Package 'maxentcpp'

May 18, 2026

Type Package

Title Maximum Entropy Species Distribution Modeling ('C++' Implementation)

Version 1.0.0

Description 'C++' implementation of Maximum Entropy (Maxent) species distribution modeling with R bindings via 'Rcpp'. Provides a high-performance reimplementation of the Maxent algorithm for modeling species geographic distributions from occurrence data and environmental variables, following Phillips et al. (2006) <doi:10.1016/j.ecolmodel.2005.03.026>. Supports linear, quadratic, product, hinge, and threshold feature transformations, spatial projection in raw, logistic, and cloglog scales, and model diagnostics including Area Under the ROC Curve (AUC), variable importance, response curves, and Multivariate Environmental Similarity Surfaces (MESS) maps.

URL https://github.com/alrobles/maxentcpp, https://alrobles.github.io/maxentcpp/

BugReports https://github.com/alrobles/maxentcpp/issues

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.0), methods

LinkingTo Rcpp, RcppEigen

Suggests testthat (>= 3.0.0), knitr, rmarkdown, terra (>= 1.7-46), raster, sf, png, rJava, maxnet

SystemRequirements C++17

Encoding UTF-8

LazyData true

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation yes

Author Angel Robles [aut, cre]

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

Repository CRAN

Date/Publication 2026-05-18 18:30:11 UTC

Contents

`clamp_grids` *Clamp Environmental Grids*

Description

Restricts environmental variable values to the training range. Returns clamped grids and a clamping indicator grid.

Usage

```
clamp_grids(grid_ptrs, var_mins, var_maxs)
```

Arguments

`grid_ptrs`	List of external pointers to Grid<float> objects.
`var_mins`	Numeric vector of minimum training values per variable.
`var_maxs`	Numeric vector of maximum training values per variable.

Value

A named list with:

clamped_grids List of external pointers to clamped Grid<float>

clamp_grid External pointer to Grid<float> with clamping magnitudes

compute_mess *Compute MESS (Multivariate Environmental Similarity Surface)*

Description

Measures how similar each cell is to the training environment. Negative values indicate novel (non-analog) conditions.

Usage

```
compute_mess(grid_ptrs, reference_values, feature_names)
```

Arguments

grid_ptrs	List of external pointers to Grid<float> objects.
reference_values	List of numeric vectors with reference values for each variable (e.g. values at training sites).
feature_names	Character vector of variable names.

Value

A named list with:

mess_grid External pointer to Grid<float> with MESS values

mod_grid External pointer to Grid<float> with MoD variable index (1-based)

compute_mess_range *Compute MESS from Min/Max Ranges*

Description

Simplified MESS using only the min/max of reference data.

Usage

```
compute_mess_range(grid_ptrs, var_mins, var_maxs)
```

Arguments

grid_ptrs	List of external pointers to Grid<float> objects.
var_mins	Numeric vector of minimum reference values.
var_maxs	Numeric vector of maximum reference values.

Value

A named list with mess_grid and mod_grid (external pointers).

`compute_percent_contribution` *Compute Percent Contribution*

Description

Computes variable contribution based on sum of absolute lambda values for features derived from each variable.

Usage

```
compute_percent_contribution(fs_ptr, feature_names)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`feature_names`	Character vector of base variable names.

Value

A data.frame with columns: name, contribution.

`compute_permutation_importance` *Compute Permutation Importance*

Description

Measures how much each environmental variable contributes to model prediction quality by permuting each variable and measuring AUC drop.

Usage

```
compute_permutation_importance(
  fs_ptr,
  grid_ptrs,
  feature_names,
  presence_rows,
  presence_cols,
  absence_rows,
  absence_cols,
  seed = 42L
)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.
grid_ptrs	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
presence_rows	Integer vector of presence site row indices.
presence_cols	Integer vector of presence site column indices.
absence_rows	Integer vector of absence site row indices.
absence_cols	Integer vector of absence site column indices.
seed	Random seed for reproducibility.

Value

A data.frame with columns: name, permutation_importance.

---

compute_response_curve

Compute Java-compatible Marginal Response Curve

Description

Like compute_response_curve() but applies the Java Maxent cloglog: cloglog_java = 1 - exp(-exp(H) * raw_java)

Usage

```
compute_response_curve(
  fs_ptr,
  grid_ptrs,
  feature_names,
  var_index,
  n_steps = 100L
)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.
grid_ptrs	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
var_index	0-based index of the variable to vary.
n_steps	Number of steps across the variable range.

Details

This matches the response curves produced by Java Maxent and dismo.

Value

A data.frame with columns: value, prediction.

`compute_response_curve_fixed` *Compute Java-compatible Response Curve with Fixed Values*

Description

Like compute_response_curve_fixed() but applies the Java Maxent cloglog: cloglog_java = 1 - exp(-exp(H) * raw_java)

Usage

```
compute_response_curve_fixed(
  fs_ptr,
  fixed_values,
  feature_names,
  var_index,
  var_min,
  var_max,
  n_steps = 100L
)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`fixed_values`	Numeric vector of fixed values for each variable.
`feature_names`	Character vector of environment variable names.
`var_index`	0-based index of the variable to vary.
`var_min`	Minimum value of the target variable.
`var_max`	Maximum value of the target variable.
`n_steps`	Number of steps.

Value

A data.frame with columns: value, prediction.

compute_variable_ranges *Compute Variable Ranges from Grids*

Description

Scans all valid cells to determine min/max of each variable.

Usage

```
compute_variable_ranges(grid_ptrs)
```

Arguments

grid_ptrs	List of external pointers to Grid<float> objects.

Value

A data.frame with columns: min, max.

coords_to_rowcol *Convert geographic coordinates to row/col*

Description

Convert geographic coordinates to row/col

Usage

```
coords_to_rowcol(dim_ptr, lon, lat)
```

Arguments

dim_ptr	External pointer to GridDimension
lon	Longitude
lat	Latitude

Value

Integer vector with row and column indices

create_grid_dimension *Create a GridDimension object*

Description

Creates a grid dimension specifying the spatial extent and resolution

Usage

```
create_grid_dimension(nrows, ncols, xll, yll, cellsize)
```

Arguments

nrows	Number of rows
ncols	Number of columns
xll	X coordinate of lower-left corner
yll	Y coordinate of lower-left corner
cellsize	Cell size (assumed square cells)

Value

External pointer to GridDimension object

create_grid_float *Create a float Grid object*

Description

Creates a raster grid for environmental variables

Usage

```
create_grid_float(dim_ptr, name, nodata_value = -9999)
```

Arguments

dim_ptr	External pointer to GridDimension
name	Grid layer name
nodata_value	Value representing missing data

Value

External pointer to Grid<float> object

create_hinge_feature *Create a HingeFeature object*

Description

Creates a piecewise-linear hinge feature. Forward hinge: eval(i) = (values[i] > min_knot) ? (values[i] - min_knot) / (max_knot - min_knot) : 0. Reverse hinge: eval(i) = (values[i] < max_knot) ? (max_knot - values[i]) / (max_knot - min_knot) : 0.

Usage

```
create_hinge_feature(values, name, min_knot, max_knot, is_reverse = FALSE)
```

Arguments

values	Numeric vector of environmental variable values
name	Feature name/identifier
min_knot	Lower knot of the hinge
max_knot	Upper knot of the hinge (must be > min_knot)
is_reverse	If TRUE, use reverse hinge; if FALSE (default), use forward hinge

Value

External pointer to HingeFeature object

create_layer *Create a Layer metadata object*

Description

Create a Layer metadata object

Usage

```
create_layer(name, type_str)
```

Arguments

name	Layer name.
type_str	Type string: "Continuous", "Categorical", "Bias", "Mask", "Probability", "Cumulative", "DebiasAvg", or "Unknown".

Value

External pointer to a Layer object.

create_linear_feature *Create a LinearFeature object*

Description

Creates a linear (normalized) feature: eval(i) = (values[i] - min) / (max - min). Returns 0 when min == max.

Usage

```
create_linear_feature(values, name, min_val, max_val)
```

Arguments

values	Numeric vector of environmental variable values
name	Feature name/identifier
min_val	Minimum value for normalization
max_val	Maximum value for normalization

Value

External pointer to LinearFeature object

create_product_feature *Create a ProductFeature object*

Description

Creates a product (interaction) feature between two environmental variables: eval(i) = norm(values1[i]) * norm(values2[i]).

Usage

```
create_product_feature(values1, values2, name, min1, max1, min2, max2)
```

Arguments

values1	Numeric vector for the first environmental variable
values2	Numeric vector for the second environmental variable
name	Feature name/identifier
min1	Minimum of the first variable
max1	Maximum of the first variable
min2	Minimum of the second variable
max2	Maximum of the second variable

Value

External pointer to ProductFeature object

`create_quadratic_feature` *Create a QuadraticFeature object*

Description

Creates a quadratic feature: eval(i) = linear_val^2 where linear_val = (values[i] - min) / (max - min).

Usage

```
create_quadratic_feature(values, name, min_val, max_val)
```

Arguments

`values`	Numeric vector of environmental variable values
`name`	Feature name/identifier
`min_val`	Minimum value for normalization
`max_val`	Maximum value for normalization

Value

External pointer to QuadraticFeature object

`create_sample` *Create a Sample object*

Description

Creates a species occurrence sample point

Usage

```
create_sample(point, row, col, lat, lon, name)
```

Arguments

`point`	Point identifier
`row`	Row index in grid
`col`	Column index in grid
`lat`	Latitude coordinate
`lon`	Longitude coordinate
`name`	Sample name/identifier

Value

External pointer to Sample object

`create_threshold_feature` *Create a ThresholdFeature object*

Description

Creates a binary step feature: eval(i) = 1.0 if values[i] > threshold, else 0.0.

Usage

```
create_threshold_feature(values, name, threshold)
```

Arguments

`values`	Numeric vector of environmental variable values
`name`	Feature name/identifier
`threshold`	The threshold value

Value

External pointer to ThresholdFeature object

`csv_close` *Close a CsvReader*

Description

Close a CsvReader

Usage

```
csv_close(reader_ptr)
```

Arguments

`reader_ptr`	External pointer to a CsvReader object.

Value

Called for side effects; returns invisibly.

csv_headers *Get CSV column headers*

Description

Get CSV column headers

Usage

```
csv_headers(reader_ptr)
```

Arguments

reader_ptr	External pointer to a CsvReader object.

Value

Character vector of header names.

csv_next_record *Read the next CSV record*

Description

Read the next CSV record

Usage

```
csv_next_record(reader_ptr)
```

Arguments

reader_ptr	External pointer to a CsvReader object.

Value

Character vector of field values, or NULL on EOF.

csv_open *Open a CSV file for reading*

Description

Open a CSV file for reading

Usage

```
csv_open(filename, has_header = TRUE)
```

Arguments

filename	Path to the CSV file.
has_header	Logical: first line contains column names (default TRUE).

Value

External pointer to a CsvReader object.

csv_read_double_column *Read an entire named column as doubles*

Description

Reads from the current position to EOF.

Usage

```
csv_read_double_column(reader_ptr, field)
```

Arguments

reader_ptr	External pointer to a CsvReader object.
field	Column name.

Value

Numeric vector.

csv_writer_close *Close a CsvWriter*

Description

Close a CsvWriter

Usage

```
csv_writer_close(writer_ptr)
```

Arguments

writer_ptr	External pointer to a CsvWriter object.

Value

Called for side effects; returns invisibly.

csv_writer_open *Open a CSV file for writing*

Description

Open a CSV file for writing

Usage

```
csv_writer_open(filename, append = FALSE, precision = 4L)
```

Arguments

filename	Path to the output CSV file.
append	Logical: append to existing file (default FALSE).
precision	Number of decimal places for doubles (default 4).

Value

External pointer to a CsvWriter object.

csv_writer_print *Write a value into the current row of a CSV*

Description

Write a value into the current row of a CSV

Usage

```
csv_writer_print(writer_ptr, column, value)
```

Arguments

writer_ptr	External pointer to a CsvWriter object.
column	Column name.
value	Value to write (character).

Value

Called for side effects; returns invisibly.

csv_writer_println *Flush the current CSV row to disk*

Description

Flush the current CSV row to disk

Usage

```
csv_writer_println(writer_ptr)
```

Arguments

writer_ptr	External pointer to a CsvWriter object.

Value

Called for side effects; returns invisibly.

csv_writer_print_double

Write a numeric value into the current row of a CSV

Description

Write a numeric value into the current row of a CSV

Usage

```
csv_writer_print_double(writer_ptr, column, value)
```

Arguments

writer_ptr	External pointer to a CsvWriter object.
column	Column name.
value	Numeric value.

Value

Called for side effects; returns invisibly.

eval_auc

Compute AUC (Area Under the ROC Curve)

Description

Computes the Wilcoxon-Mann-Whitney AUC statistic from prediction scores at presence and absence sites.

Usage

```
eval_auc(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

A named list with elements: auc, max_kappa, max_kappa_thresh.

eval_correlation *Compute Pearson Correlation*

Description

Computes Pearson correlation coefficient between two numeric vectors.

Usage

```
eval_correlation(x, y)
```

Arguments

x	Numeric vector.
y	Numeric vector (same length as x).

Value

Correlation coefficient in [-1, 1].

eval_logloss *Compute Log-Loss*

Description

Computes average cross-entropy log-loss from predictions at presence and absence sites.

Usage

```
eval_logloss(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

Average log-loss value.

eval_misclassification *Compute Misclassification Rate*

Description

Fraction of misclassified samples at threshold 0.5.

Usage

```
eval_misclassification(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

Misclassification rate in [0, 1].

eval_model *Full Model Evaluation*

Description

Computes all evaluation metrics at once: AUC, correlation, log-loss, squared error, misclassification, max kappa, and prevalence.

Usage

```
eval_model(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

A named list with: auc, max_kappa, max_kappa_thresh, correlation, square_error, logloss, misclassification, prevalence.

eval_square_error *Compute Mean Squared Error*

Description

Presence sites contribute (1-pred)^2, absence sites contribute pred^2.

Usage

```
eval_square_error(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

Mean squared error.

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

long Decimal longitude geographical coordinate, in degrees

lat Decimal latitude geographical coordinate, in degrees

Source

<doi:10.1016/j.ecolmodel.2021.109823>

extract_predictions_raw

Extract Java-compatible Raw Predictions at Sample Locations

Description

Gets Java Maxent raw scores (raw_java = exp(lp - lpNorm) / densityNorm) at specific grid cell locations.

Usage

```
extract_predictions_raw(fs_ptr, grid_ptrs, feature_names, rows, cols)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.
grid_ptrs	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
rows	Integer vector of row indices.
cols	Integer vector of column indices.

Value

Numeric vector of Java raw scores. NaN for NODATA cells.

feature_eval

Evaluate a feature at a given index

Description

Evaluate a feature at a given index

Usage

```
feature_eval(feature_ptr, index)
```

Arguments

feature_ptr	External pointer to a Feature object
index	0-based index into the data vector

Value

Feature value at that index (double)

feature_get_info *Get feature metadata*

Description

Returns a list with name, type, lambda, min, and max values.

Usage

```
feature_get_info(feature_ptr)
```

Arguments

feature_ptr	External pointer to a Feature object

Value

Named list with feature properties

generate_features *Generate features from a list of environmental variable vectors*

Description

Generates all configured feature types (linear, quadratic, product, threshold, hinge) from the supplied data vectors.

Usage

```
generate_features(
  data_list,
 feature_types = as.character(c("linear", "quadratic", "product", "threshold", "hinge")),
  n_thresholds = 10L,
  n_hinges = 10L
)
```

Arguments

data_list	Named list of numeric vectors, one per environmental variable
feature_types	Character vector of feature types to generate. Valid values: "linear", "quadratic", "product", "threshold", "hinge". Defaults to all types.
n_thresholds	Number of threshold knots per variable (default: 10)
n_hinges	Number of hinge knots per variable (default: 10)

Value

List of external pointers to Feature objects

get_grid_dimension_info *Get grid dimension properties*

Description

Get grid dimension properties

Usage

```
get_grid_dimension_info(dim_ptr)
```

Arguments

dim_ptr	External pointer to GridDimension

Value

List with dimension properties

get_layer_info *Get layer metadata*

Description

Get layer metadata

Usage

```
get_layer_info(layer_ptr)
```

Arguments

layer_ptr	External pointer to a Layer object.

Value

Named list with name and type (string).

get_sample_info *Get sample properties*

Description

Get sample properties

Usage

```
get_sample_info(sample_ptr)
```

Arguments

sample_ptr	External pointer to Sample

Value

List with sample properties

grid_float_from_matrix *Create a GridFloat from an R matrix*

Description

Create a GridFloat from an R matrix

Usage

```
grid_float_from_matrix(mat, xll, yll, cellsize, nodata = -9999, name = "")
```

Arguments

mat	Numeric matrix with grid data (NA becomes NODATA).
xll	X coordinate of lower-left corner.
yll	Y coordinate of lower-left corner.
cellsize	Cell size.
nodata	NODATA value (default -9999).
name	Grid name (default "").

Value

External pointer to a GridFloat object.

grid_float_info *Get grid information for an ASC-loaded grid*

Description

Returns metadata and basic statistics for a GridFloat.

Usage

```
grid_float_info(grid_ptr)
```

Arguments

grid_ptr	External pointer to a GridFloat object.

Value

Named list with nrows, ncols, xll, yll, cellsize, nodata, name, count_data.

grid_float_to_matrix *Extract the data from a GridFloat as an R matrix*

Description

Extract the data from a GridFloat as an R matrix

Usage

```
grid_float_to_matrix(grid_ptr)
```

Arguments

grid_ptr	External pointer to a GridFloat object.

Value

Numeric matrix (nrows × ncols). NODATA cells are set to NA.

grid_from_matrix *Set grid from matrix*

Description

Set grid from matrix

Usage

```
grid_from_matrix(grid_ptr, mat)
```

Arguments

grid_ptr	External pointer to Grid
mat	Numeric matrix of values

Value

Called for side effects; returns invisibly.

grid_get_value *Get grid value*

Description

Get grid value

Usage

```
grid_get_value(grid_ptr, row, col)
```

Arguments

grid_ptr	External pointer to Grid
row	Row index
col	Column index

Value

Grid value

grid_get_values_batch *Extract multiple values from a Grid at given row/col indices (batch)*

Description

Extract multiple values from a Grid at given row/col indices (batch)

Usage

```
grid_get_values_batch(grid_ptr, rows, cols)
```

Arguments

grid_ptr	External pointer to a Grid<float>.
rows	Integer vector of 0-based row indices.
cols	Integer vector of 0-based column indices.

Value

Numeric vector of extracted values (NA for NODATA cells).

grid_has_data *Check if grid cell has data*

Description

Check if grid cell has data

Usage

```
grid_has_data(grid_ptr, row, col)
```

Arguments

grid_ptr	External pointer to Grid
row	Row index
col	Column index

Value

TRUE if cell has valid data

grid_read_asc *Read an ESRI ASCII grid (.asc) file*

Description

Reads an ASC raster file and returns a float Grid external pointer.

Usage

```
grid_read_asc(filename)
```

Arguments

filename	Path to the .asc file.

Value

External pointer to a GridFloat object.

grid_read_file *Read a grid from file (auto-detect format)*

Description

Currently supports .asc format. Returns a float Grid.

Usage

```
grid_read_file(filename)
```

Arguments

filename	Path to the grid file.

Value

External pointer to a GridFloat object.

grid_set_value *Set grid value*

Description

Set grid value

Usage

```
grid_set_value(grid_ptr, row, col, value)
```

Arguments

grid_ptr	External pointer to Grid
row	Row index
col	Column index
value	Value to set

Value

Called for side effects; returns invisibly.

grid_to_matrix *Get grid as matrix*

Description

Get grid as matrix

Usage

```
grid_to_matrix(grid_ptr)
```

Arguments

grid_ptr	External pointer to Grid

Value

Numeric matrix of grid values

grid_write_asc *Write a float Grid to an ESRI ASCII (.asc) file*

Description

Write a float Grid to an ESRI ASCII (.asc) file

Usage

```
grid_write_asc(grid_ptr, filename, scientific = TRUE)
```

Arguments

grid_ptr	External pointer to a GridFloat object.
filename	Output file path.
scientific	Logical: use scientific notation (default TRUE).

Value

Called for side effects; returns invisibly.

layer_name_from_path *Extract a layer name from a file path*

Description

Strips directory and extension, e.g. "/data/bio1.asc" → "bio1".

Usage

```
layer_name_from_path(path)
```

Arguments

path	File path.

Value

Character: layer name.

`maxent_append_results_csv` *Append a Row to maxentResults.csv*

Description

Appends one row of species-level summary statistics to a `maxentResults.csv` file, creating the file (with a header) if it does not yet exist. Column names match the Java Maxent output for cross-tool compatibility.

Usage

```
maxent_append_results_csv(
  results_file,
  species,
  n_training,
  n_test = 0L,
  training_gain,
  training_auc,
  test_gain = NA_real_,
  test_auc = NA_real_,
  entropy,
  contributions_df,
  perm_imp_df
)
```

Arguments

`results_file`	Character: path to the results CSV file.
`species`	Character: species name.
`n_training`	Integer: number of training presence points.
`n_test`	Integer: number of test presence points (default 0).
`training_gain`	Numeric: regularized training gain.
`training_auc`	Numeric: training AUC.
`test_gain`	Numeric or `NA`: regularized test gain.
`test_auc`	Numeric or `NA`: test AUC.
`entropy`	Numeric: model entropy.
`contributions_df`	A data.frame with columns `name` and `contribution` (from `maxent_percent_contribution`).
`perm_imp_df`	A data.frame with columns `name` and `permutation_importance` (from `maxent_permutation_importan`

Value

Invisibly returns `results_file`.

Examples

```
maxent_append_results_csv(
  file.path(tempdir(), "maxentResults.csv"),
  species = "Sp1", n_training = 50L, training_gain = 1.23,
  training_auc = 0.95, entropy = 6.7,
  contributions_df = contrib, perm_imp_df = perm_imp)
```

`maxent_auc` *Compute AUC (Area Under the ROC Curve)*

Description

Computes the Wilcoxon-Mann-Whitney AUC statistic from prediction scores at presence and absence sites. Also returns max-Kappa and its threshold.

Usage

```
maxent_auc(presence, absence)
```

Arguments

`presence`	Numeric vector of prediction scores at presence sites.
`absence`	Numeric vector of prediction scores at absence sites.

Value

A named list with elements:

auc AUC value in [0, 1]

max_kappa Best Cohen's Kappa found

max_kappa_thresh Threshold at best Kappa

Examples

```
result <- maxent_auc(c(0.8, 0.9, 1.0), c(0.1, 0.2, 0.3))
result$auc  # 1.0
```

`maxent_background_indices` *Generate Background Sample Indices*

Description

Randomly selects valid (non-NODATA) cells from a reference grid for use as background points in MaxEnt modeling.

Usage

```
maxent_background_indices(grid, n = 10000L, seed = NULL)
```

Arguments

`grid`	External pointer to a GridFloat object used as a reference (e.g. an environmental layer). Only cells with valid data are eligible.
`n`	Integer: number of background points to sample (default `10000L`).
`seed`	Integer or `NULL`: random seed for reproducibility. If `NULL` (default), no seed is set.

Value

A named list with:

rows Integer vector of row indices (0-based).

cols Integer vector of column indices (0-based).

indices Integer vector of 0-based flat indices (`row * ncols + col`), suitable for `maxent_featured_space`.

Examples

```
dim <- maxent_dimension(100, 100, -120, 35, 0.1)
grid <- maxent_grid(dim, "env1")
bg <- maxent_background_indices(grid, n = 5000, seed = 42)
bg$indices  # 0-based flat indices for FeaturedSpace
```

maxent_clamp *Clamp Environmental Grids*

Description

Restricts environmental variable values to the range observed during training. Values below min are set to min, values above max are set to max. A clamping grid records the total absolute clamping at each cell.

Usage

```
maxent_clamp(env_grids, var_mins, var_maxs)
```

Arguments

env_grids	List of external pointers to Grid<float> objects.
var_mins	Numeric vector of minimum training values per variable.
var_maxs	Numeric vector of maximum training values per variable.

Value

A named list with:

clamped_grids List of external pointers to clamped grids

clamp_grid External pointer to clamping magnitude grid

Examples

```
result <- maxent_clamp(list(g1, g2), c(0, 50), c(30, 200))
clamped <- result$clamped_grids
clamp_mat <- maxent_grid_to_matrix(result$clamp_grid)
```

maxent_color_ramp *Generate the Maxent Canonical Color Ramp*

Description

Produces a vector of hex color strings matching the color ramp used by the Java Maxent software (`density/Display.java::showColor()`). The default palette cycles Red → Yellow → Green → Cyan → Blue across 1020 steps, with higher values rendered as red and lower values as blue.

Usage

```
maxent_color_ramp(n = 1020L, mode = "plain")
```

Arguments

n	Integer: number of colors to generate (default 1020, matching the Java implementation).
mode	Character: one of "plain" (default), "log" (logarithmic spacing), "blackandwhite" (greyscale, white = high, black = low), or "redandyellow" (red–yellow ramp).

Value

Character vector of n hex color strings of the form "#RRGGBB".

Examples

```
pal <- maxent_color_ramp(1020)
pal[1]     # "#FF0000"  (max value → red)
pal[510]   # near "#00FF00" (mid-point → green)
pal[1020]  # "#0000FF"  (min value → blue)
```

---

maxent_correlation *Compute Pearson Correlation*

Description

Computes the Pearson correlation coefficient between two numeric vectors.

Usage

```
maxent_correlation(x, y)
```

Arguments

x	Numeric vector.
y	Numeric vector (same length as x).

Value

Correlation coefficient in [-1, 1].

Examples

```
maxent_correlation(c(1, 2, 3), c(2, 4, 6))  # 1.0
```

maxent_csv_close *Close a CSV Reader*

Description

Close a CSV Reader

Usage

```
maxent_csv_close(reader)
```

Arguments

reader	External pointer to a CsvReader object.

Value

Invisibly returns NULL.

maxent_csv_headers *Get CSV Column Headers*

Description

Get CSV Column Headers

Usage

```
maxent_csv_headers(reader)
```

Arguments

reader	External pointer to a CsvReader object.

Value

Character vector of column names.

maxent_csv_next *Read the Next CSV Record*

Description

Read the Next CSV Record

Usage

```
maxent_csv_next(reader)
```

Arguments

reader	External pointer to a CsvReader object.

Value

Character vector of field values, or NULL on EOF.

maxent_csv_open *Open a CSV File for Reading*

Description

Opens a CSV file and reads column headers.

Usage

```
maxent_csv_open(filename, has_header = TRUE)
```

Arguments

filename	Character: path to the CSV file.
has_header	Logical: first line is header (default TRUE).

Value

External pointer to a CsvReader C++ object.

maxent_csv_read_column *Read an Entire Column as Doubles*

Description

Reads from the current file position to EOF.

Usage

```
maxent_csv_read_column(reader, field)
```

Arguments

`reader`	External pointer to a CsvReader object.
`field`	Character: column name.

Value

Numeric vector.

maxent_csv_write *Write a String Value to the Current CSV Row*

Description

Adds a `column = value` pair (as a character string) to the current row buffer. Call `maxent_csv_write_row` to flush the row.

Usage

```
maxent_csv_write(writer, column, value)
```

Arguments

`writer`	External pointer to a CsvWriter object.
`column`	Character: column name.
`value`	Character: value to write.

Value

Invisibly returns the writer object.

maxent_csv_write_close *Close a CSV Writer*

Description

Flushes any pending data and closes the CSV file.

Usage

```
maxent_csv_write_close(writer)
```

Arguments

writer	External pointer to a CsvWriter object.

Value

Invisibly returns NULL.

maxent_csv_write_num *Write a Numeric Value to the Current CSV Row*

Description

Adds a column = value pair (as a double) to the current row buffer. Call maxent_csv_write_row to flush the row.

Usage

```
maxent_csv_write_num(writer, column, value)
```

Arguments

writer	External pointer to a CsvWriter object.
column	Character: column name.
value	Numeric: value to write.

Value

Invisibly returns the writer object.

maxent_csv_write_open *Open a CSV File for Writing*

Description

Opens a CSV file and returns a writer object. Use maxent_csv_write, maxent_csv_write_num, maxent_csv_write_row, and maxent_csv_write_close to write data and close the file.

Usage

```
maxent_csv_write_open(filename, append = FALSE, precision = 4L)
```

Arguments

filename	Character: output file path.
append	Logical: append to an existing file (default FALSE).
precision	Integer: number of decimal places for numeric values (default 4).

Value

External pointer to a CsvWriter C++ object.

maxent_csv_write_row *Flush the Current Row to the CSV File*

Description

Writes all buffered column values as a single CSV row and starts a new row buffer.

Usage

```
maxent_csv_write_row(writer)
```

Arguments

writer	External pointer to a CsvWriter object.

Value

Invisibly returns the writer object.

maxent_dimension *Create a Grid Dimension Object*

Description

Defines the spatial extent and resolution of a raster grid.

Usage

```
maxent_dimension(nrows, ncols, xll, yll, cellsize)
```

Arguments

nrows	Number of rows
ncols	Number of columns
xll	X coordinate of lower-left corner
yll	Y coordinate of lower-left corner
cellsize	Cell size (square cells)

Value

GridDimension object (external pointer)

Examples

```
dim <- maxent_dimension(
  nrows = 100, ncols = 100,
  xll = -120.0, yll = 35.0,
  cellsize = 0.1
)
```

maxent_evaluate *Full Model Evaluation*

Description

Computes all evaluation metrics at once: AUC, correlation, log-loss, squared error, misclassification, max Kappa, and prevalence.

Usage

```
maxent_evaluate(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

A named list with:

auc AUC value

max_kappa Best Cohen's Kappa

max_kappa_thresh Threshold at best Kappa

correlation Pearson correlation with labels

square_error Mean squared error

logloss Cross-entropy log-loss

misclassification Misclassification rate

prevalence Fraction of presence sites

Examples

```
res <- maxent_evaluate(c(0.9, 0.85, 0.95), c(0.1, 0.15, 0.2))
res$auc          # 1.0
res$correlation  # > 0
```

---

maxent_extract_lambdas

Extract Lambda Values from a Maxent Run Result

Description

Retrieves the trained lambda (weight) coefficients from the output of maxent_run, returning them as a named numeric vector. Each element is the lambda value for the corresponding feature, and the name of each element is the feature name (e.g. "bio1", "bio1^2", "bio1'1").

Usage

```
maxent_extract_lambdas(run_result)
```

Arguments

run_result	A named list as returned by maxent_run. Must contain an element model holding the trained FeaturedSpace external pointer.

Details

The function reads lambdas directly from the trained model object, so it also works correctly after reloading coefficients with maxent_load_lambdas.

Value

A named numeric vector of lambda values, one per feature. Names are the feature names as stored in the model.

Examples

```
stack_path      <- system.file("extdata", "stack_1_12_crop.rds",
                               package = "maxentcpp")
example_rasters <- terra::unwrap(readRDS(stack_path))
grids <- list(
  bio1  = maxent_grid_from_terra(example_rasters[[1]]),
  bio12 = maxent_grid_from_terra(example_rasters[[2]])
)
data(example_occ_df)

result <- maxent_run(
  species    = "Abeillia_abeillei",
  env_grids  = grids,
  occ_df     = example_occ_df,
  output_dir = tempdir(),
  lon_col    = "long",
  lat_col    = "lat")

lambdas <- maxent_extract_lambdas(result)
print(lambdas)
```

`maxent_extract_occurrence_env_terra`

Extract environmental values for occurrences from a terra SpatRaster

---

Description

Streams finite raster rows through the same callback-backed provider path used by `maxent_featured_space_from_rast`, then returns values at occurrence locations.

Usage

```
maxent_extract_occurrence_env_terra(
  rast,
  occurrences,
  lon_col = "longitude",
  lat_col = "latitude"
)
```

Arguments

rast	A terra::SpatRaster.
occurrences	Either a two-column matrix/data.frame (x,y) or a vector of 1-based raster cell indices.
lon_col	Longitude column name when occurrences is a data.frame.
lat_col	Latitude column name when occurrences is a data.frame.

Value

Numeric matrix with one row per retained occurrence and one column per raster layer.

---

maxent_extract_predictions_cloglog

Extract Cloglog Predictions at Sample Locations

Description

Gets Java Maxent cloglog scores at specific grid cell locations:

$$cloglog = 1 - exp(-exp(H) \cdot raw)$$

Usage

```
maxent_extract_predictions_cloglog(model, env_grids, feature_names, rows, cols)
```

Arguments

model	External pointer to a FeaturedSpace object.
env_grids	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
rows	Integer vector of row indices.
cols	Integer vector of column indices.

Value

Numeric vector of cloglog scores in [0, 1]. NaN for NODATA cells.

See Also

maxent_extract_predictions_raw, maxent_project_cloglog

`maxent_extract_predictions_logistic` *Extract Logistic Predictions at Sample Locations*

Description

Gets Java Maxent logistic scores at specific grid cell locations:

$$logistic = \frac{exp(H) \cdot raw}{1 + exp(H) \cdot raw}$$

Usage

```
maxent_extract_predictions_logistic(
  model,
  env_grids,
  feature_names,
  rows,
  cols
)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.
`rows`	Integer vector of row indices.
`cols`	Integer vector of column indices.

Value

Numeric vector of logistic scores in [0, 1]. NaN for NODATA cells.

See Also

`maxent_extract_predictions_raw`, `maxent_project_logistic`

`maxent_extract_predictions_raw` *Extract Raw Predictions at Sample Locations*

Description

Gets Java Maxent raw scores at specific grid cell locations:

$$raw = \frac{exp(lp - lpNorm)}{densityNorm}$$

Usage

```
maxent_extract_predictions_raw(model, env_grids, feature_names, rows, cols)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.
`rows`	Integer vector of row indices.
`cols`	Integer vector of column indices.

Details

This matches the raw output of Java Maxent and **dismo**.

Value

Numeric vector of Java raw scores. NaN for NODATA cells.

See Also

`maxent_project_raw`

`maxent_featured_space` *Create a FeaturedSpace Object*

Description

Constructs a MaxEnt FeaturedSpace from background point data, occurrence sample indices, and a list of pre-built Feature objects.

Usage

```
maxent_featured_space(
  num_points,
  sample_indices,
  features,
  bias_weights = NULL
)
```

Arguments

`num_points`	Integer: total number of background points.
`sample_indices`	Integer vector: 0-based indices (into the background array) of the occurrence sample locations.
`features`	List of external pointers to Feature objects, as returned by `maxent_linear_feature()`, `maxent_hinge_feature()`, `maxent_generate_features()`, etc.
`bias_weights`	Optional numeric vector of per-point bias weights (length `num_points`). When supplied, background density is computed as `bias[i] * exp(lp[i] - lpn)` instead of the standard `exp(lp[i] - lpn)`. This mirrors Java Maxent's `biasFile` parameter. Pass `NULL` (default) for uniform (unbiased) background.

Value

External pointer to a FeaturedSpace C++ object.

Examples

```
n   <- 100L
idx <- 90:99  # 0-based sample indices
env <- seq(0, 1, length.out = n)
f   <- maxent_linear_feature(env, "env1")
fs  <- maxent_featured_space(n, idx, list(f))
```

`maxent_featured_space_create` *Create a FeaturedSpace object*

Description

Constructs a MaxEnt FeaturedSpace from background point count, occurrence sample indices, and a list of Feature objects.

Usage

```
maxent_featured_space_create(
  num_points,
  sample_indices,
  feature_ptrs,
  bias_weights = numeric(0)
)
```

Arguments

`num_points`	Integer: number of background points.
`sample_indices`	Integer vector: 0-based indices of occurrence samples in the background array.
`feature_ptrs`	List of external pointers to Feature objects (from `create_linear_feature()` etc.).
`bias_weights`	Optional numeric vector of per-point bias weights (length `num_points`). When supplied, background density is computed as `bias[i] * exp(lp[i] - lpn)` instead of the standard `exp(lp[i] - lpn)`. Pass an empty vector (default) for uniform (unbiased) background.

Value

External pointer to a FeaturedSpace object.

`maxent_featured_space_from_callback` *Create a FeaturedSpace from a streaming background provider*

Description

Builds a `FeaturedSpace` by draining a callback-style background provider (typically backed by a `terra::SpatRaster` block loop) one tile at a time, then constructing features from the concatenated `(num_points x num_layers)` matrix via `generate_features`.

Usage

```
maxent_featured_space_from_callback(
  num_points,
  num_layers,
  layer_names,
  sample_indices,
  feature_types,
  n_thresholds,
  n_hinges,
  next_tile_fn,
  reset_fn,
  use_cache = TRUE
)
```

Arguments

num_points	Integer: total number of finite background points the provider will emit (sum of nrow() across tiles).
num_layers	Integer: number of environmental variables / raster layers per tile row.
layer_names	Character vector of length num_layers giving the name for each layer / column (used when generating feature names such as bio1^2, bio1*bio2).
sample_indices	Integer vector: 0-based indices of occurrence samples in the concatenated background stream.
feature_types	Character vector of feature types to generate. See maxent_generate_features.
n_thresholds	Number of threshold knots per variable.
n_hinges	Number of hinge knots per variable.
next_tile_fn	R function returning the next tile as a numeric matrix (nrow x num_layers), or a 0-row matrix / NULL when the stream is exhausted. The function is responsible for filtering out rows containing NAs.
reset_fn	R function (no arguments) that rewinds the underlying stream to the beginning.
use_cache	Logical; wrap the callback stream in CachingBackgroundProvider.

Details

This is the C++ entry point used by maxent_featured_space_from_rast; end users should prefer that R-level wrapper.

Value

External pointer to a FeaturedSpace object.

maxent_featured_space_from_rast *Create a FeaturedSpace directly from a terra SpatRaster*

Description

Streams a terra::SpatRaster block-by-block into the C++ FeaturedSpace streaming constructor, filtering NA cells on the fly and handing the concatenated (num_points x nlyr(rast)) matrix to maxent_generate_features so feature objects are built inside C++ without the R caller having to materialise the full raster stack first.

Usage

```
maxent_featured_space_from_rast(
  rast,
  sample_indices,
  feature_types = c("linear", "quadratic", "product", "threshold", "hinge"),
  n_thresholds = 10L,
  n_hinges = 10L,
  enable_streaming_eval = TRUE
)
```

Arguments

rast	A terra::SpatRaster with one layer per environmental variable. Layers must be named; names are passed through to maxent_generate_features so generated features carry the same identifiers as the layer names (e.g. bio1^2, bio1*bio2).
sample_indices	Integer vector: 0-based indices of occurrence samples in the concatenated stream of finite background cells. See maxent_raster_sample_indices for a helper that converts occurrence data.frames into these indices.
feature_types	Character vector of feature types to generate. Any subset of c("linear", "quadratic", "product", "threshold","hinge"). Defaults to all types.
n_thresholds	Integer; number of threshold knots per variable.
n_hinges	Integer; number of hinge knots per variable.
enable_streaming_eval	Logical; when TRUE, enables streaming-evaluation mode after object construction.

Details

This is the Phase E.2 entry point described in docs/ARCHITECTURE_terra_raster.md (sections 3.1 and 3.2). The result is bit-for-bit identical to the dense path (maxent_generate_features -> maxent_featured_space) when the same data, sample indices, and feature configuration are used.

Value

External pointer to a FeaturedSpace object.

See Also

maxent_featured_space, maxent_featured_space_from_callback, maxent_generate_features.

Examples

```
stack_path <- system.file("extdata", "stack_1_12_crop.rds",
                          package = "maxentcpp")
r <- terra::unwrap(readRDS(stack_path))
fs <- maxent_featured_space_from_rast(
    r,
    sample_indices = c(0L, 1L, 2L),
    feature_types  = c("linear", "quadratic")
)
```

---

maxent_featured_space_info

Get FeaturedSpace metadata

Description

Get FeaturedSpace metadata

Usage

```
maxent_featured_space_info(fs_ptr)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.

Value

Named list with num_points, num_samples, num_features, density_normalizer, linear_predictor_normalizer, has_bias.

maxent_feature_eval *Evaluate a Feature at a Given Index*

Description

Evaluates the feature function at the specified index (1-based, as is standard in R). Internally converts to 0-based indexing for C++.

Usage

```
maxent_feature_eval(feature, index)
```

Arguments

feature	External pointer to a Feature C++ object.
index	1-based integer index into the data vector.

Value

Numeric scalar: the feature value at that index.

Examples

```
vals <- c(0, 5, 10)
f <- maxent_linear_feature(vals, "temp")
maxent_feature_eval(f, 2)  # 5/10 = 0.5
```

maxent_feature_info *Get Feature Properties*

Description

Returns a list of metadata for a feature object.

Usage

```
maxent_feature_info(feature)
```

Arguments

feature	External pointer to a Feature C++ object.

Value

Named list with elements: name, type, lambda, min, max, size.

Examples

```
vals <- c(0, 5, 10)
f <- maxent_linear_feature(vals, "temp")
maxent_feature_info(f)
```

`maxent_fit` *Train a MaxEnt Model*

Description

Runs the sequential coordinate-ascent MaxEnt optimization on a FeaturedSpace.

Usage

```
maxent_fit(
  featured_space,
  max_iter = 500L,
  convergence = 1e-05,
  beta_multiplier = 1,
  min_deviation = 0.001
)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object (from `maxent_featured_space()`).
`max_iter`	Maximum number of training iterations (default 500).
`convergence`	Convergence threshold: stop when the per-20-iteration loss improvement is below this value (default 1e-5).
`beta_multiplier`	Regularization multiplier (default 1.0). Higher values increase regularization strength.
`min_deviation`	Minimum sample deviation floor used in regularization (default 0.001).

Value

Named list with:

loss Final regularized loss (scalar).

entropy Shannon entropy of the trained distribution.

iterations Number of training iterations completed.

converged Logical: whether the convergence threshold was reached.

lambdas Numeric vector of final lambda (weight) values.

Examples

```
n   <- 50L
idx <- c(5L, 15L, 25L, 35L, 45L)
env <- list(bio1 = runif(n), bio12 = runif(n))
feats <- maxent_generate_features(env, types = "linear")
fs  <- maxent_featured_space(n, idx, feats)
result <- maxent_fit(fs, max_iter = 100, convergence = 1e-5)
result$loss
```

---

maxent_generate_features

Generate Features from Environmental Variable Data

Description

Automatically generates all configured feature types from one or more environmental variable vectors.

Usage

```
maxent_generate_features(
  data,
  types = c("linear", "quadratic", "product", "threshold", "hinge"),
  n_thresholds = 10L,
  n_hinges = 10L
)
```

Arguments

`data`	Named list of numeric vectors, one per environmental variable.
`types`	Character vector of feature types to generate. Valid values: `"linear"`, `"quadratic"`, `"product"`, `"threshold"`, `"hinge"`. Defaults to all types.
`n_thresholds`	Integer; number of threshold knots per variable (default: 10).
`n_hinges`	Integer; number of hinge knots per variable (default: 10).

Value

Named list of external pointers to Feature C++ objects.

Examples

```
env_data <- list(
  temperature   = c(15, 20, 25, 18, 22),
  precipitation = c(100, 200, 150, 80, 300)
)
features <- maxent_generate_features(env_data, types = c("linear", "hinge"))
length(features)
```

maxent_get_entropy *Get entropy of a FeaturedSpace distribution*

Description

Get entropy of a FeaturedSpace distribution

Usage

```
maxent_get_entropy(fs_ptr)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.

Value

Shannon entropy (non-negative scalar).

maxent_get_loss *Get current loss of a FeaturedSpace*

Description

Get current loss of a FeaturedSpace

Usage

```
maxent_get_loss(fs_ptr)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.

Value

Scalar loss value (negative log-likelihood).

maxent_get_weights *Get current distribution weights from a FeaturedSpace*

Description

Get current distribution weights from a FeaturedSpace

Usage

```
maxent_get_weights(fs_ptr)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.

Value

Numeric vector of normalized weights (sums to 1).

maxent_grid *Create a Grid Object*

Description

Creates a raster grid for storing environmental variable data.

Usage

```
maxent_grid(dim, name = "", nodata_value = -9999)
```

Arguments

dim	GridDimension object defining spatial extent
name	Grid layer name
nodata_value	Value representing missing data (default: -9999)

Value

Grid object (external pointer)

Examples

```
dim <- maxent_dimension(100, 100, -120, 35, 0.1)
grid <- maxent_grid(dim, "temperature")
```

maxent_grid_from_matrix

Create a Grid from an R Matrix

Description

Builds a GridFloat from a numeric matrix and spatial parameters. NA values in the matrix are stored as the nodata value.

Usage

```
maxent_grid_from_matrix(mat, xll, yll, cellsize, nodata = -9999, name = "")
```

Arguments

mat	Numeric matrix.
xll	X coordinate of lower-left corner.
yll	Y coordinate of lower-left corner.
cellsize	Cell size.
nodata	NODATA sentinel value (default -9999).
name	Grid name (default "").

Value

External pointer to a GridFloat C++ object.

maxent_grid_from_terra

Convert a terra SpatRaster to a maxentcpp GridFloat

Description

Converts a single-layer terra::SpatRaster into a maxentcpp GridFloat external pointer. The raster must have exactly one layer and square cells (equal x and y resolution).

Usage

```
maxent_grid_from_terra(r, name = NULL)
```

Arguments

r	A single-layer terra::SpatRaster object.
name	Character: grid name (default: layer name from the raster).

Details

NA values in the raster are stored as NODATA (sentinel -9999).

Requires the **terra** package (listed in Suggests).

For the **raster** package, an equivalent workflow is:

```
library(raster)
r <- raster("bio1.tif")
mat <- as.matrix(r)
e <- extent(r)
g <- maxent_grid_from_matrix(mat, xll = e@xmin, yll = e@ymin,
                             cellsize = res(r)[1],
                             name = names(r))
```

Value

External pointer to a GridFloat C++ object.

Examples

```
stack_path <- system.file("extdata", "stack_1_12_crop.rds",
                          package = "maxentcpp")
r <- terra::unwrap(readRDS(stack_path))
g <- maxent_grid_from_terra(r[[1]])
maxent_grid_info(g)
```

---

maxent_grid_info *Get Grid Information*

Description

Returns metadata about a GridFloat object read from a file.

Usage

```
maxent_grid_info(grid)
```

Arguments

grid	External pointer to a GridFloat object.

Value

Named list with: nrows, ncols, xll, yll, cellsize, nodata, name, count_data.

maxent_grid_to_matrix *Convert Grid to R Matrix*

Description

Extracts the data from a GridFloat as an R numeric matrix. NODATA cells are converted to `NA`.

Usage

```
maxent_grid_to_matrix(grid)
```

Arguments

`grid`	External pointer to a GridFloat object.

Value

Numeric matrix (`nrows` × `ncols`).

maxent_grid_to_terra *Convert a maxentcpp GridFloat to a terra SpatRaster*

Description

Creates a `terra::SpatRaster` from a maxentcpp GridFloat external pointer.

Usage

```
maxent_grid_to_terra(grid, crs = "EPSG:4326")
```

Arguments

`grid`	External pointer to a GridFloat C++ object.
`crs`	Character: coordinate reference system string (default `"EPSG:4326"`, i.e. WGS 84 longitude/latitude).

Details

Requires the **terra** package (listed in Suggests).

For the **raster** package, an equivalent workflow is:

```
library(raster)
info <- maxent_grid_info(grid)
mat  <- maxent_grid_to_matrix(grid)
r <- raster(mat,
            xmn = info$xll,
            xmx = info$xll + info$ncols * info$cellsize,
            ymn = info$yll,
            ymx = info$yll + info$nrows * info$cellsize,
            crs = "+proj=longlat +datum=WGS84")
```

Value

A single-layer `terra::SpatRaster`.

Examples

```
stack_path <- system.file("extdata", "stack_1_12_crop.rds",
                          package = "maxentcpp")
r <- terra::unwrap(readRDS(stack_path))
g <- maxent_grid_from_terra(r[[1]])
r2 <- maxent_grid_to_terra(g)
```

---

maxent_hinge_feature *Create a Hinge Feature*

Description

Creates a piecewise-linear hinge feature.

Usage

```
maxent_hinge_feature(values, name, min_knot, max_knot, reverse = FALSE)
```

Arguments

`values`	Numeric vector of environmental variable values.
`name`	Character string: feature name/identifier.
`min_knot`	Lower knot of the hinge.
`max_knot`	Upper knot of the hinge (must be strictly greater than `min_knot`).
`reverse`	Logical; if `TRUE`, use reverse hinge. Default is `FALSE` (forward hinge).

Details

Forward hinge (`reverse = FALSE`): `eval(i) = max(0, (values[i] - min_knot) / (max_knot - min_knot))`

Reverse hinge (`reverse = TRUE`): `eval(i) = max(0, (max_knot - values[i]) / (max_knot - min_knot))`

Value

External pointer to a HingeFeature C++ object.

Examples

```
vals <- c(0, 5, 10, 3)
f <- maxent_hinge_feature(vals, "temperature_hinge", min_knot = 2, max_knot = 8)
maxent_feature_eval(f, 2)  # values[2] = 5 -> (5-2)/(8-2) = 0.5
```

---

maxent_layer *Create a Layer Metadata Object*

Description

Create a Layer Metadata Object

Usage

```
maxent_layer(name, type = "Continuous")
```

Arguments

name	Character: layer name.
type	Character: layer type. One of "Continuous", "Categorical", "Bias", "Mask", "Probability", "Cumulative", "DebiasAvg", or "Unknown".

Value

External pointer to a Layer C++ object.

---

maxent_layer_info *Get Layer Metadata*

Description

Get Layer Metadata

Usage

```
maxent_layer_info(layer)
```

Arguments

layer	External pointer to a Layer object.

Value

Named list with name and type.

maxent_layer_name *Extract Layer Name from a File Path*

Description

Strips directory prefix and extension.

Usage

```
maxent_layer_name(path)
```

Arguments

path	Character: file path.

Value

Character: layer name (e.g., "/data/bio1.asc" → "bio1").

maxent_linear_feature *Create a Linear Feature*

Description

Creates a linear (normalized) feature that evaluates to (values[i] - min) / (max - min). Returns 0 when min == max.

Usage

```
maxent_linear_feature(values, name, min_val = NULL, max_val = NULL)
```

Arguments

values	Numeric vector of environmental variable values.
name	Character string: feature name/identifier.
min_val	Minimum value for normalization. If NULL (default), computed as min(values).
max_val	Maximum value for normalization. If NULL (default), computed as max(values).

Value

External pointer to a LinearFeature C++ object.

Examples

```
vals <- c(0, 5, 10, 3)
f <- maxent_linear_feature(vals, "temperature")
maxent_feature_eval(f, 1)  # index 1 (R) -> 0-based index 0
```

maxent_load_lambdas *Load Model Lambdas from File*

Description

Reads model coefficients from a .lambdas file and applies them to a FeaturedSpace that was created with the same features.

Usage

```
maxent_load_lambdas(featured_space, file)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object (must have features with the same names as those in the file).
`file`	Character: path to the lambdas file.

Value

Invisibly returns the FeaturedSpace external pointer (updated in-place).

Examples

```
maxent_load_lambdas(fs, tempfile(fileext = ".lambdas"))
```

maxent_logloss *Compute Log-Loss*

Description

Computes the average cross-entropy log-loss from predictions at presence and absence sites.

Usage

```
maxent_logloss(presence, absence)
```

Arguments

`presence`	Numeric vector of prediction scores at presence sites.
`absence`	Numeric vector of prediction scores at absence sites.

Value

Average log-loss value (lower is better).

Examples

```
maxent_logloss(c(0.9, 0.8), c(0.1, 0.2))
```

`maxent_mess` *Compute MESS (Multivariate Environmental Similarity Surface)*

Description

Measures how similar each cell is to the training environment using the full distribution of reference values. Negative MESS values indicate novel (non-analog) environments.

Usage

```
maxent_mess(env_grids, reference_values, feature_names)
```

Arguments

`env_grids`	List of external pointers to Grid<float> objects.
`reference_values`	List of numeric vectors with reference values for each variable (e.g. values at training sites).
`feature_names`	Character vector of variable names.

Value

A named list with:

mess_grid External pointer to Grid<float> with MESS values

mod_grid External pointer to Grid<float> with Most Dissimilar Variable index (1-based)

Examples

```
result <- maxent_mess(list(g1, g2),
            list(temp_train_vals, precip_train_vals),
            c("temp", "precip"))
mess_mat <- maxent_grid_to_matrix(result$mess_grid)
mod_mat  <- maxent_grid_to_matrix(result$mod_grid)
```

maxent_mess_range *Compute MESS from Min/Max Ranges*

Description

Simplified MESS analysis using only the min/max of the reference data rather than the full distribution.

Usage

```
maxent_mess_range(env_grids, var_mins, var_maxs)
```

Arguments

env_grids	List of external pointers to Grid<float> objects.
var_mins	Numeric vector of minimum reference values per variable.
var_maxs	Numeric vector of maximum reference values per variable.

Value

A named list with mess_grid and mod_grid (external pointers).

maxent_misclassification *Compute Misclassification Rate*

Description

Fraction of misclassified samples at threshold 0.5.

Usage

```
maxent_misclassification(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

Misclassification rate in [0, 1].

Examples

```
maxent_misclassification(c(0.8, 0.9), c(0.1, 0.2))  # 0.0
```

maxent_model_entropy *Get Model Entropy*

Description

Returns the Shannon entropy of the current MaxEnt distribution.

Usage

```
maxent_model_entropy(featured_space)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object.

Value

Scalar: Shannon entropy (non-negative).

maxent_model_loss *Get Model Loss*

Description

Returns the current (negative log-likelihood) loss of the model.

Usage

```
maxent_model_loss(featured_space)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object.

Value

Scalar: current loss value.

maxent_model_weights *Get Model Distribution Weights*

Description

Returns the normalized probability weights of the current MaxEnt distribution over the background points.

Usage

```
maxent_model_weights(featured_space)
```

Arguments

featured_space	External pointer to a FeaturedSpace object.

Value

Numeric vector of weights that sum to 1.

maxent_percent_contribution *Compute Percent Contribution*

Description

Computes variable contribution based on sum of absolute lambda values for features derived from each variable. Results sum to 100

Usage

```
maxent_percent_contribution(model, feature_names)
```

Arguments

model	External pointer to a FeaturedSpace object.
feature_names	Character vector of base variable names.

Value

A data.frame with columns:

name Variable name

contribution Percent contribution

Examples

```
n   <- 50L
idx <- c(5L, 15L, 25L, 35L, 45L)
env <- list(temp = runif(n), precip = runif(n))
feats <- maxent_generate_features(env, types = "linear")
model <- maxent_featured_space(n, idx, feats)
maxent_fit(model, max_iter = 100, convergence = 1e-3)
contrib <- maxent_percent_contribution(model, c("temp", "precip"))
contrib  # data.frame with name and contribution
```

`maxent_permutation_importance` *Compute Permutation Importance*

Description

Measures how much each environmental variable contributes to model prediction quality by permuting each variable and measuring AUC drop. Results are normalised so that all importances sum to 100

Usage

```
maxent_permutation_importance(
  model,
  env_grids,
  feature_names,
  presence_rows,
  presence_cols,
  absence_rows,
  absence_cols,
  seed = 42L
)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names, matching the order of env_grids.
`presence_rows`	Integer vector of presence site row indices (0-based).
`presence_cols`	Integer vector of presence site column indices (0-based).
`absence_rows`	Integer vector of absence/background site row indices.
`absence_cols`	Integer vector of absence/background site column indices.
`seed`	Random seed for reproducibility (default 42).

Value

A data.frame with columns:

name Variable name

permutation_importance Normalised importance (%)

Examples

```
imp <- maxent_permutation_importance(model, list(g1, g2),
         c("temp", "precip"),
         pres_rows, pres_cols, abs_rows, abs_cols)
imp  # data.frame with name and permutation_importance
```

---

maxent_plot_response_curves

Write Response Curve PNGs for All Variables

Description

Generates response curve plots for each environmental variable and saves them as PNG files, replicating `density/ResponsePlot.java::makeplot()`. A full-size PNG and an optional thumbnail are written for every variable.

Usage

```
maxent_plot_response_curves(
  model,
  env_grids,
  feature_names,
  output_dir,
  species,
  var_indices = NULL,
  n_steps = 100L,
  thumbnail = TRUE,
  write_dat = FALSE
)
```

Arguments

`model`	External pointer to a trained FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names (one entry per element of `env_grids`).
`output_dir`	Character: directory under which a `plots/` sub-directory will be created.
`species`	Character: species name (used in file names).
`var_indices`	Integer vector of 0-based variable indices to plot. Defaults to all variables.

n_steps	Integer: number of steps in each curve (default 100).
thumbnail	Logical: also write a 210 × 140 pixel thumbnail PNG (default TRUE).
write_dat	Logical: also write a tab-delimited .dat file of the curve data (default FALSE).

Value

Invisibly returns a named list of file paths written.

Examples

```
maxent_plot_response_curves(
  model, list(g1, g2), c("bio1", "bio12"),
  output_dir = tempdir(), species = "Sp1")
```

---

maxent_plot_variable_importance

Write a Variable Importance Bar Chart PNG

Description

Creates a horizontal bar chart comparing percent contribution and permutation importance for each environmental variable, replicating density/Runner.java::makeJackknifePlots().

Usage

```
maxent_plot_variable_importance(
  contributions_df,
  perm_imp_df,
  species,
  output_dir
)
```

Arguments

contributions_df	A data.frame with columns name and contribution (from maxent_percent_contribution).
perm_imp_df	A data.frame with columns name and permutation_importance (from maxent_permutation_importan
species	Character: species name (used in the filename).
output_dir	Character: directory under which a plots/ sub-directory will be created.

Value

Invisibly returns the path to the PNG file.

Examples

```
maxent_plot_variable_importance(contrib, perm_imp,
  species = "Sp1", output_dir = tempdir())
```

`maxent_predict` *Predict with a trained FeaturedSpace model*

Description

Computes raw Gibbs distribution scores for new environmental data.

Usage

```
maxent_predict(fs_ptr, new_data)
```

Arguments

`fs_ptr`	External pointer to a trained FeaturedSpace object.
`new_data`	Numeric matrix with one row per new point and one column per feature (values must be pre-evaluated, i.e., the feature transformation already applied).

Value

Numeric vector of raw scores (unnormalized).

`maxent_predict_model` *Predict with a Trained MaxEnt Model*

Description

Computes raw Gibbs distribution scores for new environmental data, using the feature lambdas stored in the trained FeaturedSpace.

Usage

```
maxent_predict_model(featured_space, newdata)
```

Arguments

`featured_space`	External pointer to a trained FeaturedSpace object.
`newdata`	Numeric matrix: one row per new point, one column per feature. Column values must be the *already-evaluated* feature values (e.g., from running `maxent_feature_eval()` for each feature and each point).

Value

Numeric vector of raw (unnormalized) prediction scores.

Examples

```
# After training, predict on 5 new points with 2 features each
newdata <- matrix(runif(10), nrow = 5, ncol = 2)
preds <- maxent_predict_model(fs, newdata)
```

`maxent_print_results` *Print Maxent Model Results*

Description

Prints a summary of Maxent model performance metrics to the console, replicating the style of the **dismo** package's MaxEnt output. The report includes sample counts, training (and optionally test) evaluation statistics, and a ranked variable-contributions table.

Usage

```
maxent_print_results(
  species,
  eval_result,
  contributions_df,
  perm_imp_df,
  n_presence,
  n_background,
  test_eval_result = NULL,
  n_test = 0L,
  fit_result = NULL
)
```

Arguments

`species`	Character: species name.
`eval_result`	Named list returned by `maxent_evaluate` (training predictions vs background).
`contributions_df`	Data.frame with columns `name` and `contribution` (from `maxent_percent_contribution`).
`perm_imp_df`	Data.frame with columns `name` and `permutation_importance` (from `maxent_permutation_importanc`
`n_presence`	Integer: number of training presence records.
`n_background`	Integer: number of background records.
`test_eval_result`	Named list returned by `maxent_evaluate` for test data, or `NULL` (default).
`n_test`	Integer: number of test presence records (default `0L`).
`fit_result`	Named list returned by `maxent_fit` or `NULL`. Used to report regularized training gain and entropy.

Value

Invisibly returns a named list with all reported metrics.

Examples

```
maxent_print_results(
  species          = "Sp1",
  eval_result      = maxent_evaluate(pres_preds, bg_preds),
  contributions_df = contrib,
  perm_imp_df      = perm_imp,
  n_presence       = length(pres_rows),
  n_background     = length(bg_rows))
```

---

maxent_product_feature

Create a Product Feature

Description

Creates an interaction feature between two environmental variables: `eval(i)` = `norm(values1[i]) * norm(values2[i])`.

Usage

```
maxent_product_feature(
  values1,
  values2,
  name,
  min1 = NULL,
  max1 = NULL,
  min2 = NULL,
  max2 = NULL
)
```

Arguments

`values1`	Numeric vector for the first environmental variable.
`values2`	Numeric vector for the second environmental variable (must have the same length as `values1`).
`name`	Character string: feature name/identifier.
`min1`	Minimum of the first variable. If `NULL` (default), computed as `min(values1)`.
`max1`	Maximum of the first variable. If `NULL` (default), computed as `max(values1)`.
`min2`	Minimum of the second variable. If `NULL` (default), computed as `min(values2)`.
`max2`	Maximum of the second variable. If `NULL` (default), computed as `max(values2)`.

Value

External pointer to a ProductFeature C++ object.

Examples

```
temp <- c(0, 5, 10)
prec <- c(100, 200, 150)
f <- maxent_product_feature(temp, prec, "temp_x_prec")
maxent_feature_eval(f, 2)
```

`maxent_project_cloglog` *Project Model onto Grids (Cloglog Output)*

Description

Applies the Java Maxent cloglog transform to grid predictions:

$$cloglog = 1 - exp(-exp(H) \cdot raw)$$

Usage

```
maxent_project_cloglog(model, env_grids, feature_names)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.

Details

where H is the model entropy and raw is the normalised probability. This matches the cloglog output of Java Maxent and **dismo**, and is the recommended output for reporting.

Value

External pointer to a Grid<float> with cloglog scores in [0, 1].

See Also

`maxent_project_raw`, `maxent_project_logistic`

`maxent_project_logistic` *Project Model onto Grids (Logistic Output)*

Description

Applies the Java Maxent logistic transform to grid predictions:

$$logistic = \frac{exp(H) \cdot raw}{1 + exp(H) \cdot raw}$$

Usage

```
maxent_project_logistic(model, env_grids, feature_names)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.

Details

where H is the model entropy and raw is the normalised probability. This matches the logistic output of Java Maxent and **dismo**.

Value

External pointer to a Grid<float> with logistic scores in [0, 1].

See Also

`maxent_project_raw`, `maxent_project_cloglog`

`maxent_project_raw` *Project Model onto Grids (Raw Output)*

Description

Produces Java Maxent "raw" scores for every grid cell:

$$raw = \frac{exp(lp - lpNorm)}{densityNorm}$$

Usage

```
maxent_project_raw(model, env_grids, feature_names)
```

Arguments

model	External pointer to a FeaturedSpace object.
env_grids	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.

Details

This matches the raw output of the Java Maxent software and the **dismo** R package. See vignette("PREDICTION_SCALES") or docs/PREDICTION_SCALES.md for a full description of prediction scales.

Value

External pointer to a Grid<float> with Java raw scores.

See Also

maxent_project_cloglog, maxent_project_logistic

---

maxent_quadratic_feature

Create a Quadratic Feature

Description

Creates a quadratic feature that evaluates to linear_val^2 where linear_val = (values[i] - min) / (max - min).

Usage

```
maxent_quadratic_feature(values, name, min_val = NULL, max_val = NULL)
```

Arguments

values	Numeric vector of environmental variable values.
name	Character string: feature name/identifier.
min_val	Minimum value for normalization. If NULL (default), computed as min(values).
max_val	Maximum value for normalization. If NULL (default), computed as max(values).

Value

External pointer to a QuadraticFeature C++ object.

Examples

```
vals <- c(0, 5, 10)
f <- maxent_quadratic_feature(vals, "temperature_sq")
maxent_feature_eval(f, 2)  # evaluates at index 2 (1-based -> 0-based: 1)
```

maxent_raster_sample_indices *Convert occurrence coordinates to 0-based stream indices*

Description

Maps occurrence cell indices (as produced by `terra::cellFromXY`) into 0-based indices within the concatenated stream of finite (non-NA) background cells emitted by `maxent_featured_space_from_rast`.

Usage

```
maxent_raster_sample_indices(rast, cells)
```

Arguments

`rast`	A `terra::SpatRaster`.
`cells`	Integer vector of 1-based cell indices (as returned by `terra::cellFromXY`).

Value

Integer vector of 0-based indices within the finite-cell stream. Occurrence cells that fall on NA raster values (and therefore never appear in the stream) are silently dropped and a warning is emitted.

maxent_read_asc *Read an ESRI ASCII Grid File*

Description

Reads a `.asc` raster file into a GridFloat object.

Usage

```
maxent_read_asc(filename)
```

Arguments

`filename`	Character: path to the `.asc` file.

Value

External pointer to a GridFloat C++ object.

Examples

```
g <- maxent_read_asc("bio1.asc")
info <- maxent_grid_info(g)
print(info)
```

maxent_read_grid *Read a Grid File (Auto-Detect Format)*

Description

Reads a raster grid file. The format is detected from the file extension. Currently supports `.asc` (ESRI ASCII Grid).

Usage

```
maxent_read_grid(filename)
```

Arguments

`filename`	Character: path to the grid file.

Value

External pointer to a GridFloat C++ object.

maxent_read_lambdas *Read feature lambdas from a file*

Description

Restores model coefficients from a .lambdas file. The FeaturedSpace must have been created with the same features (same names and order).

Usage

```
maxent_read_lambdas(fs_ptr, filename)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`filename`	Character: path to the lambdas file.

Value

Called for side effects; returns invisibly.

maxent_read_occurrences *Read Species Occurrence Data*

Description

Reads species presence records from a CSV file or an R `data.frame` and converts them to row/column indices suitable for `maxent_featured_space`.

Usage

```
maxent_read_occurrences(
  file_or_df,
  dim,
  lon_col = "longitude",
  lat_col = "latitude",
  name_col = NULL
)
```

Arguments

`file_or_df`	Either a character path to a CSV file, or a `data.frame` already loaded in R (e.g. from `rgbif::occ_data()`, GBIF download, or `read.csv()`).
`dim`	A GridDimension object (from `maxent_dimension`) defining the study area grid.
`lon_col`	Character: name of the longitude column (default `"longitude"`).
`lat_col`	Character: name of the latitude column (default `"latitude"`).
`name_col`	Character or `NULL`: column for sample names. If `NULL` (default), sequential names are generated.

Value

A named list with:

samples List of Sample external pointers.

rows Integer vector of row indices (0-based).

cols Integer vector of column indices (0-based).

indices Integer vector of 0-based flat indices (`row * ncols + col`), suitable for `maxent_featured_space`.

Examples

```
dim <- maxent_dimension(100, 100, -120, 35, 0.1)
occ <- data.frame(longitude = c(-118.5, -119.0),
                  latitude  = c(36.5, 37.0))
result <- maxent_read_occurrences(occ, dim)
result$indices  # 0-based flat indices for FeaturedSpace
```

`maxent_response_curve` *Compute Marginal Response Curve*

Description

Generates a response curve by varying one environmental variable from its minimum to maximum value while holding all other variables at their mean. Applies the Java Maxent cloglog transform, matching the output of Java Maxent and **dismo**:

$$cloglog = 1 - exp(-exp(H) \cdot raw)$$

Usage

```
maxent_response_curve(
  model,
  env_grids,
  feature_names,
  var_index,
  n_steps = 100L
)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`env_grids`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.
`var_index`	0-based index of the variable to vary.
`n_steps`	Number of steps across the variable range (default 100).

Value

A data.frame with columns:

value Environmental variable value

prediction Cloglog-transformed prediction

See Also

`maxent_response_curve_fixed`

Examples

```
curve <- maxent_response_curve(model, list(g1, g2),
           c("temp", "precip"), var_index = 0)
plot(curve$value, curve$prediction, type = "l")
```

`maxent_response_curve_fixed` *Compute Response Curve with Fixed Values*

Description

Like `maxent_response_curve` but the user supplies explicit fixed values for the non-target variables. Applies the Java Maxent cloglog transform, matching the output of Java Maxent and **dismo**:

$$cloglog = 1 - exp(-exp(H) \cdot raw)$$

Usage

```
maxent_response_curve_fixed(
  model,
  fixed_values,
  feature_names,
  var_index,
  var_min,
  var_max,
  n_steps = 100L
)
```

Arguments

`model`	External pointer to a FeaturedSpace object.
`fixed_values`	Numeric vector of fixed values for each variable.
`feature_names`	Character vector of environment variable names.
`var_index`	0-based index of the variable to vary.
`var_min`	Minimum value of the target variable.
`var_max`	Maximum value of the target variable.
`n_steps`	Number of steps (default 100).

Value

A data.frame with columns: value, prediction.

See Also

`maxent_response_curve`

`maxent_run` *Run a Complete Maxent Species Distribution Modelling Workflow*

Description

Provides a single high-level entry point that mirrors running the Java Maxent GUI with one click. Starting from raw environmental grids and occurrence records, the function:

1. Reads and validates occurrence records.
2. Samples background points.
3. Extracts environmental values and builds features.
4. Trains the Maxent model.
5. Projects the model onto the full landscape (cloglog output).
6. Evaluates model performance (AUC).
7. Computes percent contribution and permutation importance.
8. Writes all standard output files (lambdas, prediction PNG, response curve PNGs, omission CSV, sample-predictions CSV, and `maxentResults.csv`).
9. Prints a performance summary to the console.

Usage

```
maxent_run(
  species,
  env_grids,
  occ_df,
  output_dir,
  lon_col = "long",
  lat_col = "lat",
  n_background = 10000L,
  types = c("linear", "quadratic", "hinge"),
  n_hinges = 15L,
  max_iter = 500L,
  seed = 42L,
  bias_weights = NULL,
  response_curves = TRUE,
  pictures = TRUE
)
```

Arguments

`species`	Character: species name (used in file names and the console report).
`env_grids`	Named list of external pointers to Grid<float> objects. The *names* of the list are used as the environmental variable names.
`occ_df`	A `data.frame` (or CSV path) with occurrence records.

output_dir	Character: directory where all output files are written (created if it does not exist).
lon_col	Character: longitude column name in occ_df (default "long").
lat_col	Character: latitude column name in occ_df (default "lat").
n_background	Integer: number of background points (default 10000).
types	Character vector of feature types passed to maxent_generate_features (default c("linear", "quadratic", "hinge")).
n_hinges	Integer: number of hinge knots (default 15).
max_iter	Integer: maximum training iterations (default 500).
seed	Integer: random seed for background sampling (default 42).
bias_weights	Optional numeric vector of per-background-point bias weights. Must have length equal to n_background (the background points only, not including occurrence points). When supplied, the background density is weighted by bias[i] * exp(lp[i] - lpn), mirroring Java Maxent's biasFile. Pass NULL (default) for uniform (unbiased) background.
response_curves	Logical: write response curve PNGs (default TRUE).
pictures	Logical: write prediction map PNG (default TRUE).

Value

A named list with:

model Trained FeaturedSpace external pointer.

fit_result List returned by maxent_fit.

evaluation List returned by maxent_evaluate.

contributions Data.frame of percent contributions.

permutation_importance Data.frame of permutation importances.

output_dir The output directory used.

Examples

```
stack_path      <- system.file("extdata", "stack_1_12_crop.rds",
                               package = "maxentcpp")
example_rasters <- terra::unwrap(readRDS(stack_path))
grids <- list(
  bio1  = maxent_grid_from_terra(example_rasters[[1]]),
  bio12 = maxent_grid_from_terra(example_rasters[[2]])
)
data(example_occ_df)

result <- maxent_run(
  species    = "Abeillia_abeillei",
  env_grids  = grids,
  occ_df     = example_occ_df,
  output_dir = tempdir(),
```

```
  lon_col    = "long",
  lat_col    = "lat")

result$evaluation$auc
```

`maxent_sample` *Create a MaxEnt Sample Object*

Description

Creates a sample point representing a species occurrence location.

Usage

```
maxent_sample(lon, lat, name = "", dim = NULL)
```

Arguments

`lon`	Longitude coordinate
`lat`	Latitude coordinate
`name`	Sample identifier/name
`dim`	Optional GridDimension object to calculate row/col indices

Value

Sample object (external pointer)

Examples

```
sample <- maxent_sample(lon = -118.5, lat = 36.5, name = "site1")
```

`maxent_save_lambdas` *Save Model Lambdas to File*

Description

Writes the trained model coefficients (lambdas) to a CSV file in the standard Maxent .lambdas format.

Usage

```
maxent_save_lambdas(featured_space, file)
```

Arguments

`featured_space`	External pointer to a trained FeaturedSpace object.
`file`	Character: path to the output file.

Value

Invisibly returns the output file path.

Examples

```
maxent_save_lambdas(fs, tempfile(fileext = ".lambdas"))
```

`maxent_sequential_fit` *Train a MaxEnt Model with the Java-Equivalent Sequential Optimizer*

Description

Runs the full `density.Sequential` optimizer ported from the original Java Maxent 3.4.4 on a FeaturedSpace. Unlike `maxent_fit()`, which uses the historical `goodAlpha`-only loop, this trainer reproduces the real Java optimizer: feature selection via `deltaLossBound`, Newton step with 1-D line search (`searchAlpha`), every-10-iteration `doParallelUpdate` with undo on loss-violating batch steps, and per-feature state tracking.

Usage

```
maxent_sequential_fit(
  featured_space,
  max_iter = 500L,
  convergence = 1e-05,
  beta_multiplier = 1,
  min_deviation = 0.001,
  parallel_update_frequency = 10L,
  disable_convergence_test = FALSE,
  trajectory_iterations = integer(0)
)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object (from `maxent_featured_space()`).
`max_iter`	Maximum number of iterations (default `500L`).
`convergence`	Convergence threshold on the per-20-iteration loss improvement (default `1e-5`). Ignored when `disable_convergence_test = TRUE`.
`beta_multiplier`	Regularization multiplier (default `1.0`).
`min_deviation`	Minimum sample-deviation floor used in regularization (default `0.001`).

`parallel_update_frequency`
: Iteration frequency at which `doParallelUpdate` runs (default `10L`, matching Java).

`disable_convergence_test`
: Logical: when `TRUE`, the loop runs a fixed `max_iter` iterations with no early stop. This is required for deterministic per-iteration trajectory comparisons against the Java oracle. Default `FALSE`.

`trajectory_iterations`
: Integer vector of 1-based iteration indices at which to capture (`loss`, `entropy`, `lambdas`) snapshots. May be empty. Snapshots outside `[1, max_iter]` are silently dropped.

Details

The lambda trajectory produced by this trainer matches the Java oracle's to 10^{-6} on $\|\Delta\lambda\|_\infty$ at every checkpoint, on both the symmetric and asymmetric fixtures in `maxentcppCompTest` — see `docs/ARCHITECTURE_xtensor_openmp.md` and the Phase C baseline report for details.

Value

Named list with:

loss Final regularized loss (scalar).

entropy Shannon entropy of the trained distribution.

iterations Number of training iterations completed.

converged Logical: whether the convergence threshold was reached (always `FALSE` when `disable_convergence_test = TRUE`).

lambdas Numeric vector of final lambda values.

trajectory A `data.frame` with one row per captured checkpoint and columns `iteration`, `loss`, `entropy`, `lambda_0`, `lambda_1`, ..., `lambda_{J-1}`.

See Also

`maxent_fit`, `maxent_featured_space`.

Examples

```
fs  <- maxent_featured_space(100L, 90:99, list(f))
res <- maxent_sequential_fit(
    fs,
    max_iter                = 500L,
    disable_convergence_test = TRUE,
    trajectory_iterations    = c(1L, 2L, 5L, 10L, 50L, 100L, 500L))
print(res$trajectory)
```

maxent_set_sample_expectations *Set sample expectations for a FeaturedSpace*

Description

Computes sample expectations and regularization deviations for each feature. Called automatically by `maxent_train()`, but exposed for advanced use.

Usage

```
maxent_set_sample_expectations(
  fs_ptr,
  beta_multiplier = 1,
  min_deviation = 0.001
)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`beta_multiplier`	Regularization multiplier (default 1.0).
`min_deviation`	Minimum sample deviation (default 0.001).

Value

Called for side effects; returns invisibly.

maxent_space_info *Get FeaturedSpace Information*

Description

Returns basic metadata about a FeaturedSpace object.

Usage

```
maxent_space_info(featured_space)
```

Arguments

`featured_space`	External pointer to a FeaturedSpace object.

Value

Named list with: `num_points`, `num_samples`, `num_features`, `density_normalizer`, `linear_predictor_normalizer`.

maxent_square_error *Compute Mean Squared Error*

Description

Presence sites contribute (1 - pred)^2, absence sites contribute pred^2.

Usage

```
maxent_square_error(presence, absence)
```

Arguments

presence	Numeric vector of prediction scores at presence sites.
absence	Numeric vector of prediction scores at absence sites.

Value

Mean squared error.

Examples

```
maxent_square_error(c(1.0, 1.0), c(0.0, 0.0))  # 0.0
```

maxent_threshold_feature *Create a Threshold Feature*

Description

Creates a binary step feature: `eval(i)` = 1 if `values[i]` > `threshold`, else `0`.

Usage

```
maxent_threshold_feature(values, name, threshold)
```

Arguments

values	Numeric vector of environmental variable values.
name	Character string: feature name/identifier.
threshold	The threshold value.

Value

External pointer to a ThresholdFeature C++ object.

Examples

```
vals <- c(1, 5, 10, 3)
f <- maxent_threshold_feature(vals, "temperature_threshold", threshold = 5)
maxent_feature_eval(f, 3)  # values[3] = 10 > 5 -> 1
```

`maxent_train` *Train a FeaturedSpace model*

Description

Runs the sequential coordinate-ascent MaxEnt optimization.

Usage

```
maxent_train(
  fs_ptr,
  max_iter = 500L,
  convergence = 1e-05,
  beta_multiplier = 1,
  min_deviation = 0.001
)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`max_iter`	Maximum number of training iterations (default 500).
`convergence`	Convergence threshold (default 1e-5).
`beta_multiplier`	Regularization multiplier (default 1.0).
`min_deviation`	Minimum sample deviation floor (default 0.001).

Value

Named list with elements: `loss`, `entropy`, `iterations`, `converged`, `lambdas`.

maxent_train_terra *Train MaxEnt directly from a terra SpatRaster*

Description

High-level convenience wrapper that builds a streaming FeaturedSpace from `rast`, maps occurrence locations to finite-stream sample indices, and trains with `maxent_fit`.

Usage

```
maxent_train_terra(
  rast,
  occurrences,
  lon_col = "longitude",
  lat_col = "latitude",
  feature_types = c("linear", "quadratic", "product", "threshold", "hinge"),
  n_thresholds = 10L,
  n_hinges = 10L,
  max_iter = 500L,
  convergence = 1e-05,
  beta_multiplier = 1,
  min_deviation = 0.001
)
```

Arguments

`rast`	A `terra::SpatRaster`.
`occurrences`	Either a two-column matrix/data.frame (x,y) or a vector of 1-based raster cell indices.
`lon_col`	Longitude column name when `occurrences` is a data.frame.
`lat_col`	Latitude column name when `occurrences` is a data.frame.
`feature_types`	Feature type set passed to feature generation.
`n_thresholds`	Number of threshold knots.
`n_hinges`	Number of hinge knots.
`max_iter`	Maximum training iterations.
`convergence`	Convergence threshold.
`beta_multiplier`	Regularization multiplier.
`min_deviation`	Minimum deviation floor.

Value

A named list with training results plus `model` and `sample_indices`.

maxent_variable_ranges *Compute Variable Ranges from Grids*

Description

Scans all valid cells in the grids to determine the [min, max] range of each environmental variable.

Usage

```
maxent_variable_ranges(env_grids)
```

Arguments

env_grids	List of external pointers to Grid<float> objects.

Value

A data.frame with columns: min, max (one row per variable).

maxent_write_asc *Write a Grid to ESRI ASCII Format*

Description

Writes a GridFloat to a .asc file.

Usage

```
maxent_write_asc(grid, filename, scientific = TRUE)
```

Arguments

grid	External pointer to a GridFloat object.
filename	Character: output file path.
scientific	Logical: use scientific notation for floating-point values (default TRUE).

Value

Invisibly returns the output file path.

Examples

```
maxent_write_asc(g, tempfile(fileext = ".asc"))
```

`maxent_write_lambdas` *Write feature lambdas to a file*

Description

Saves the trained model coefficients in CSV format compatible with the original Java Maxent .lambdas file format.

Usage

```
maxent_write_lambdas(fs_ptr, filename)
```

Arguments

`fs_ptr`	External pointer to a trained FeaturedSpace object.
`filename`	Character: path to the output file.

Value

Called for side effects; returns invisibly.

`maxent_write_omission_csv`
Write an Omission / Threshold CSV

Description

Computes cumulative predictions at all background + presence locations and derives nine standard threshold statistics, writing the result to `<output_dir>/<species>_omission.csv`. The output format mirrors `density/Runner.java::writeCumulativeIndex()`.

Usage

```
maxent_write_omission_csv(
  model,
  env_grids,
  feature_names,
  presence_rows,
  presence_cols,
  output_dir,
  species,
  test_rows = NULL,
  test_cols = NULL
)
```

Arguments

model	External pointer to a trained FeaturedSpace object.
env_grids	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
presence_rows	Integer vector of presence row indices (0-based).
presence_cols	Integer vector of presence column indices (0-based).
output_dir	Character: directory for output files.
species	Character: species name (used in the filename).
test_rows	Integer vector of test row indices (0-based) or NULL (default).
test_cols	Integer vector of test column indices (0-based) or NULL (default).

Value

Invisibly returns the path to the written CSV file.

Examples

```
maxent_write_omission_csv(model, list(g1, g2), c("bio1", "bio12"),
  pres_rows, pres_cols, output_dir = tempdir(), species = "Sp1")
```

---

maxent_write_prediction_png

Write a Maxent Prediction Grid as a PNG Image

Description

Converts a GridFloat prediction grid to a colour PNG image using the canonical Maxent colour ramp (red = high, blue = low). Optionally overlays presence and test-point locations, and renders a small legend.

Usage

```
maxent_write_prediction_png(
  grid,
  filename,
  presence_rows = NULL,
  presence_cols = NULL,
  test_rows = NULL,
  test_cols = NULL,
  mode = "plain",
  legend = TRUE,
  width = 800L,
  height = 600L
)
```

Arguments

grid	External pointer to a GridFloat prediction grid (e.g. from maxent_project_cloglog).
filename	Character: path for the output PNG file.
presence_rows	Integer vector of presence row indices (0-based) or NULL (default).
presence_cols	Integer vector of presence column indices (0-based) or NULL (default).
test_rows	Integer vector of test-set row indices (0-based) or NULL (default).
test_cols	Integer vector of test-set column indices (0-based) or NULL (default).
mode	Colour mode passed to maxent_color_ramp: one of "plain" (default), "log", "blackandwhite", or "redandyellow".
legend	Logical: draw a colour-bar legend (default TRUE).
width	Integer: PNG width in pixels (default 800).
height	Integer: PNG height in pixels (default 600).

Value

Invisibly returns filename.

Examples

```
pred <- maxent_project_cloglog(model, list(g1, g2), c("bio1", "bio12"))
maxent_write_prediction_png(pred, tempfile(fileext = ".png"))
```

---

maxent_write_sample_predictions
Write Sample Predictions CSV

Description

Computes model predictions at presence (and optionally test) locations and writes <output_dir>/<species>_samplePredic

Usage

```
maxent_write_sample_predictions(
  model,
  env_grids,
  feature_names,
  presence_rows,
  presence_cols,
  output_dir,
  species,
  test_rows = NULL,
  test_cols = NULL
)
```

Arguments

model	External pointer to a trained FeaturedSpace object.
env_grids	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.
presence_rows	Integer vector of presence row indices (0-based).
presence_cols	Integer vector of presence column indices (0-based).
output_dir	Character: directory for the output file.
species	Character: species name (used in the filename).
test_rows	Integer vector of test row indices (0-based) or NULL (default).
test_cols	Integer vector of test column indices (0-based) or NULL (default).

Value

Invisibly returns the path to the written CSV.

Examples

```
maxent_write_sample_predictions(model, list(g1, g2), c("bio1", "bio12"),
  pres_rows, pres_cols, output_dir = tempdir(), species = "Sp1")
```

---

print.maxent_sample *Print Sample Information*

Description

Print Sample Information

Usage

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

Arguments

x	Sample object
...	Additional arguments (ignored)

Value

Invisibly returns x.

project_cloglog *Project Model onto Grids (Java-compatible cloglog output)*

Description

Applies the Java Maxent cloglog formula: cloglog_java = 1 - exp(-exp(H) * raw_java)

Usage

```
project_cloglog(fs_ptr, grid_ptrs, feature_names)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.
grid_ptrs	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.

Details

where H is the model entropy and raw_java is the normalised probability. This matches the cloglog output of Java Maxent and dismo.

Value

External pointer to a Grid<float> with Java cloglog scores in [0, 1].

project_logistic *Project Model onto Grids (Java-compatible logistic output)*

Description

Applies the Java Maxent logistic formula: logistic_java = (exp(H) * raw_java) / (1 + exp(H) * raw_java)

Usage

```
project_logistic(fs_ptr, grid_ptrs, feature_names)
```

Arguments

fs_ptr	External pointer to a FeaturedSpace object.
grid_ptrs	List of external pointers to Grid<float> objects.
feature_names	Character vector of environment variable names.

Details

where H is the model entropy and raw_java is the normalised probability. This matches the logistic output of Java Maxent and dismo.

Value

External pointer to a Grid<float> with Java logistic scores in [0, 1].

`project_raw` *Project Model onto Grids (Java-compatible raw output)*

Description

Applies a trained FeaturedSpace model to produce Java Maxent "raw" scores: raw_java = exp(lp - lpNorm) / densityNorm

Usage

```
project_raw(fs_ptr, grid_ptrs, feature_names)
```

Arguments

`fs_ptr`	External pointer to a FeaturedSpace object.
`grid_ptrs`	List of external pointers to Grid<float> objects.
`feature_names`	Character vector of environment variable names.

Details

This matches the raw output of the Java Maxent software and the dismo R package.

Value

External pointer to a Grid<float> with Java raw scores.

sample_get_feature *Get feature value from a sample*

Description

Get feature value from a sample

Usage

```
sample_get_feature(sample_ptr, feature_name, default_val = 0)
```

Arguments

sample_ptr	External pointer to Sample
feature_name	Name of the feature
default_val	Default value if feature not found

Value

Feature value

sample_set_feature *Set feature value on a sample*

Description

Set feature value on a sample

Usage

```
sample_set_feature(sample_ptr, feature_name, value)
```

Arguments

sample_ptr	External pointer to Sample
feature_name	Name of the feature
value	Feature value

Value

Called for side effects; returns invisibly.

Index