighted()].

## Examples

```
result <- optimize_niche(
  env_occ    = example_env_occ_2d,
  env_m      = example_env_m_2d,
  num_starts = 5L,
  breadth    = 0.1,
  likelihood = "weighted"
)
print(result)
assess(result)
```

---

`optimize_niche_xptr` *Optimize niche model using a compiled XPtr backend*

---

## Description

Runs ucminfcpp::ucminf_xptr() over a compiled C++ objective function created by create_niche_obj_ptr(). Supports single-start and multi-start optimization. Ensures that all starting vectors are numeric doubles and finite.

## Usage

```
optimize_niche_xptr(
  start = NULL,
  xptr,
 control = ucminfcpp::ucminf_control(grad = "central", gradstep = c(1e-06, 1e-08),
    maxeval = 200),
  multi_start = FALSE
)
```

### Arguments

| | |
|---|---|
| start | Numeric vector (single start) or list of numeric vectors (multi-start). |
| xptr | External pointer created by create_niche_obj_ptr(). |
| control | List of control parameters for ucminfcpp::ucminf_control(). |
| multi_start | Logical; TRUE if multiple starts are supplied. |

### Value

A list with:

- par — best parameter vector
- value — negative log-likelihood
- conv — convergence code
- all_results — (multi-start only) table of all runs

### Examples

```
## optimize_niche() uses optimize_niche_xptr() internally;
## call it directly for full control over each optimization run:
fit <- optimize_niche(
  env_occ    = example_env_occ_3d,
  env_m      = example_env_m_3d,
  num_starts = 3L,
  likelihood = "ip_weighted"
)
fit$best$value
```

---

## predict.xnicher *Habitat-suitability raster from a fitted* xnicher *object*

---

### Description

predict method for objects of class "xnicher" returned by optimize_niche. Reconstructs the niche centroid $\mu$ and covariance $\Sigma$ from the optimizer's best$theta (mu, log_sigma, v) parameterization, then evaluates the standardized Gaussian suitability map of Jimenez et al. (2022, Eq. 2) over an environmental SpatRaster.

### Usage

```
## S3 method for class 'xnicher'
predict(
  object,
  env,
  ...,
```

```
  return_log = FALSE,
  threads = RcppParallel::defaultNumThreads(),
  output = "",
  overwrite = FALSE,
  wopt = list()
)
```

## Arguments

| | |
|---|---|
| object | A "xnicher" object returned by optimize_niche. |
| env | A multi-layer SpatRaster, one layer per environmental variable. When object$var_names is non-NULL (the default for fits produced by optimize_niche() on a column-named env_occ), the layers of env are reordered to match var_names and any name not present in env produces an error. When object$var_names is NULL the layers are used in their existing order. This lets you call predict on a future-climate SpatRaster that contains the same variables in any order, including extra layers. |
| ... | Currently ignored. |
| return_log | Logical. FALSE (default) returns suitability in $(0, 1]$; TRUE returns log-suitability in $(-\infty, 0]$. |
| threads | Integer. Threads for the inner C++ kernel. Default RcppParallel::defaultNumThreads(). |
| output | Character. File path for the output GeoTIFF. The empty string "" (default) returns an in-memory raster. |
| overwrite | Logical. Forwarded to habitat_suitability. |
| wopt | List. Forwarded to habitat_suitability. |

## Details

The same Gaussian suitability surface is used for every fitted family: "weighted", "ip_weighted", "presence_only", and the skew-normal / skew-t variants all project the fitted $(\mu, \Sigma)$ geometry. For skew fits, the skewness and tail parameters affect fitting but are not used by this Gaussian projection.

Internally:

1. $\mu = \theta_{1:p}$.
2. $\sigma = \exp(\theta_{p+1:2p})$.
3. C-vine partial-correlation block $v = \theta_{2p+1:\,length(\theta)}$ is mapped to a correlation Cholesky factor via cvine_cholesky.
4. $L = \mathrm{diag}(\sigma)\, L_{corr}$, $\Sigma = LL^\top$.

## Value

A one-layer SpatRaster named "suitability" (or "log_suitability"). Returned invisibly when output is non-empty.

## See Also

habitat_suitability, niche_suitability_cpp, cvine_cholesky.

## Examples

```
fit <- optimize_niche(
  env_occ    = example_env_occ_2d,
  env_m      = example_env_m_2d,
  num_starts = 5L,
  likelihood = "weighted"
)
r1 <- terra::rast(nrows = 10, ncols = 10)
r2 <- terra::rast(nrows = 10, ncols = 10)
terra::values(r1) <- rnorm(100)
terra::values(r2) <- rnorm(100)
env <- c(r1, r2)
names(env) <- colnames(example_env_occ_2d)
suit <- predict(fit, env)
```

---

`print.xnicher` *Print a xnicher object*

---

## Description

Displays a concise summary of a `"xnicher"` optimization result, including the likelihood type, number of starts, convergence count, and the best log-likelihood achieved.

## Usage

```
## S3 method for class 'xnicher'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | A `"xnicher"` object returned by `optimize_niche`. |
| `...` | Ignored. |

## Value

Invisibly returns `x`.

## Examples

```
result <- optimize_niche(
  env_occ    = example_env_occ_2d,
  env_m      = example_env_m_2d,
  num_starts = 5L,
  likelihood = "ip_weighted"
)
print(result)
```

## rhipicephalus — *Rhipicephalus microplus occurrence and background environment data*

### Description

Example dataset for *Rhipicephalus microplus* (cattle tick) presence points and a random background sample inside the accessible area (M).

### Usage

```
rhipicephalus
```

### Format

A list with four data frames:

**coords_occ** Geographic coordinates (lon, lat) of presence points.

**env_occ** Environmental values of bio1 and rh at presence points.

**coords_m** Geographic coordinates (lon, lat) of background points in M.

**env_m** Environmental values of bio1 and rh at background points.

### Source

Occurrence and background data prepared from the Garrapatas dataset; raster layers are annual mean temperature (bio1, in °C) and relative humidity (rh, in percent).

### Examples

```
str(rhipicephalus)
head(rhipicephalus$env_occ)
```

## rhipicephalus_occ — *Rhipicephalus microplus occurrence coordinates*

### Description

Presence coordinates for *Rhipicephalus microplus*.

### Usage

```
rhipicephalus_occ
```

## Format

A data frame with 334 rows and 3 variables:

**species** Species scientific name

**lon** Decimal longitude, in degrees

**lat** Decimal latitude, in degrees

## Source

Garrapatas dataset.

## Examples

```
head(rhipicephalus_occ)
```

---

# start_theta *Starting values for niche model on math scale*

---

## Description

Starting values for niche model on math scale

## Usage

```
start_theta(env_occ, skew = FALSE, skew_t = FALSE)
```

## Arguments

| | |
|---|---|
| env_occ | Data frame with environmental values at presence points. |
| skew | Logical. If TRUE, append p zeros for the skew parameters (alpha_1, ..., alpha_p). The Gaussian-only start (skew = FALSE, the default) is used for "presence_only", "weighted", and "ip_weighted"; the skew start is used for "skew_normal" and "skew_normal_weighted". |
| skew_t | Logical. If TRUE, append log(10) for the skew-t degrees-of-freedom parameter log_r after the alpha block (used for "skew_t" and "skew_t_weighted"). Implies skew = TRUE. |

## Value

Numeric vector of starting values for ‘theta‘.

## Examples

```
start_theta(example_env_occ_2d)
start_theta(example_env_occ_2d, skew = TRUE)
start_theta(example_env_occ_2d, skew_t = TRUE)
```

start_theta_multiple *Generate multiple starting points for niche model optimization*

## Description

Creates a set of starting parameter vectors on the math scale for use with the log-likelihood functions. Ensures all starting values are strictly numeric and finite, even if env_data is supplied as a data frame.

## Usage

```
start_theta_multiple(
  env_data,
  num_starts = 100,
  quant_vec = c(0.1, 0.5, 0.9),
  method = "sobol",
  skew = FALSE,
  skew_t = FALSE
)
```

## Arguments

| | |
|---|---|
| env_data | Environmental data (matrix or data frame). Must be numeric. |
| num_starts | Integer, number of starting points. |
| quant_vec | Quantiles for mu ranges. |
| method | "sobol" (Sobol design) or "uniform". |
| skew | Logical. If TRUE, append p skewness parameters (alpha_1, ..., alpha_p) to each starting vector with the default range [-3, 3]. Use this for the "skew_normal" / "skew_normal_weighted" likelihoods. Default FALSE. |
| skew_t | Logical. If TRUE, additionally append a single log_r parameter (default range [log 2, log 100]) after the alpha block. Use for "skew_t" / "skew_t_weighted". Implies skew = TRUE. |

## Value

A data frame of dimension num_starts × num_parameters.

## Examples

```
starts <- start_theta_multiple(
  env_data   = example_env_occ_2d,
  num_starts = 5L,
  method     = "uniform"
)
dim(starts)
```

# Index