

Package: xsdm (via r-universe)

October 8, 2026

Type Package

Title Demographic Approach to Species Distribution Model

Version 1.0.1

Description Integrates concepts of stochastic demography into species distribution modelling. The main approach maximizes a likelihood function based on environmental information and presence/absence records. This is used to reconstruct species' fundamental ecological niches and to project their potential geographic range. Data requirements include species presence/absence records and a timeseries of environmental data.

License AGPL (≥ 3)

Imports checkmate, expm, Rcpp ($\geq 1.1.0$), RcppParallel ($\geq 5.1.10$), stats, terra, tibble, ucminfcpp, furrr, future, future.callr, sobol, purrr

LinkingTo Rcpp, RcppParallel, ucminfcpp

SystemRequirements GNU make, C++17

Encoding UTF-8

Depends R ($\geq 4.1.0$)

LazyData true

LazyDataCompression xz

Suggests clue, knitr, rmarkdown, testthat ($\geq 3.0.0$)

URL <https://xsdm-project.github.io/xsdm/>,
<https://xsdm-project.github.io/xsdm-devel/>

BugReports <https://github.com/xsdm-project/xsdm-devel/issues>

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

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

Repository <https://xsdm-project.r-universe.dev>

Date/Publication 2026-07-01 20:34:31 UTC

RemoteUrl <https://github.com/xsdm-project/xsdm-devel>

RemoteRef HEAD

RemoteSha 93eaa5c52f2ce7649130f3985cb456d1787d3eb6

Contents

xsdm-package	3
bio_to_math	3
build_orthogonal_matrix	5
convert_equivalence_class	6
create_mask	7
create_param_vector_masked	8
dist_between_params	9
env_data_array	11
example_1	11
example_2	13
example_3	14
expit	14
extract_orthogonal_matrix_parameters	15
habitat_suitability	16
interpret_parameters	17
like_ltsg	20
like_neg_ltsg	21
log_prob_detect	22
log_prob_detect_cpp	23
loglmexp	24
loglpexp	25
loglik_bio	26
loglik_bio_cpp	27
loglik_math	28
loglik_math_cpp	31
make_mask_names	32
math_to_bio	33
num_env_var	34
num_par	35
optimize_likelihood	36
profile_likelihood	37
start_parms	39
vsp	40

Index	42
--------------	-----------

xsdm-package

*xsdm: Demographic Approach to Species Distribution Model***Description**

Integrates concepts of stochastic demography into species distribution modelling. The main approach maximizes a likelihood function based on environmental information and presence/absence records. This is used to reconstruct species' fundamental ecological niches and to project their potential geographic range. Data requirements include species presence/absence records and a time-series of environmental data.

Author(s)

Maintainer: Angel Luis Robles Fernandez <a.l.robles.fernandez@gmail.com> ([ORCID](#))

Authors:

- Angel Luis Robles Fernandez <a.l.robles.fernandez@gmail.com> ([ORCID](#))
- Emilio Berti <emilio.berti@idiv.de> ([ORCID](#))
- Daniel C. Reuman <reuman@ku.edu> ([ORCID](#))

See Also

Useful links:

- <https://xsdm-project.github.io/xsdm/>
- <https://xsdm-project.github.io/xsdm-devel/>
- Report bugs at <https://github.com/xsdm-project/xsdm-devel/issues>

bio_to_math

Converts parameters from the biological scale to the math (unconstrained) scale

Description

Given a list with biological-scale parameters ('mu', 'sigltill', 'sigrtill', 'ctil', 'pd', 'o_mat'), returns a named numeric vector on the math scale, in the canonical order produced by 'make_mask_names(p)': 'mu1..mup', 'sigltill1..p', 'sigrtill1..p', 'o_par1..q', 'ctil', 'pd', where 'q = p*(p-1)/2' and 'p = length(mu) = nrow(o_mat)'.

Usage

```
bio_to_math(parms_bio)
```

Arguments

`parms_bio` A named list with entries: `'mu'`, `'sigltil'`, `'sigrttil'`, `'ctil'`, `'pd'`, `'o_mat'`.

Details

Transformations: - `'mu'` : identity - `'sigltil'` : `'log()'` - `'sigrttil'` : `'log()'` - `'ctil'` : identity - `'pd'` : `'logit()'` - `'o_mat'` : lower-triangular parameters recovered via `'extract_orthogonal_matrix_parameters()'` (see Details)

The `'o_mat'` entries are mapped to a vector via the principal matrix logarithm, i.e., one of the (skew-symmetric) matrices S such that `'o_mat = expm(S)'`. The math-scale parameters `'o_par'` are then the strictly lower-triangular elements of S . For `'p = 1'`, there are no `'o_par'` entries. Note, however, that the principal logarithm is not defined for all special orthogonal matrices (even though all such matrices are in the image of the matrix exponential), so this function may fail for some valid `'o_mat'` inputs.

Value

A named numeric vector on the math scale, ordered per `'make_mask_names(p)'`.

See Also

`[math_to_bio()]`, `[make_mask_names()]`, `[build_orthogonal_matrix()]`

Examples

```
## --- p = 1 (no o_par entries) ---
mu1 <- 10
sigltil1 <- 1.2
sigrttil1 <- 0.8
bio_parameters <- list(
  mu      = c(mu1),
  sigltil = c(sigltil1),
  sigrttil = c(sigrttil1),
  ctil    = 0.3,
  pd      = 0.85,
  o_mat   = matrix(1, 1, 1) # 1x1 orthogonal
)
math1 <- bio_to_math(bio_parameters)
# Canonical names
names(math1)
# Back to biological scale
math_parameters <- math_to_bio(math1)
all.equal(math_parameters$mu, bio_parameters$mu)
all.equal(math_parameters$sigltil, bio_parameters$sigltil)
all.equal(math_parameters$sigrttil, bio_parameters$sigrttil)
all.equal(math_parameters$ctil, bio_parameters$ctil)
all.equal(math_parameters$pd, bio_parameters$pd)

## --- p = 2 (includes one o_par) ---
mu2 <- c(11, 5)
sigltil2 <- c(1.1, 1.5)
```

```

sigrtil2 <- c(1.4, 1.3)
ctil2 <- -0.2
pd2 <- 0.9
o_par2 <- 0.25
O2 <- build_orthogonal_matrix(o_par2)
bio_parameters_2d <- list(
  mu      = mu2,
  sigltil = sigltil2,
  sigrtil = sigrtil2,
  ctil    = ctil2,
  pd      = pd2,
  o_mat   = O2
)
math_parameters_2d <- bio_to_math(bio_parameters_2d)
# check canonical name order produced by make_mask_names(2)
identical(names(math_parameters_2d), names(make_mask_names(2)))

```

build_orthogonal_matrix

Build an orthogonal matrix from a real-parameter vector

Description

Constructs a $k \times k$ orthogonal matrix O by exponentiating a skew-symmetric matrix S built by assigning its lower-triangular entries from the input vector. Specifically, the function sets S_{ij} (for $i > j$) from ‘entries’, mirrors it to enforce $S = L - L^\top$, and returns ‘expm::expm(S)’, which is guaranteed orthogonal because $\exp(S)$ is orthogonal whenever S is real skew-symmetric.

Usage

```
build_orthogonal_matrix(entries)
```

Arguments

entries	A numeric vector (possibly ‘NULL’). If ‘NULL’, returns the ‘1 x 1’ identity. Otherwise, its length must be $n = k(k-1)/2$ for some integer $k \geq 2$, supplying the strictly lower-triangular entries of a skew-symmetric generator.
---------	--

Details

The dimension ‘k’ is inferred from ‘length(entries)’ via the relation $n = k(k-1)/2$, so ‘length(entries)’ must equal a triangular number.

- Dimension inference uses $k = \frac{1+\sqrt{1+8n}}{2}$ where $n = \text{length}(\text{entries})$. If k is not an integer, the input is invalid and an error is thrown. - Orthogonality follows from the fact that $S^\top = -S$ implies $\exp(S)^\top \exp(S) = I$. - Note the function actually returns a special orthogonal matrix, i.e., the determinant is +1.

Value

A 'k x k' orthogonal matrix. If 'entries' is 'NULL', returns 'matrix(1, 1, 1)' (identity).

Examples

```
# 1x1 identity (NULL input)
build_orthogonal_matrix(NULL)

# 2x2 orthogonal matrix from one parameter
O2 <- build_orthogonal_matrix(0.0)
all.equal(t(O2) %*% O2, diag(2)) # should be TRUE

# 3x3 example: length(entries) = 3 (= 3*2/2), so k = 3
O3 <- build_orthogonal_matrix(c(0.1, -0.2, 0.3))
all.equal(t(O3) %*% O3, diag(3), tolerance = 1e-10)
```

convert_equivalence_class

Converts a set of parameters to other representatives of the same equivalence class

Description

Model parameters in the biological scale are only determined up to an equivalence class. This function converts a set of parameters to another set of equivalent parameters.

Usage

```
convert_equivalence_class(p, flip, perm)
```

Arguments

p	Named list with entries mu, sigltil, sigrttil, ctil, pd, and o_mat
flip	Vector of binaries corresponding to which columns of o_mat are to have their sign change (which a concomitant switch of the corresponding entries of sigltil and sigrttil). Length must equal the number of columns of o_mat.
perm	Permutation to be applied to the columns of o_mat.

Value

List with entries o_mat, sigltil, and sigrttil

Examples

```
convert_equivalence_class(
  p = example_1$optim_par_list,
  flip = c(1, 0),
  perm = c(1, 2)
)
```

create_mask

*Create a parameter mask aligned with 'make_mask_names()'***Description**

Constructs a named numeric vector whose names and length match the canonical schema returned by 'make_mask_names()'. Optionally fills selected entries from a user-supplied named vector 'mask'. The output is intended for use within downstream functions (e.g., 'loglik_math') that need parameters in a fixed order and with standard names: 'mu1' ... 'mup', 'sigltill1', ..., 'sigltipl', 'sigrtill1', ..., 'sigrtipl', 'ctil', 'pd', and o_mati for i ranging from 1 to $(p^2-p)/2$.

Usage

```
create_mask(mask = NULL, p = 1)
```

Arguments

mask	Named numeric vector (default 'NULL'). Names must be a subset of those produced by 'make_mask_names(p)'. Values are inserted into the corresponding positions; all other entries of the output are 'NA_real_'.
p	A positive integer representing the number of environmental variables to be used in the xsdm model, i.e., <code>dim(env_dat)[3]</code> for the <code>env_dat</code> argument of the <code>loglik_math</code> function.

Value

A named numeric vector of length 'num_par(p)' with names in the canonical order given above. Entries are initialized to 'NA_real_' except for those provided in 'mask'.

See Also

```
[make_mask_names()], [num_par()]
```

Examples

```
# Empty mask for p = 2 (all NA values)
create_mask(p = 2)

# Partially filled mask; unspecified entries remain NA
create_mask(mask = c(mu1 = 11, sigltill1 = Inf, pd = 1, ctil = -2), p = 2)
```

```
# p = 1 has no o_par entries
create_mask(mask = c(mu1 = 7, pd = 0.5), p = 1)
```

```
create_param_vector_masked
```

Create a complete parameter vector with canonical names (no NAs allowed)

Description

Builds the full parameter vector for use within ‘loglik_math’ using canonical names from ‘make_mask_names(p)’ via ‘create_mask(mask = mask, p)’. The output **always** contains all canonical names (length = ‘num_par(p)’). ‘mask’ is applied first (optional), then ‘param_vector’ overrides (required). The final vector must have **no NA values**; if any entry remains ‘NA’, the function throws an error listing which names are missing.

Usage

```
create_param_vector_masked(param_vector, mask = NULL, p)
```

Arguments

param_vector	Named numeric vector (required). Names must be a subset of the canonical names returned by ‘create_mask(mask = NULL, p)’. Values in ‘param_vector’ override those set by ‘mask’.
mask	Named numeric vector (optional). Names must be canonical and are applied before ‘param_vector’.
p	A positive integer representing the number of environmental variables to be used in the xsdm model, i.e., <code>dim(env_dat)[3]</code> for the <code>env_dat</code> argument of the <code>loglik_math</code> function

Details

Canonical names follow ‘loglik_math’ conventions - see Details of that function for the canonical ordering we use.

This is a thin R wrapper around the internal C++ implementation `xsdm:::build_canonical_param_vector_cpp`. The pre-port pure-R implementation is preserved internally as `xsdm:::create_param_vector_masked_r` for parity testing.

Value

A named numeric vector of length ‘num_par(p)’ with **no NA** entries and canonical names in the documented order.

See Also

[`make_mask_names()`], [`create_mask()`], and ‘loglik_math’ (Details).

Examples

```
## --- p = 1 ---
p1 <- 1
# Canonical names typically: mu1, sigltil1, sigrttil1, cttil, pd
pv1 <- c(sigltil1 = 1.0, sigrttil1 = 2.0, cttil = 0.2) # fills remaining slots
mask1 <- c(mu1 = -1, pd = 0.5)
out1 <- create_param_vector_masked(param_vector = pv1, mask = mask1, p = p1)

## --- p = 2 (includes o_par1) ---
p2 <- 2
pv2 <- c(
  sigltil1 = 1.0, sigltil2 = 1.1, sigrttil1 = 2.0, sigrttil2 = 2.2,
  cttil = 0.3, o_par1 = 0.0
)
mask2 <- c(mu1 = 0.1, mu2 = 0.2, pd = 0.05)
out2 <- create_param_vector_masked(param_vector = pv2, mask = mask2, p = p2)

## --- p = 3 (includes o_par1..3) ---
p3 <- 3
pv3 <- c(
  sigltil1 = 1.0, sigltil2 = 1.1, sigltil3 = 1.2,
  sigrttil1 = 2.0, sigrttil2 = 2.1, sigrttil3 = 2.2,
  cttil = 0.4, o_par1 = -0.2, o_par2 = 0.0, o_par3 = 0.15
)
mask3 <- c(mu1 = 0.1, mu2 = 0.2, mu3 = 0.3, pd = 0.01)
out3 <- create_param_vector_masked(param_vector = pv3, mask = mask3, p = p3)
```

dist_between_params	<i>Distance in parameter space between two sets of parameters</i>
---------------------	---

Description

Computes a distance in parameter space between two parameter sets of the xsdm model. The o_mat, sigltil, and sigrttil parameters are only determined up to an equivalence class; this function returns the minimum distance over all equivalence-class representatives of p1, using the Hungarian (Kuhn–Munkres) linear sum assignment algorithm to avoid enumerating every permutation and sign flip. Distance is measured in sum-squared-errors on the biological scale, except that sigltil and sigrttil are inverted before comparison (that is the scale on which distance is most meaningful for those parameters).

Usage

```
dist_between_params(p1, p2, mask = NULL, give_closest_rep = FALSE)
```

Arguments

p1	First set of parameters. May be math-scale (a named numeric vector whose names complement mask) or biological-scale (a named list with entries mu, sigltil, sigrttil, cttil, pd, o_mat). On the math scale a coordinate may be +/-Inf only
----	--

	if it is a sig* or pd entry (the boundary-model case where a shape parameter saturates); sig* are compared on the inverse scale, so an infinite width contributes a finite distance. NA is never allowed.
p2	Second set of parameters; same format options as p1.
mask	Same format as the mask argument to <code>loglik_math</code> and <code>start_parms</code> . Ignored if both p1 and p2 are on the biological scale. Otherwise the names of mask must exactly complement the names of whichever of p1 / p2 is on the math scale.
give_closest_rep	If TRUE, also returns the member of the equivalence class of p1 that attains the minimum distance (biological scale). Default FALSE.

Details

All numerics besides the linear sum assignment are computed in R. The assignment problem itself is solved by an in-package C++ implementation of the classical $O(n^3)$ Hungarian algorithm, exposed (unexported) as `xsdm:::solve_lsap_cpp`. An R-level alternative is `clue::solve_LSAP`; the two are compared in `tests/testthat/test-solve_lsap_cpp.R`.

Value

If `give_closest_rep` is FALSE, a single number: the distance. Otherwise a list with entries `distance` and `representative`.

References

- H. W. Kuhn (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly* 2(1-2), 83–97.
- J. Munkres (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the SIAM* 5(1), 32–38.
- R. Jonker and A. Volgenant (1987). A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems. *Computing* 38, 325–340.
- K. Hornik (2005). A CLUE for CLUster Ensembles. *Journal of Statistical Software* 14(12). (See also the **clue** package on CRAN for an alternative R-level LSAP implementation.)

Examples

```
# Using lists on the biological scale
par_list <- math_to_bio(example_1$optim_par_vec)
par_list_equivalent <- math_to_bio(example_1$optim_par_vec_equivalent)
dist_between_params(
  p1 = par_list,
  p2 = par_list_equivalent
)

# Using vectors on the math scale
dist_between_params(
  p1 = example_1$optim_par_vec,
  p2 = example_1$optim_par_vec_equivalent
)
```

env_data_array	<i>Get an array of environmental data from presence-absence points.</i>
----------------	---

Description

Get an array of environmental data from presence-absence points.

Usage

```
env_data_array(env_data, occ = NULL)
```

Arguments

env_data	List of environmental variables time series stacks (each a SpatRaster with multiple layers).
occ	Occurrence data frame. Should contain columns "name", "lon", "lat", "presence". If NULL, returns data for all raster cells.

Value

A 3D array of dimensions M (points or cells) $\times$ N (time steps) $\times$ P (environmental variables). The first dimension has no dimnames; the second is named "time" with layer names from the first raster; the third is named "var" with the names of 'env_data'.

Examples

```
bio1_ts <- terra::unwrap(example_1$bio01)
bio12_ts <- terra::unwrap(example_1$bio12)
env_data <- list(bio1 = bio1_ts, bio12 = bio12_ts)
occ <- example_1$occ_df[1:5, ]
# Return array corresponding to each presence absence provided
env_data_array(env_data, occ)
# Return all the environmental in the rasters
env_data_array(env_data, occ)
```

example_1	<i>Consolidated example data for the xsdm package</i>
-----------	---

Description

A named list containing all example datasets used in the package's documentation and examples.

Usage

```
example_1
```

Format

A list of 11 objects:

- `par_vec` Named numeric vector of length 9. Math-scale parameters for a 2-variable model ($p = 2$). Canonical names: `mu1`, `mu2`, `sigltil1`, `sigltil2`, `sigrttil1`, `sigrttil2`, `ctil`, `pd`, `o_par1`.
- `bio01` A packed `SpatRaster` (use `terra::unwrap()`) with 128 x 123 cells and 39 layers. Annual average temperature (`bio1`) for 1980-2018, CHELSA 2.1 data, centred on southern New Mexico, USA.
- `bio12` A packed `SpatRaster` (use `terra::unwrap()`) with 128 x 123 cells and 39 layers. Annual precipitation (`bio12`) for the same region and time period.
- `env_array` A 3-D numeric array with dimensions 4000 (locations) x 39 (time) x 2 (variables). Contains the environmental data (`bio1` and `bio12`, both divided by 100) extracted from the rasters at the `occ_df` locations.
- `occ_df` A data frame with 4000 rows and 4 columns: `name` (character), `lon` (longitude), `lat` (latitude), `presence` (0/1). Occurrence records for a virtual species.
- `occ_vec` An integer vector of length 4000. Binary presence/absence (0/1) corresponding to `occ_df$presence`.
- `true_par_list` A list of biological-scale parameters (the "true" parameter set used to generate the virtual species). Contains `mu`, `sigltil`, `sigrttil`, `ctil`, `pd`, `o_mat`.
- `optim_par_list` A list of biological-scale parameters (the MLE fit for the example). Contains `mu`, `sigltil`, `sigrttil`, `ctil`, `pd`, `o_mat`.
- `optim_par_vec` A named numeric vector of length 9. Math-scale parameters corresponding to `optim_par_list`.
- `optim_par_vec_equivalent` A named numeric vector of length 9. A different math-scale representation that belongs to the same equivalence class as `optim_par_vec`. Used to test `dist_between_params()`.
- `par_table` A data.frame with 9 columns (one per math-scale parameter) and 100 rows of parameter combinations.

Details

All rasters (`bio01`, `bio12`) are stored as packed `SpatRaster` objects to reduce package size. Before using them, unpack with `terra::unwrap()`, e.g.: `bio1 <- terra::unwrap(example_1$bio01)`.

The environmental data are originally from CHELSA v2.1 (<https://www.chelsa-climate.org/>). The virtual species was generated from the parameters in `true_par_list`.

Source

Berti et al., 2025 ([doi:10.1101/2024.10.30.621023](https://doi.org/10.1101/2024.10.30.621023))

Examples

```
# Access the list
names(example_1)

# Unpack a raster

bio1 <- terra::unwrap(example_1$bio01)
```

```
# Use a parameter set
math_to_bio(example_1$par_vec)
```

example_2	<i>Consolidated example data for the xsdm. This is environmental data array and an occurrence presence absence vector of Ophisaurus ventralis. A named list containing all example datasets used in the package’s documentation and examples.</i>
-----------	---

Description

Consolidated example data for the xsdm. This is environmental data array and an occurrence presence absence vector of Ophisaurus ventralis. A named list containing all example datasets used in the package’s documentation and examples.

Usage

```
example_2
```

Format

A list of 2 objects:

env_array A 3-D numeric array with dimensions ‘2728 (locations) × 39 (time) × 2 (variables)’. Contains the environmental data (bio1 and bio12) extracted from the rasters for all locations.

occ_vec An integer vector of length 2728. Binary presence/absence (0/1) for the same locations as ‘env_array’.

Source

Berti et al., 2025 ([doi:10.1101/2024.10.30.621023](https://doi.org/10.1101/2024.10.30.621023))

Examples

```
# Access the list
names(example_2)
```

example_3	<i>Consolidated example data for the xsdm. This is environmental data array and an occurrence presence absence vector Blarina carolinensis A named list containing all example datasets used in the package's documentation and examples.</i>
-----------	---

Description

Consolidated example data for the xsdm. This is environmental data array and an occurrence presence absence vector Blarina carolinensis A named list containing all example datasets used in the package's documentation and examples.

Usage

```
example_3
```

Format

A list of 2 objects:

env_array A 3-D numeric array with dimensions '1156 (locations) × 39 (time) × 2 (variables)'.

Contains the environmental data (bio1 and bio12) extracted from the rasters for all locations.

occ_vec An integer vector of length 1156 Binary presence/absence (0/1) for the same locations as 'env_array'.

Source

Berti et al., 2025 ([doi:10.1101/2024.10.30.621023](https://doi.org/10.1101/2024.10.30.621023))

Examples

```
# Access the list
names(example_3)
```

expit	<i>Functions to take the expit of numerical vectors. expit $\exp(x)/(1 + \exp(x))$</i>
-------	---

Description

Functions to take the expit of numerical vectors. expit $\exp(x)/(1 + \exp(x))$

Usage

```
expit(x)
```

Arguments

x A numeric value

Value

A real vector corresponding to the expits of x

Examples

```
expit(0)
expit(0.5)
expit(-1)
```

```
extract_orthogonal_matrix_parameters
```

Extract a math-scale real-parameter vector corresponding to a special orthogonal matrix

Description

Computes the principal matrix logarithm of a special orthogonal matrix ‘o_mat’, then returns the strictly lower-triangular entries of the resulting skew-symmetric matrix. This is a partial inverse of ‘build_orthogonal_matrix’, up to the periodicity of the exponential map.

Usage

```
extract_orthogonal_matrix_parameters(o_mat)
```

Arguments

o_mat A $k \times k$ special orthogonal matrix ($t(o_mat) \% \% o_mat = I$ and $\det(o_mat) = 1$).

Details

The matrix exponential $\exp : \mathfrak{so}(k) \rightarrow SO(k)$ is surjective but not injective: different skew-symmetric matrices can exponentiate to the same orthogonal matrix. This function uses the ****principal matrix logarithm**** as implemented in ‘expm::logm’. Consequently, it may fail (or produce complex results) for matrices that have eigenvalues equal to -1 (i.e., rotations by π). Such matrices lie on the cut locus of the exponential map and do not possess a unique real logarithm. If you encounter this, consider perturbing the matrix slightly away from the problematic rotation.

Value

A numeric vector of length $k(k-1)/2$ containing the strictly lower-triangular entries of the skew-symmetric generator. For $k = 1$, returns NULL (the identity matrix).

Examples

```
o_par2 <- 0.25
O2 <- build_orthogonal_matrix(o_par2)
extract_orthogonal_matrix_parameters(O2)
```

habitat_suitability	<i>Tiled habitat-suitability map from environmental raster stacks</i>
---------------------	---

Description

Evaluates the log detection probability (or its exponential, the probability of detection) for every cell of a list of multi-layer terra::SpatRaster objects, processing the inputs in memory-bounded blocks so that arbitrarily large grids can be handled without loading the entire dataset into R memory. Each block is forwarded to [log_prob_detect_cpp](#), the xtensor-backed C++ kernel that consolidates the like_neg_ltsgr() -> like_ltsgr() call chain.

Usage

```
habitat_suitability(
  param_list,
  env_list,
  output = "",
  overwrite = FALSE,
  return_prob = TRUE,
  threads = 0L,
  wopt = list()
)
```

Arguments

param_list	A named list of biological-scale parameters. Must contain mu, sigltil, sigrttil, o_mat, ctil and pd. See log_prob_detect for details of each element.
env_list	A list of SpatRaster objects, one per environmental variable. Each raster must have the same number of layers (time steps) and identical spatial geometry (extent, resolution, CRS). Minimum length 1.
output	Character scalar. File path for the output GeoTIFF. The empty string "" (default) creates an in-memory SpatRaster.
overwrite	Logical scalar. If TRUE, an existing file at output is overwritten. Default FALSE.
return_prob	Logical scalar. If TRUE (default), the output cell values are probabilities of detection (range [0,1]). If FALSE, the cell values are log-probabilities (range $(-\infty, 0]$).
threads	Integer scalar. Number of parallel threads forwarded to log_prob_detect_cpp . Use 0 (default) to let RcppParallel pick the number of threads automatically.
wopt	List. Additional write options forwarded to writeStart . Default list().

Details

Internally the function uses **terra**'s streaming block-loop API:

1. `readStart` is called on every raster in `env_list`.
2. `writeStart` is called on the output raster, which returns a block schedule chosen by terra's memory manager.
3. For each block, `readValues` reads a horizontal strip from every input raster into a matrix; the strips are packed into a flat column-major vector and passed to `log_prob_detect_cpp`. Cells that are NA in any variable or time step are masked out and re-inserted as NA in the output.
4. `writeValues` writes the per-cell results.
5. `readStop` and `writeStop` are called via `on.exit` to ensure file handles are released even if an error occurs.

At most one block of pixels is held in R memory at any time, making the function suitable for continental or global rasters.

Value

A `SpatRaster` with one layer named either "habitat_suitability" (when `return_prob = TRUE`) or "log_prob_detect" (when `return_prob = FALSE`). The raster is returned invisibly when output `!= ""`.

See Also

`log_prob_detect_cpp`, `log_prob_detect`, `vsp`, `writeStart`

Examples

```
data("example_1", package = "xsgm")
bio01 <- terra::unwrap(example_1$bio01) / 100
bio12 <- terra::unwrap(example_1$bio12) / 100
env_list <- list(bio01 = bio01, bio12 = bio12)
suit <- habitat_suitability(
  param_list = example_1$true_par_list,
  env_list   = env_list,
  return_prob = TRUE
)
suit
```

Description

Due to the parameter reduction step which was carried out to eliminate structural non-identifiability in the xsdm model, parameter interpretation is more difficult. This function helps with that difficulty, displaying contours for the inferred log growth-environment function. The shapes of these contours are determined by inference, even though their levels are not; and the shapes are generally more informative anyway. See the manual documents “The xsdm model” and “How to fit xsdm models with species occurrence data using xsdm” for additional details.

Usage

```
interpret_parameters(
  param_list,
  plot_indices,
  plot_lims = NULL,
  env_dat = NULL,
  occ = NULL,
  breadth = 1,
  ...
)
```

Arguments

param_list	A named list of xsdm model parameters such as returned by <code>math_to_bio</code> . Must contain elements <code>mu</code> , <code>sigltil</code> , <code>sigrttil</code> , <code>ctil</code> , <code>pd</code> , and <code>o_mat</code> .
plot_indices	A length-1 or length-2 integer vector of indices of environmental variables against which the growth-environment function is to be plotted. For a length-2 vector, the first index is the horizontal axis, the second the vertical. Other environmental variables are held at their values in <code>param_list\$mu</code> .
plot_lims	Optional list of the same length as <code>plot_indices</code> , each element a 2-vector giving the plotting extent *relative to* <code>mu</code> . If <code>NULL</code> (the default) and <code>env_dat</code> is supplied, limits are auto-derived via <code>auto_plot_lims_()</code> using the <code>breadth</code> argument. The auto-derived limits cover the full observed environmental range plus a symmetric margin on each side.
env_dat	Optional 3D numeric array of environmental data with dimensions (locations) $\times$ (time) $\times$ (variables). Required for the two-panel (presence vs non-detection) display and for auto-derived <code>plot_lims</code> . If <code>NULL</code> , a single-panel legacy plot is drawn and <code>plot_lims</code> must be supplied.
occ	Optional length-(locations) logical or 0/1 vector of presence/absence. Required together with <code>env_dat</code> for the two-panel display.
breadth	Scalar in $[0, 1]$ controlling how wide the auto-derived plotting window is around <code>mu</code> : <code>breadth = 1</code> (default) shows the full min-max environmental range plus a 10% margin on each side; <code>breadth = 0</code> collapses to essentially a single point. Ignored when <code>plot_lims</code> is supplied.
...	Additional graphical arguments passed to <code>plot</code> (1D case) or <code>image</code> (2D case).

Details

If `env_dat` and `occ` are provided, two panels are drawn side by side: on the left the growth-environment function is shown together with the environmental values at presence locations (`occ == 1`); on the right the same function is shown together with the environmental values at non-detections (`occ == 0`). Both panels share identical contour breaks (bivariate case) or identical axes (univariate case), so the two are directly comparable.

The log growth-environment function is determined by inference only up to an affine transformation $g = af(e) + b$ with $a > 0$. Its contours are therefore unlabelled in the output; their shape is what is interpretively meaningful. In code the function is

$$y(e) = - \sum_i \left(\frac{[u_i]_+}{\sigma_i^R} + \frac{[u_i]_-}{\sigma_i^L} \right)^2, \quad u = O^T(e - \mu),$$

which is always ≤ 0 , attains its maximum 0 at $e = \mu$, and decreases without bound as e moves away from μ . Consequently the numeric values on the y-axis of the univariate plot and the numeric values of the image colors in the bivariate plot carry no units of their own.

Value

Invisibly returns the (possibly auto-derived) `plot_lims` list, so downstream code can reuse the same limits. The main purpose of the function is its side effect: plots are sent to the default graphics device.

Examples

```
# Two-panel (presence vs non-detection) plot with auto-derived limits
interpret_parameters(
  example_1$true_par_list,
  plot_indices = c(1, 2),
  env_dat      = example_1$env_array,
  occ          = example_1$occ_vec
)

# Narrower auto-derived window
interpret_parameters(
  example_1$true_par_list,
  plot_indices = c(1, 2),
  env_dat      = example_1$env_array,
  occ          = example_1$occ_vec,
  breadth      = 0.7
)
```

like_ltsg

*Compute likelihood for LTSG model***Description**

Compute likelihood for LTSG model

Compute likelihood for LTSG model

Arguments

mu	Numeric vector of means (length equal to number of rows in ‘env_m’)
env_m	Numeric matrix of environmental data. Must be column-major with time varying fastest: column j corresponds to (location, time) via index $j*q + i$. This matches the memory layout expected by the underlying C++ implementation.
dl_mat	Diagonal matrix (as NumericMatrix)
drl_mat	Diagonal matrix (as NumericMatrix)
ortho_m	Numeric matrix (orthogonal basis)
q	Integer, number of rows for reshaping
r	Integer, number of columns for reshaping.

Details

This function calculates a likelihood-like measure using orthogonal matrices, environmental data, and diagonal matrices, leveraging parallel computation.

This function calculates a likelihood-like measure using orthogonal matrices, environmental data, and diagonal matrices, leveraging parallel computation.

Value

A numeric vector of length ‘r’ with computed sums.

A numeric vector of length ‘r’ with computed sums.

Examples

```
mu <- c(1, 2)
ortho_m <- matrix(1:4, nrow = 2)
env_m <- matrix(1:4, nrow = 2)
dl_mat <- diag(2)
drl_mat <- diag(2)
like_ltsg(mu, env_m, dl_mat, drl_mat, ortho_m, q = 1, r = 2)
```

like_neg_ltsg	<i>Long-term stochastic growth rate worker function for the xsdm model</i>
---------------	--

Description

Computes the negative of the long-term stochastic growth rate, plus $\log(\lambda_{\max})$, for the xsdm model, for each location. This is the fast version of a worker function that relies on C++ code, optimized using **RcppParallel**. The legacy pure-R reference is preserved as `xsdm:::like_neg_ltsg_r` for testing and comparison.

Usage

```
like_neg_ltsgr(
  env_dat,
  mu,
  sigltil,
  sigrttil,
  o_mat,
  num_threads = RcppParallel::defaultNumThreads()
)
```

Arguments

env_dat	The environmental data array, dimensions (number of locations) x (time series length) x (number of environmental variables). Must not contain missing values.
mu	Vector of optimal environmental values. Length $p = \dim(\text{env_dat})[3]$. Unconstrained real numbers.
sigltil	Vector specifying width of the growth-environment function. Length $p = \dim(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.
sigrttil	Vector specifying width of the growth-environment function. Length $p = \dim(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.
o_mat	An orthogonal matrix, dimensions p by p .
num_threads	Number of threads for parallel computation. Defaults to <code>RcppParallel::defaultNumThreads()</code> .

Details

Internally, this function:

1. Reshapes the environmental data into a matrix.
2. Computes inverse (diagonal) matrices for asymmetry adjustments.
3. Calls the C++ function `like_ltsg()` for efficient likelihood computation.

Value

A vector of length equal to the number of locations, as described above.

Note

Ensure that `env_dat` has no missing values. The parameter vectors `mu`, `sigltil`, and `sigrttil` must have length equal to the number of environmental variables (`p`). The argument `o_mat` must be a $p \times p$ orthogonal matrix (i.e., `o_mat %*% t(o_mat)` is the identity).

Examples

```
# Example usage:
like_neg_ltsgr(env_dat = example_1$env_array,
               mu       = example_1$true_par_list$mu,
               sigltil  = example_1$true_par_list$sigltil,
               sigrttil = example_1$true_par_list$sigrttil,
               o_mat    = example_1$true_par_list$o_mat)
```

log_prob_detect	<i>Probability of detection of the species in each location</i>
-----------------	---

Description

Computes the probability of detection of the species in each location for the xsdm model, given environmental data and model parameters.

Usage

```
log_prob_detect(
  env_dat,
  mu,
  sigltil,
  sigrttil,
  o_mat,
  ctil,
  pd,
  return_prob = FALSE,
  num_threads = RcppParallel::defaultNumThreads()
)
```

Arguments

<code>env_dat</code>	The environmental data array, dimensions $n_{\text{loc}} \times n_{\text{time}} \times p$ (number of locations $\times$ time-series length $\times$ number of environmental variables). Must be a 3-dimensional array with no missing values.
<code>mu</code>	Vector of optimal environmental values. Length $p = \text{dim}(\text{env_dat})[3]$. Unconstrained real numbers.
<code>sigltil</code>	Vector specifying width of the growth-environment function. Length $p = \text{dim}(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.
<code>sigrttil</code>	Vector specifying width of the growth-environment function. Length $p = \text{dim}(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.

<code>o_mat</code>	An orthogonal matrix, dimensions p by p .
<code>ctil</code>	Scalar. Relates to the center of the detection-link function.
<code>pd</code>	Maximum probability of detection of the species. Parameter between 0 and 1.
<code>return_prob</code>	Logical (default FALSE). Flag to return probabilities of detection instead their logs.
<code>num_threads</code>	Number of threads for parallel computation. Defaults to <code>RcppParallel::defaultNumThreads()</code> .

Details

This is a thin R wrapper around the C++ implementation `log_prob_detect_cpp`; the optimizer hot path is pure C++. A pure-R reference implementation, `log_prob_detect_r`, is kept internal to the package and is used only by the parity tests in `tests/testthat/test-log_prob_detect_r_vs_cpp.R`.

Value

A vector of length equal to the number of locations, containing the probabilities of detection (or their logs) of the species in each location.

Examples

```
mu <- c(-1, 5.046939)
sigltil <- c(1.036834, 1.556083)
sigrtil <- c(1.538972, 1.458738)
ctil <- -2
pd <- 0.9
o_mat <- matrix(c(-0.4443546, 0.8958510, -0.8958510, -0.4443546), ncol = 2)
```

<code>log_prob_detect_cpp</code>	<i>Compute log detection probabilities from a flat environmental data vector</i>
----------------------------------	--

Description

C++ implementation of `log_prob_detect()` that accepts environmental data as a flat numeric vector with explicit dimension metadata. This signature is designed for block-by-block raster evaluation where each block is passed as a contiguous vector rather than a 3-D R array.

Usage

```
log_prob_detect_cpp(
  env_dat_vec,
  env_dat_dims,
  mu,
  sigltil,
  sigrtil,
  o_mat,
  ctil,
```

```
pd,
  return_prob = FALSE,
  num_threads = 0L
)
```

Arguments

env_dat_vec	Numeric vector. Column-major flat representation of a 3-D array with logical dimensions $c(n_loc, ts_length, p)$: variable k (1-indexed) occupies positions $(k-1)*n_loc*ts_length + 1$ to $k*n_loc*ts_length$, and within that block pixels (locations) vary fastest.
env_dat_dims	Integer vector of length 3: $c(n_loc, ts_length, p)$.
mu	Numeric vector of length p . Optimal environmental values.
sigltil	Numeric vector of length p . Positive; Inf entries are allowed (treated as zero inverse-scale).
sigrttil	Numeric vector of length p . Positive; Inf entries are allowed.
o_mat	Numeric matrix, $p \times p$ orthogonal.
ctil	Scalar. Center of the detection-link function.
pd	Scalar in $(0, 1]$. Maximum probability of detection.
return_prob	Logical. If TRUE, return probabilities; if FALSE (default) return log-probabilities.
num_threads	Integer. Number of parallel threads. 0 (default) uses <code>RcppParallel::defaultNumThreads()</code> .

Details

Collapses the R call chain `like_neg_1tsgr_cpp()` -> `like_1tsg()` into a single xtensor-accelerated C++ function.

Value

Numeric vector of length n_loc .

log1mexp	<i>Numerically stable $\log(1 - \exp(-a))$</i>
----------	---

Description

Computes $\log(1 - \exp(-a))$ accurately for non-negative ‘ a ’, using two different formulas depending on whether ‘ a ’ is above or below ‘ $\log(2)$ ’.

Usage

```
log1mexp(a, cutoff = log(2))
```

Arguments

a	Numeric vector of non-negative values. ‘NA’ values are preserved; negative values emit a warning and return ‘NaN’.
cutoff	Positive numeric scalar. Threshold between the two formulas; ‘log(2)’ is near-optimal.

Value

A numeric vector the same length as ‘a’ with $\log(1 - \exp(-a))$.

References

Mächler, M. (2012). *Accurately Computing $\log(1 - \exp(-|a|))$.* CRAN package ‘copula’ vignette.

See Also

[log1pexp](#), [log1p](#), [expm1](#)

Examples

```
a <- 2^seq(-20, 5, length.out = 10)
cbind(a, log(1 - exp(-a)), log1mexp(a))
```

log1pexp	<i>Numerically stable ‘$\log(1 + \exp(x))$’</i>
----------	--

Description

Computes $\log(1 + \exp(x))$ accurately for any real ‘x’, avoiding overflow as ‘x -> +Inf’ and catastrophic cancellation as ‘x -> -Inf’.

Usage

```
log1pexp(x, c0 = -37, c1 = 18, c2 = 33.3)
```

Arguments

x	Numeric vector. ‘NA’ values are preserved.
c0, c1, c2	Numeric scalars defining the switch points between four asymptotically optimal formulas. Defaults (-37, 18, 33.3) are from Mächler (2012) and should not normally be changed.

Value

A numeric vector the same length as ‘x’ with $\log(1 + \exp(x))$.

References

Mächler, M. (2012). *Accurately Computing $\log(1 - \exp(-\text{la}))$.* CRAN package ‘copula’ vignette.

See Also

[log1mexp](#), [log1p](#), [expm1](#)

Examples

```
x <- seq(-40, 40, by = 10)
cbind(x, log1p(exp(x)), log1pexp(x))
```

loglik_bio	<i>Log-likelihood function for the xsdm model, parameters on the biological scale.</i>
------------	--

Description

Computes the log-likelihood for the xsdm model given environmental data, a vector of occurrences and pseudo-absences, and model parameters on the biological scale.

Usage

```
loglik_bio(
  env_dat,
  occ,
  mu,
  sigltil,
  sigrttil,
  o_mat,
  ctil,
  pd,
  return_prob = FALSE,
  sum_log_p = TRUE,
  num_threads = RcppParallel::defaultNumThreads()
)
```

Arguments

env_dat	The environmental data array, dimensions $n_{\text{loc}} \times n_{\text{time}} \times p$ (number of locations $\times$ time-series length $\times$ number of environmental variables). Must be a 3-dimensional array with no missing values.
occ	Presence/pseudo-absence binary vector. Same length as dimension 1 of env_dat.
mu	Vector of optimal environmental values. Length $p = \text{dim}(\text{env_dat})[3]$. Unconstrained real numbers.
sigltil	Vector specifying width of the growth-environment function. Length $p = \text{dim}(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.

sigrttil	Vector specifying width of the growth-environment function. Length $p = \dim(\text{env_dat})[3]$. Positive real numbers, Inf entries also allowed.
o_mat	An orthogonal matrix, dimensions p by p .
ctil	Scalar. Relates to the center of the detection-link function.
pd	Maximum probability of detection of the species. Parameter between 0 and 1.
return_prob	Logical (default FALSE). Flag to return likelihood instead of log-likelihood.
sum_log_p	Logical (default TRUE). If FALSE, returns the individual log-likelihoods (or likelihoods, if return_prob is TRUE) associated with the individual locations, instead of their sum (product, if return_prob is TRUE).
num_threads	Number of threads for parallel computation. Defaults to <code>RcppParallel::defaultNumThreads()</code> .

Details

This is a thin R wrapper around the C++ implementation `loglik_bio_cpp`; the optimizer hot path (`sum_log_p = TRUE`, `return_prob = FALSE`) is pure C++. The non-default flag combinations (`sum_log_p = FALSE` or `return_prob = TRUE`) are computed by delegating to the C++-backed `log_prob_detect` and reducing in R. A pure-R reference implementation, `loglik_bio_r`, is kept internal to the package and is used only by the parity tests in `tests/testthat/test-loglik_bio_r_vs_cpp.R`.

Value

A single value, the log-likelihood (or the likelihood, if `return_prob` is TRUE); or a vector of location specific values of `sum_log_p` is FALSE.

Examples

```
ll <- loglik_bio(
  env_dat = example_1$env_array,
  occ = example_1$occ_vec,
  mu = example_1$true_par_list$mu,
  sigltil = example_1$true_par_list$sigltil,
  sigrttil = example_1$true_par_list$sigrttil,
  o_mat = example_1$true_par_list$o_mat,
  ctil = example_1$true_par_list$ctil,
  pd = example_1$true_par_list$pd
)
ll
```

loglik_bio_cpp	<i>Pure-C++ log-likelihood for the xsdm model (biological-scale parameters)</i>
----------------	---

Description

Computes the log-likelihood directly in C++ without any R callback. Equivalent to `loglik_bio(..., sum_log_p = TRUE, return_prob = FALSE)`.

Usage

```
loglik_bio_cpp(
  env_dat_vec,
  env_dat_dims,
  occ,
  mu,
  sigltil,
  sigrtil,
  o_mat,
  ctil,
  pd,
  num_threads = 0L
)
```

Arguments

env_dat_vec	Flat numeric vector containing env_dat in column-major order (as produced by <code>as.vector(env_dat)</code>).
env_dat_dims	Integer vector of length 3: <code>c(n_loc, ts_length, p)</code> .
occ	Integer vector of length <code>n_loc</code> , 0 or 1.
mu	Numeric vector, length <code>p</code> .
sigltil	Positive numeric vector, length <code>p</code> .
sigrtil	Positive numeric vector, length <code>p</code> .
o_mat	A <code>p x p</code> orthogonal matrix (column-major).
ctil	Scalar.
pd	Scalar in $(0, 1]$.
num_threads	Number of threads for the inner xtensor kernel (0 = RcppParallel default).

Value

Scalar log-likelihood.

loglik_math	<i>Log-likelihood function for the xsdm model, parameters on the math scale.</i>
-------------	--

Description

Computes the log-likelihood for the xsdm model given environmental data, a vector of occurrences and pseudo-absences, and model parameters on the math scale. This is the function that one optimizes to fit xsdm with data.

Usage

```
loglik_math(
  param_vector,
  env_dat,
  occ,
  mask = NULL,
  num_threads = RcppParallel::defaultNumThreads(),
  negative = TRUE
)
```

Arguments

param_vector	A named numeric vector of math-scale parameters. When mask = NULL, the names must exactly match the canonical schema returned by <code>make_mask_names(p)</code> where <code>p = dim(env_dat)[3]</code> , and the length must equal <code>num_par(p)</code> . When mask is supplied, param_vector should contain only the names not present in mask, in the canonical order. In both cases the vector is combined with mask via <code>create_param_vector_masked</code> and then mapped to biological-scale parameters via <code>math_to_bio</code> . Must not contain missing values. See Details for the full naming and ordering conventions.
env_dat	The environmental data array, dimensions <code>n_loc x n_time x p</code> (number of locations x time-series length x number of environmental variables). Must be a 3-dimensional array with no missing values.
occ	Presence/pseudo-absence binary vector. Same length as dimension 1 of env_dat.
mask	For optionally keeping some parameters at fixed values during optimization. Either NULL or a named numeric vector. The NULL case means all parameters are in param_vector, corresponding to the case where all parameters will be adjustable by the optimizer when this function is passed to it as the objective function. In the non-NULL case, names of entries of mask must correspond precisely to parameter names (see Details), and then those values are used. In that case, param_vector is construed to contain the values of the other parameters, in order (see Details). So, in particular, the length of param_vector plus the length of mask must equal the total number of parameters for the model, which is determined by <code>dim(env_dat)[3]</code> . The most common case for most applications will be mask=NULL. Entries of mask are interpreted on the math scale.
num_threads	Number of threads for parallel computation. Defaults to <code>RcppParallel::defaultNumThreads()</code> .
negative	Logical. If TRUE returns the negative of the log-likelihood instead of the log-likelihood itself. Facilitates optimization with some optimizers.

Details

Optimizing the likelihood and profiling requires conventions for transforming parameters from unconstrained spaces to the constrained space of possible parameters which can be accepted by `loglik_bio`. This function and `math_to_bio` implement those conventions, and also allow for optimizations while keeping one or more parameters fixed, including potentially at boundary values. Typically `loglik_math` is the function one optimizes numerically in order to fit xsdm or a boundary model with data, or to profile a fitted model. For what follows, denote `dim(env_dat)[3]` by `p`.

We start by explaining the case `mask=NULL`, for which all model parameters are in `param_vector`. The parameters of `param_vector` are assumed to appear in the following order:

1. Parameters for μ , of which there are p ;
2. Parameters which are exp-transformed to get the entries of `sigltil`, of which there are p ;
3. Parameters which are exp-transformed to get the entries of `sigrttil`, of which there are p ;
4. The parameter `ctil`;
5. A parameter which is expit transformed to get `pd`;
6. Parameters which are inserted via column-major order into the lower- triangle of a skew-symmetric matrix which is then transformed by the matrix exponential to get `o_mat`, of which there are $(p^2-p)/2$.

Thus, when `mask` is `NULL`, `param_vector` must be an unconstrained numeric vector of length $3*p+2+(p^2-p)/2$ with no missing values.

The argument `mask` is used in the event one wants to fix certain parameters and optimize over the remaining parameters. This argument must be a named numeric vector with unique names being some but not all of the $3*p+2+(p^2-p)/2$ following: `mu1`, `mu2`, ..., `mup`, `sigltil1`, `sigltil2`, ..., `sigltilp`, `sigrttil1`, `sigrttil2`, ..., `sigrttilp`, `ctil`, `pd`, and `o_mati` for i ranging from 1 to $(p^2-p)/2$. These names must be used exactly. See the function `make_mask_names`, which facilitates the construction of a correctly formatted `mask` argument. Entries of `mask` are on the math scale; `bio_to_math` can convert biological-scale constraints to math scale.

The missing entries of `mask` are filled in using the entries of `param_vector`, in the order specified above, and then the transformations described above (implemented by `math_to_bio`) are applied to get biological-scale parameters which are passed to `loglik_bio` to get the log likelihood.

Entries of `mask` corresponding to `sigltil` or `sigrttil` can be `Inf`. Likewise, the entry of `mask` corresponding to `pd` can be `Inf` (on the math scale, corresponding to a biological-scale value of 1). This functionality is used to fit boundary models. Entries of `param_vector` must be finite.

Value

A single value, the log-likelihood (or the negative log-likelihood, if `negative` is `TRUE`).

Implementation

This is a thin R wrapper around the C++ implementation `loglik_math_cpp`; the optimizer hot path is pure C++. A pure-R reference, `loglik_math_r`, is kept internal to the package and is used only by the parity tests in `tests/testthat/test-loglik_math_r_vs_cpp.R`.

Examples

```
# Testing the function with the example data
loglik_math(
  param_vector = example_1$par_vec,
  env_dat = example_1$env_array,
  occ = example_1$occ_vec
)
# Mute one parameter to use the mask
par_vec <- example_1$par_vec[-2]
```

```

mask_parameters_a <- c(mu2 = 6.5)
loglik_math(
  param_vector = par_vec,
  env_dat = example_1$env_array,
  occ = example_1$occ_vec,
  mask = mask_parameters_a
)
# Return the negative
loglik_math(
  param_vector = example_1$par_vec,
  env_dat = example_1$env_array,
  occ = example_1$occ_vec,
  negative = TRUE
)

```

loglik_math_cpp	<i>Pure-C++ log-likelihood for the xsdm model (math-scale parameters)</i>
-----------------	---

Description

Computes the log-likelihood directly in C++ without any R callback in the inner loop. Semantically equivalent to the R function `loglik_math`.

Usage

```

loglik_math_cpp(
  param_vector,
  env_dat,
  occ,
  mask = NULL,
  negative = TRUE,
  num_threads = 0L
)

```

Arguments

<code>param_vector</code>	Named numeric vector of math-scale parameters. When ‘mask’ is NULL, must contain every canonical name for the dimension <code>p</code> implied by ‘env_dat’. When ‘mask’ is supplied, contains only the free (non-masked) parameters.
<code>env_dat</code>	3D numeric array with dimensions <code>(n_loc, ts_length, p)</code> . No missing values allowed.
<code>occ</code>	Integer or logical vector of length <code>n_loc</code> , 0/1 or FALSE/TRUE.
<code>mask</code>	Optional named numeric vector of fixed parameters.
<code>negative</code>	Logical; if TRUE (default) returns the negative log-likelihood (the value to be minimized).
<code>num_threads</code>	Integer; 0 leaves the <code>RcppParallel</code> default.

Value

A scalar double.

make_mask_names	<i>Function to facilitate the creation of the argument mask to the function loglik_math</i>
-----------------	---

Description

The argument mask to the function loglik_math is required to follow some very specific conventions in order to reduce the risk of errors coming from mismatched arguments. This function facilitates the creation of such vectors.

Usage

```
make_mask_names(p)
```

Arguments

p A positive integer representing the number of environmental variables to be used in the xsdm model, i.e., `dim(env_dat)[3]` for the `env_dat` argument of the `loglik_math` function

Details

The output has length $3 \cdot p + (p^2 - p)/2 + 2$. The names of the entries are `mu1`, `mu2`, ..., `mup`, `sigltil1`, `sigltil2`, ..., `sigltilp`, `sigrttil1`, `sigrttil2`, ..., `sigrttilp`, `o_pari` for `i` ranging from 1 to $(p^2 - p)/2$, `ctil`, and `pd`. All entries are NA.

Value

A named numeric vector full of NAs, with the names generated according to the conventions in Details of the function `loglik_math`. See also Details below.

Examples

```
make_mask_names(2)
```

math_to_bio

Convert parameters from the math scale to the biological scale

Description

Transforms an unconstrained "math-scale" parameter vector into the corresponding biologically interpretable parameters. The input must be a **named numeric vector** whose names exactly match the canonical schema returned by [make_mask_names\(p\)](#) for some integer $p \geq 1$.

The canonical names are (in order):

- mu1, mu2, ..., mup
- sigltil1, sigltil2, ..., sigltilp
- sigrtil1, sigrtil2, ..., sigrtilp
- ctil
- pd
- o_par1, o_par2, ..., o_parq where $q = p(p - 1)/2$

These are the names generated by [make_mask_names\(p\)](#).

Usage

```
math_to_bio(param_vector)
```

Arguments

param_vector A **named** numeric vector of math-scale parameters. The names must be exactly those returned by [make_mask_names\(p\)](#), and the length must equal [num_par\(p\)](#). No missing values are allowed. Passing an unnamed vector or a vector whose names do not match the canonical order will raise an error.

Details

The transformations applied are:

- mu: identity (unchanged)
- sigltil, sigrtil: [exp\(\)](#)
- ctil: identity (unchanged)
- pd: [expit\(\)](#)
- o_par: used to build an orthogonal matrix via [build_orthogonal_matrix](#)

For $p = 1$ there are no o_par entries; the orthogonal matrix is simply a 1-by-1 identity.

This is a thin R wrapper around the internal C++ implementation `xsdm:::math_to_bio_cpp`. The pre-port pure-R implementation is preserved internally as `xsdm:::math_to_bio_r` for parity testing.

Value

A named list of biological-scale parameters with elements: mu, sigltil, sigrttil, ctil, pd, o_mat.

See Also

[make_mask_names](#), [num_par](#), [num_env_var](#), [loglik_math](#), [bio_to_math](#)

Examples

```
# Create your own vector of parameter for p = 1 (no o_par entries),
# We use the function make_mask_names with p = 1 to get the correct names and
# length
p1_names <- make_mask_names(1)
math_vec <- p1_names
math_vec[] <- c(11, log(1.2), log(0.8), -6.7, -1.13)
# We get a list with parameters in biological scale
math_to_bio(math_vec)

# For p = 2 (includes o_par1) -- using the shipped example vector
math_to_bio(example_1$par_vec)
```

num_env_var	<i>Get the number of environmental variables given the number of parameters</i>
-------------	---

Description

Inverts num_par(p) to recover p from n, the number of parameters of the main xsdm model. Uses the closed-form solution of the quadratic: $2n = p^2 + 5p + 4$, i.e. $p = (-5 + \sqrt{9 + 8n})/2$. Errors if n is not a valid value of num_par(p) for some integer $p \geq 1$.

Usage

```
num_env_var(n)
```

Arguments

n Integerish scalar: total number of parameters.

Value

A single integer p, the number of environmental variables.

Examples

```

num_env_var(5) # -> 1 (since num_par(1) = 5)
num_env_var(9) # -> 2 (since num_par(2) = 9)
num_env_var(14) # -> 3 (since num_par(3) = 14)
# round-trip check:
p <- 4
stopifnot(num_env_var(num_par(p)) == p)

```

num_par

Get the number of parameters of the main xsdm model given the number of environmental variables to be considered

Description

Get the number of parameters of the main xsdm model given the number of environmental variables to be considered

Usage

```
num_par(p)
```

Arguments

p A positive integer representing the number of environmental variables to be used in the xsdm model, i.e., `dim(env_dat)[3]` for the `env_dat` argument of the `loglik_math` function

Details

For instance, in the ‘p=1’ case, the xsdm model parameters are ‘mu’, ‘sigltil’, and ‘sigrttil’ (which are scalars in the ‘p=1’ case); ‘ctil’, and ‘pd’ (which are scalars for any value ‘p’). That makes 5 parameters, so this function returns 5. In the ‘p=2’ case, the parameters are ‘mu’, ‘sigltil’, and ‘sigrttil’ (each of which is now a length-2 vector); ‘ctil’, and ‘pd’ (again scalars); and the single parameter pertaining to ‘o_mat’; for a total of 9.

Value

An integer with the number of parameters

Examples

```
num_par(2)
```

optimize_likelihood	<i>Optimize the xsdm log-likelihood from multiple starts (ucminfcpp)</i>
---------------------	--

Description

Runs multiple ucminfcpp optimizations from starting values generated by `start_parms()`, optionally in parallel. Returns one row per start with the achieved log-likelihood, convergence code, and a full math-scale parameter vector (list-column) reconstructed with mask.

Usage

```
optimize_likelihood(
  env_dat,
  occ,
  mask = NULL,
  num_starts = 100L,
  breadth = 1,
  parallel = FALSE,
  num_cores = max(1L, future::availableCores() - 1L),
  num_threads = RcppParallel::defaultNumThreads(),
  control = list(),
  verbose = FALSE
)
```

Arguments

<code>env_dat</code>	3D array: (locations x time x variables), no NAs.
<code>occ</code>	Logical or integerish 0/1 vector, length = <code>nrow(env_dat)</code> .
<code>mask</code>	NULL or named numeric math-scale values to fix.
<code>num_starts</code>	Integer. Number of starting points. Default 100.
<code>breadth</code>	Scalar in $[0, 1]$ controlling how wide the search ranges are around their (fixed, data-driven) center. <code>breadth = 1</code> (the default) reproduces the pre-v0.3 behaviour that corresponded to <code>quant_vec = c(0.1, 0.5, 0.9)</code> ; <code>breadth = 0</code> collapses every range to essentially a single point, equivalent to <code>quant_vec = c(0.5 - 1e-6, 0.5, 0.5 + 1e-6)</code> . Values in between interpolate linearly.
<code>parallel</code>	Logical. If TRUE, distribute starts via future/furrr .
<code>num_cores</code>	Integer ≥ 1 . Cores used if <code>parallel=TRUE</code> . Default is the number of cores available - 1.
<code>num_threads</code>	Integer ≥ 1 . Threads used inside <code>loglik_math</code> . If <code>parallel=TRUE</code> , consider <code>num_threads=1</code> .
<code>control</code>	Named list merged into ucminfcpp control. User wins over defaults.
<code>verbose</code>	Logical. If TRUE, prints compact progress messages.

Value

A list with:

- solutions: data.frame sorted by decreasing loglik, columns: start_id, loglik, convergence, and list-column full_par (complete math-scale vector).
- best: list with par (full math-scale vector), loglik, and convergence.

Examples

```
optimize_likelihood(
  env_dat = example_1$env_array[1:4, , ],
  occ = example_1$occ_vec[1:4],
  num_starts = 4L
)
```

profile_likelihood	<i>Basic (non-adaptive) tool for profiling the likelihood</i>
--------------------	---

Description

Basic (non-adaptive) tool for profiling the likelihood

Usage

```
profile_likelihood(
  profile_parameter = "mu1",
  increment_left = 0.1,
  increment_right = increment_left,
  num_steps_left = 20L,
  num_steps_right = num_steps_left,
  alpha = 0.95,
  optim_param_vector,
  env_dat,
  occ,
  mask = NULL,
  num_threads = RcppParallel::defaultNumThreads(),
  control = list(),
  verbose = FALSE
)
```

Arguments

profile_parameter Character. Name of the parameter to profile. Profiles are done on the math scale.

increment_left Numeric. Step size (math scale) when moving to the left, from the start point of the parameter point estimate, to construct the profile.

increment_right	Numeric. Step size (math scale) when moving to the right, from the start point of the parameter point estimate, to construct the profile.
num_steps_left	Integer. Maximum number of steps to take to the left.
num_steps_right	Integer. Maximum number of steps to take to the right.
alpha	Numeric value between 0 and 1. Confidence level used for the likelihood ratio (LR) threshold: $\text{threshold} = \text{MLE_loglik} - \text{qchisq}(\alpha, 1)/2$.
optim_param_vector	Named numeric. MLE parameters on math scale.
env_dat	3D array (locations x time x variables).
occ	Logical, either 0 or 1, vector (length = number of locations).
mask	Named numeric or NULL. Parameters kept fixed (math scale).
num_threads	Integer. Threads used internally by log-likelihood.
control	Named list. Control passed to <code>ucminfcpp::ucminf_xptr(control = ...)</code> . User-specified entries override defaults: • <code>grad = "central"</code> • <code>gradstep = c(1e-6, 1e-8)</code> • <code>grtol = 1e-5</code> • <code>xtol = 1e-12</code> • <code>stepmax = 5</code> • <code>maxeval = 2000</code> If you want optimizer iteration trace, set <code>control\$trace > 0</code> .
verbose	Logical. If TRUE, prints compact progress messages; otherwise silent.

Value

A list with:

- `profile`: data.frame with columns `param`, `value_math`, `loglik`, `convergence`, and a list-column `full_par`. (No side/step columns.)
- `found_better`: logical; TRUE if any profiled point exceeds the MLE log-likelihood.
- `threshold`: numeric; LR threshold used.
- `parameters`: data.frame with the parameters found in each step of the profiling

Examples

```
## Minimal profiling example (fast): 1 step left + 1 step right
res <- profile_likelihood(
  profile_parameter = "mu1",
  increment_left = 0.2,
  increment_right = 0.2,
  num_steps_left = 1L, # one iteration on the left
  num_steps_right = 1L, # one iteration on the right
  alpha = 0.95,
```

```

optim_param_vector = example_1$optim_par_vec,
env_dat = example_1$env_array,
occ = example_1$occ_vec,
num_threads = 1L, # keep it fast and deterministic
control = list(maxeval = 20),
verbose = FALSE
)
# Check the structure of the output:
res$profile
res$threshold
res$found_better
## Full math-scale parameter vectors used at each evaluated point:
res$parameter_df

```

start_parms

Starting parameters for the optimization

Description

Generate starting parameters for the optimization of the xsdm log-likelihood. Starting points are constructed from the environmental conditions at observed presences (where `occ == 1`) using a Latin hypercube design for the parameters based on the Sobol' low-discrepancy sequence.

Usage

```
start_parms(env_dat, mask = NULL, breadth = 1, num_starts = 100)
```

Arguments

env_dat	The environmental array for only the observed occurrences
mask	Either NULL or a named numeric vector. Names must be as specified by calling <code>make_mask_names</code> . The NULL case means planned optimizations will be over all model parameters, so start parameter sets should include all model parameters. The non-NULL case means some parameters will not be specified in the output of this function because the optimizations which are planned will fix those parameters anyway. The most common case for most applications will be <code>mask=NULL</code> .
breadth	Scalar in $[0, 1]$ controlling how wide the search ranges are around their (fixed, data-driven) center. <code>breadth = 1</code> (the default) reproduces the pre-v0.3 behaviour that corresponded to <code>quant_vec = c(0.1, 0.5, 0.9)</code> ; <code>breadth = 0</code> collapses every range to essentially a single point, equivalent to <code>quant_vec = c(0.5 - 1e-6, 0.5, 0.5 + 1e-6)</code> . Values in between interpolate linearly.
num_starts	The number of samples of the hypercube

Details

The bounds and center of the search range for a μ parameter are based on the quantiles in `quant_vec` applied to all observations of that environmental variable, over space and time. The relationship between `quant_vec` and the width of the ranges selected for the other parameters varies, but generally wider ranges in `quant_vec` produce wider ranges for start parameters.

Value

A data frame with samples for each parameter to optimize

Examples

```
env_dat <- example_1$env_array[example_1$occ_vec == 1, , ]
start_parms(env_dat)
start_parms(env_dat, mask = c(mu2 = 5, pd = 1))
```

vsp	<i>Generate a virtual species probability map with presence/absence sampling</i>
-----	--

Description

Creates a virtual species probability-of-detection map based on environmental time-series data and a set of species-specific parameters, then samples presence/absence points based on a user-defined probability threshold.

Usage

```
vsp(param_list, env_data, size_presence, size_absence, threshold = 0.5)
```

Arguments

param_list	A named list of biological-scale parameters required by <code>'log_prob_detect()'</code> . Must include <code>'mu'</code> , <code>'sigltl'</code> , <code>'sigrtl'</code> , <code>'ctil'</code> , <code>'pd'</code> , and <code>'o_mat'</code> . Values like <code>'sigltl'</code> / <code>'sigrtl'</code> can be <code>'Inf'</code> .
env_data	A named list of time-series raster objects (e.g., from the <code>'terra'</code> package). Each element must be a <code>'SpatRaster'</code> with the same geometry and number of layers.
size_presence	Integer. Number of sample points to draw from cells where the detection probability exceeds <code>'threshold'</code> .
size_absence	Integer. Number of sample points to draw from cells where the detection probability is less than or equal to <code>'threshold'</code> .
threshold	Numeric in <code>'[0, 1]'</code> . Probability cutoff used to distinguish presence vs. absence sampling areas. Default <code>'0.5'</code> .

Details

Internally the function:

1. Computes a habitat suitability raster using `'habitat_suitability()'`.
2. Splits the raster into two layers based on `'threshold'`: cells with `prob > threshold` (presence pool) and `≤ threshold` (absence pool).
3. Samples `'size_presence'` and `'size_absence'` points from each pool (without replacement), with probabilities proportional to the suitability value.
4. Generates a binomial outcome for each sampled point using its suitability as the probability of success.

Value

A tibble with columns `'lon'`, `'lat'`, `'presence'` (0/1), where each row corresponds to a sampled point. The presence/absence is drawn from a binomial distribution using the habitat suitability value as the success probability.

See Also

`[habitat_suitability()]`, `[log_prob_detect()]`, `[terra::spatSample()]`

Examples

```
data("example_1", package = "xsgm")
bio1_ts <- terra::unwrap(example_1$bio01) / 100
bio12_ts <- terra::unwrap(example_1$bio12) / 100
env_data <- list(bio1 = bio1_ts, bio12 = bio12_ts)

vsp(
  param_list = example_1$true_par_list,
  env_data = env_data,
  size_presence = 100,
  size_absence = 100,
  threshold = 0.7
)
```

Index

- * **datasets**
 - example_1, [11](#)
 - example_2, [13](#)
 - example_3, [14](#)
- bio_to_math, [3](#), [34](#)
- build_orthogonal_matrix, [5](#), [33](#)
- convert_equivalence_class, [6](#)
- create_mask, [7](#)
- create_param_vector_masked, [8](#), [29](#)
- dist_between_params, [9](#)
- env_data_array, [11](#)
- example_1, [11](#)
- example_2, [13](#)
- example_3, [14](#)
- expit, [14](#), [33](#)
- expm1, [25](#), [26](#)
- extract_orthogonal_matrix_parameters,
[15](#)
- habitat_suitability, [16](#)
- interpret_parameters, [17](#)
- like_ltsg, [20](#)
- like_neg_ltsg, [21](#)
- log1mexp, [24](#), [26](#)
- log1p, [25](#), [26](#)
- log1pexp, [25](#), [25](#)
- log_prob_detect, [16](#), [17](#), [22](#)
- log_prob_detect_cpp, [16](#), [17](#), [23](#)
- loglik_bio, [26](#)
- loglik_bio_cpp, [27](#)
- loglik_math, [10](#), [28](#), [34](#)
- loglik_math_cpp, [31](#)
- make_mask_names, [29](#), [32](#), [33](#), [34](#)
- math_to_bio, [29](#), [33](#)
- num_env_var, [34](#), [34](#)
- num_par, [29](#), [33](#), [34](#), [35](#)
- on.exit, [17](#)
- optimize_likelihood, [36](#)
- profile_likelihood, [37](#)
- readStart, [17](#)
- readStop, [17](#)
- readValues, [17](#)
- SpatRaster, [16](#)
- start_parms, [10](#), [39](#)
- vsp, [17](#), [40](#)
- writeStart, [16](#), [17](#)
- writeStop, [17](#)
- writeValues, [17](#)
- xsdm (xsdm-package), [3](#)
- xsdm-package, [3](#)