# Package: EcoNiche (via r-universe)

October 30, 2026

**Type** Package

**Title** Community Niche Position and Width Estimation Tools

**Version** 1.0.3

**Description** Provides methods for estimating species niche position and niche breadth under continuous environmental gradients. The package implements canonical correspondence analysis (CCA), partial CCA (pCCA), generalized additive models (GAM), and Levins' niche breadth metrics for species-level and community-level analyses. Methods are based on ter Braak (1986) <doi:10.2307/1938672>, Okie et al. (2015) <doi:10.1098/rspb.2014.2630>, Feng et al. (2020) <doi:10.1111/mec.15441>, Wood (2017) <doi:10.1201/9781315370279>, and Levins (1968, ISBN:978-0691080628).

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** ggplot2, mgcv, vegan,

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, viridisLite

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**RoxygenNote** 7.3.3

**URL** https://github.com/yedeng-lab/EcoNiche

**BugReports** https://github.com/yedeng-lab/EcoNiche/issues

**Repository** https://yedeng-lab.r-universe.dev

**Date/Publication** 2026-08-01 07:44:57 UTC

**RemoteUrl** https://github.com/yedeng-lab/econiche

**RemoteRef** HEAD

**RemoteSha** 46632fa1e2a1ed28efe6b0af9fbbe3ae2b5bb9ff

# Contents

cca_calc_gradient . . . 2
cca_calc_group . . . 3
cca_calc_species . . . 4
cca_fit_ordination . . . 5
cca_plot_group_env . . . 6
cca_prep_env . . . 7
cca_workflow . . . 8
cca_workflow_gradient . . . 9
cca_workflow_group . . . 10
gam_calc_sitewidth . . . 11
gam_fit_model . . . 12
gam_plot_species . . . 13
levins_calc_binned . . . 15
levins_calc_group . . . 16
levins_plot_pos_width . . . 17
niche_width_calc . . . 18
plant_env . . . 20
plant_group . . . 20
plant_otu . . . 21

**Index** **22**

---

`cca_calc_gradient` *Plot Site Niche Position vs Environmental Gradient (Step 4)*

---

## Description

Evaluates the relationship between the environmental gradient and the aggregated niche position of sites.

## Usage

```
cca_calc_gradient(env, site_pos, var, make_plot = TRUE, galaxy_colnum = TRUE)
```

## Arguments

| | |
|---|---|
| `env` | A sample-by-environment data.frame (rows = samples). |
| `site_pos` | Named numeric vector of site scores on the position axis. |
| `var` | Environmental variable to plot against (name or index). |
| `make_plot` | Logical; if `TRUE`, returns a ggplot object. |
| `galaxy_colnum` | Logical; for numeric indices, whether to treat them as 1-based indices. |

## Value

A list containing:

**data** Data frame used for plotting (ENV, NichePosition).

**plot** The ggplot object.

---

`cca_calc_group` *Calculate Niche Width Summary by Group (Step 5)*

---

## Description

Computes aggregated niche width statistics at the sample level and group level. Supports two calculation modes via `choice`: "all" (global context) or group-specific context.

## Usage

```
cca_calc_group(otu, site_width, group, choice = "all")
```

## Arguments

| | |
|---|---|
| `otu` | OTU/species matrix (rows = taxa, columns = samples). |
| `site_width` | Site scores for width axis (usually matrix). |
| `group` | A named vector or factor defining groups. Names must match sample IDs. |
| `choice` | Calculation mode. If `"all"`, computes niche width using all samples. Otherwise, computes within-group niche widths. |

## Value

A list containing:

**sample_summary** Data frame of sample-level statistics (ID, AverageValue, StandardDeviation/Error).

**group_summary** Data frame of group-level statistics.

`cca_calc_species` *Calculate Species Niche Width and Position (Step 3)*

## Description

Computes the niche width and niche position for each species using the site scores from Step 2. Generates a plot of Niche Width vs. Niche Position.

## Usage

```
cca_calc_species(
  otu,
  site_width,
  site_pos,
  method = c("lm", "loess"),
  make_plot = TRUE,
  top_node = 10000
)
```

## Arguments

| | |
|---|---|
| `otu` | OTU/species matrix (rows = taxa, columns = samples). |
| `site_width` | Site scores for width axis (usually matrix from partial CCA). |
| `site_pos` | Site scores for position axis (usually vector from CCA axis 1). |
| `method` | Smoothing method for the plot ("lm" or "loess"). |
| `make_plot` | Logical; if `TRUE`, returns a ggplot object. |
| `top_node` | Integer; maximum number of most abundant species to use for loess smoothing calculation (default 10000). Points are plotted for all valid species. |

## Value

A list containing:

**species** Data frame of species traits (NichePosition, NicheWidth).

**plot** The ggplot object (if `make_plot = TRUE`).

`cca_fit_ordination` *Fit CCA and Partial-CCA Ordination (Step 2)*

## Description

Performs Constrained Correspondence Analysis (CCA) and Partial CCA to obtain site scores for niche position and niche width calculations.

## Usage

```
cca_fit_ordination(
  otu,
  env,
  sel,
  covariates,
  standardize = TRUE,
  galaxy_colnum = TRUE
)
```

## Arguments

| | |
|---|---|
| `otu` | An OTU/species matrix or data frame (rows = taxa, columns = samples). |
| `env` | A data.frame of environmental variables (rows = samples). |
| `sel` | Variables (names or indices) defining the width axis (used in partial CCA). |
| `covariates` | Variables (names or indices) defining the position axis (used in CCA and as covariates in partial CCA). |
| `standardize` | Logical; if `TRUE`, environmental variables are standardized. Default is `TRUE`. |
| `galaxy_colnum` | Logical; for numeric indices, whether to treat them as Galaxy-style 1-based indices (skipping column 1). |

## Value

A list containing:

**site_width** Matrix of site scores from the partial CCA (width axes).

**site_pos** Named vector of site scores from the first axis of the CCA (position axis).

**site_pos_mat** Matrix of all site scores from the CCA.

**cca** The CCA model object.

**pcca** The Partial CCA model object.

`cca_plot_group_env` *Plot Sample/Group Niche Width vs Environmental Gradient (Step 6)*

## Description

Visualizes the relationship between environmental gradients and niche width at the sample or group level.

## Usage

```
cca_plot_group_env(
  env,
  sample_summary,
  var,
  group = NULL,
  plot_type = c("sample", "group", "both"),
  method = c("lm", "loess"),
  make_plot = TRUE,
  galaxy_colnum = TRUE,
  show_ci = TRUE,
  annotate_stats = TRUE
)
```

## Arguments

| | |
|---|---|
| `env` | A sample-by-environment data.frame. |
| `sample_summary` | Output from `cca_calc_group` (step 5). |
| `var` | Environmental variable to plot against (name or index). |
| `group` | Named vector/factor of group labels (required for group plots). |
| `plot_type` | Type of plot: "sample", "group", or "both". |
| `method` | Smoothing method ("lm" or "loess"). |
| `make_plot` | Logical; if `TRUE`, returns ggplot object(s). |
| `galaxy_colnum` | Logical; for numeric indices, whether to treat them as 1-based indices. |
| `show_ci` | Logical; whether to show confidence intervals on the smooth line. |
| `annotate_stats` | Logical; whether to annotate R-squared and P-value on the plot. |

## Value

A list containing data frames and ggplot objects for samples and/or groups.

cca_prep_env *Prepare Environmental Data for CCA (Step 1)*

## Description

Performs standardization and Principal Component Analysis (PCA) on environmental variables. This step assists in selecting constrained variables and covariates for downstream CCA or partial-CCA.

## Usage

```
cca_prep_env(
  env,
  sel,
  constrain = NULL,
  standardize = TRUE,
  galaxy_colnum = TRUE
)
```

## Arguments

| | |
|---|---|
| env | A data.frame of environmental variables. Rows must be sample IDs and columns must be environmental variables. |
| sel | A vector of variable names or column indices to be used as constrained variables. |
| constrain | Optional. A vector of variable names or column indices to be used as covariates. (Note: in the legacy workflow this parameter corresponds to 'covariates'). |
| standardize | Logical; if TRUE, environmental variables are standardized (centered and scaled) column-wise. Default is TRUE. |
| galaxy_colnum | Logical; if TRUE and numeric indices are provided, they are treated as 1-based indices where column 1 is assumed to be SampleID (and thus indices are decremented by 1). Default is TRUE. |

## Value

A list containing:

**env_sd** The standardized environmental data frame.

**sel_idx** Indices of selected variables.

**con_idx** Indices of constrained covariates.

**pca** The PCA result object (from prcomp).

**combined** A data frame combining SampleID, PCA axes, and constrained variables.

**cor_res** A matrix/table of correlations between PCA axes and original variables, including significance levels.

cca_workflow *CCA-Based Niche Analysis Workflow Controller*

## Description

A controller function that orchestrates the CCA-based niche analysis workflow. It dispatches the execution to either the gradient workflow (Steps 1-4) or the group workflow (Steps 1-2, 5-6) based on the specified mode.

## Usage

```
cca_workflow(mode = c("gradient", "group"), ...)
```

## Arguments

| | |
|---|---|
| mode | A character string specifying the analysis mode. Must be one of "gradient" or "group". |
| ... | Additional arguments passed to the specific workflow functions (cca_workflow_gradient or cca_workflow_group). |

## Value

A list containing the results of the executed workflow steps. See cca_workflow_gradient or cca_workflow_group for structure details.

## Examples

```
data("plant_otu")
data("plant_env")
otu <- plant_otu[rowSums(plant_otu > 0) >= 15, ]
res <- cca_workflow(
  mode = "gradient",
  otu = otu,
  env = plant_env,
  sel = c("AMP", "Moisture", "AP", "pH"),
  covariates = "AMT",
  var = "AMT",
  method = "loess",
  galaxy_colnum = FALSE,
  make_plot = FALSE,
  top_node = 20
)
str(res, max.level = 1)
```

`cca_workflow_gradient` *Execute Gradient-Based Workflow (Steps 1-4)*

## Description

Runs the complete environmental gradient analysis pipeline: 1. Prepare env data (PCA) 2. Fit CCA/Partial-CCA 3. Calculate species niche traits 4. Analyze gradient vs niche position

## Usage

```
cca_workflow_gradient(
  otu,
  env,
  sel,
  covariates,
  standardize = TRUE,
  method = c("lm", "loess"),
  var = sel[1],
  galaxy_colnum = TRUE,
  make_plot = TRUE,
  top_node = 10000
)
```

## Arguments

| | |
|---|---|
| `otu` | An OTU/species matrix or data frame (rows = taxa, columns = samples). |
| `env` | A data.frame of environmental variables. Rows must be sample IDs and columns must be environmental variables. |
| `sel` | A vector of variable names or column indices to be used as constrained variables. |
| `covariates` | Variables (names or indices) defining the position axis (used in CCA and as covariates in partial CCA). |
| `standardize` | Logical; if `TRUE`, environmental variables are standardized (centered and scaled) column-wise. Default is `TRUE`. |
| `method` | Smoothing method for the plot ("lm" or "loess"). |
| `var` | Environmental variable for gradient plotting (passed to Step 4). |
| `galaxy_colnum` | Logical; if `TRUE` and numeric indices are provided, they are treated as 1-based indices where column 1 is assumed to be SampleID (and thus indices are decremented by 1). Default is `TRUE`. |
| `make_plot` | Logical; if `TRUE`, returns a ggplot object. |
| `top_node` | Integer; maximum number of most abundant species to use for loess smoothing calculation (default 10000). Points are plotted for all valid species. |

## Value

A named list containing outputs of steps 1 through 4.

`cca_workflow_group` *Execute Group-Based Workflow (Steps 1-2, 5-6)*

## Description

Runs the complete group-based analysis pipeline: 1. Prepare env data (PCA) 2. Fit CCA/Partial-CCA 5. Calculate group niche widths 6. Plot group/sample niche width vs gradient

## Usage

```
cca_workflow_group(
  otu,
  env,
  sel,
  group,
  covariates,
  standardize = TRUE,
  method = c("lm", "loess"),
  var = sel[1],
  choice = "all",
  galaxy_colnum = TRUE,
  make_plot = TRUE,
  plot_type = c("sample", "group", "both"),
  show_ci = TRUE,
  annotate_stats = TRUE
)
```

## Arguments

| | |
|---|---|
| `otu` | An OTU/species matrix or data frame (rows = taxa, columns = samples). |
| `env` | A data.frame of environmental variables. Rows must be sample IDs and columns must be environmental variables. |
| `sel` | A vector of variable names or column indices to be used as constrained variables. |
| `group` | A named vector or factor defining groups. Names must match sample IDs. |
| `covariates` | Variables (names or indices) defining the position axis (used in CCA and as covariates in partial CCA). |
| `standardize` | Logical; if `TRUE`, environmental variables are standardized (centered and scaled) column-wise. Default is `TRUE`. |
| `method` | Smoothing method ("lm" or "loess"). |
| `var` | Environmental variable for gradient plotting (passed to Step 6). |
| `choice` | Calculation mode. If `"all"`, computes niche width using all samples. Otherwise, computes within-group niche widths. |
| `galaxy_colnum` | Logical; if `TRUE` and numeric indices are provided, they are treated as 1-based indices where column 1 is assumed to be SampleID (and thus indices are decremented by 1). Default is `TRUE`. |

| | |
|---|---|
| make_plot | Logical; if TRUE, returns ggplot object(s). |
| plot_type | Type of plot: "sample", "group", or "both". |
| show_ci | Logical; whether to show confidence intervals on the smooth line. |
| annotate_stats | Logical; whether to annotate R-squared and P-value on the plot. |

## Value

A named list containing outputs of steps 1, 2, 5, and 6.

---

# gam_calc_sitewidth *Calculate Sample-Level Niche Width Weighted by Abundance*

---

## Description

Computes the abundance-weighted mean niche breadth for each sample, given an OTU-by-sample table and OTU-level niche breadth estimates (e.g., from gam_fit_model). The formula is:

$$B_j = \sum_i (abundance_{ij} \times breadth50_i) / \sum_i abundance_{ij}$$

## Usage

```
gam_calc_sitewidth(
  otu,
  niche_df,
  otu_col = "OTU",
  width_col = "breadth50",
  weight_mode = c("auto", "counts", "relative")
)
```

## Arguments

| | |
|---|---|
| otu | An OTU-by-sample matrix or data frame. |
| niche_df | A data frame containing at least columns for OTU IDs and niche width values. |
| otu_col | Character string; the column name in niche_df containing OTU IDs. Default is "OTU". |
| width_col | Character string; the column name in niche_df containing niche width values. Default is "breadth50". |
| weight_mode | Character string; how to handle abundance weights. One of "auto", "counts", or "relative". |

## Value

A data frame with columns sample and Bw50_abundance_weighted.

gam_fit_model *Fit GAM-Based Niche Curves for All OTUs*

## Description

Fits a Generalized Additive Model (GAM) response curve for each OTU along a single environmental gradient. This function handles both count data (using Negative Binomial or Poisson families) and relative abundance data (using logit-transformed Gaussian models). It estimates the niche optimum and the 50% niche breadth (breadth50) for each taxon.

## Usage

```
gam_fit_model(
  otu,
  env,
  env_var,
  data_type = c("auto", "count", "relative"),
  count_family = c("nb", "poisson"),
  use_offset = TRUE,
  lib_size = NULL,
  min_prev = 0.1,
  min_total = 100,
  min_mean = 1e-05,
  k_spline = 5,
  n_grid = 200,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| otu | An OTU-by-sample matrix or data frame. Rows should be OTUs/taxa and columns should be samples. Values can be counts or relative abundances. |
| env | A sample-by-environment data frame. Rows must be samples and columns must be environmental variables. |
| env_var | A character string specifying the column name in env to use as the environmental gradient. |
| data_type | Character string specifying the data type. One of "auto" (heuristic detection), "count", or "relative". |
| count_family | Character string specifying the error distribution family for count data. One of "nb" (Negative Binomial, default) or "poisson". |
| use_offset | Logical; whether to use an offset term (log library size) in the GAM for count data. Default is TRUE. |
| lib_size | Optional named numeric vector of library sizes (sequencing depth) for each sample. If NULL and data_type="count", column sums of otu are used. |

| | |
|---|---|
| min_prev | Minimum prevalence threshold (proportion of samples with abundance > 0). OTUs below this threshold are skipped. Default is 0.10. |
| min_total | Minimum total abundance threshold. Relevant for count data. Default is 100. |
| min_mean | Minimum mean abundance threshold. Relevant for relative abundance data. Default is 1e-5. |
| k_spline | Integer; the upper limit on the spline basis dimension for the smooth term s(env, k = ...). Default is 5. |
| n_grid | Integer; number of grid points along the gradient used to estimate optimum and niche breadth. Default is 200. |
| verbose | Logical; whether to print progress messages. Default is TRUE. |

## Value

A data frame with one row per OTU and the following columns:

**OTU** OTU identifier.

**n_nonzero** Number of samples with non-zero abundance.

**prevalence** Proportion of samples with non-zero abundance.

**total_counts** Total counts (for count data) or NA.

**r2** Adjusted R-squared (for relative abundance models) or NA.

**dev_expl** Proportion of deviance explained.

**edf** Effective degrees of freedom of the smooth term.

**F_stat** Test statistic (F or Chi-sq) for the smooth term.

**p_value** Approximate p-value for the smooth term.

**optimum_env** Environmental value at the peak of the fitted curve.

**env50_min** Lower environmental bound where fitted abundance >= 50% of peak.

**env50_max** Upper environmental bound where fitted abundance >= 50% of peak.

**breadth50** Niche breadth (env50_max - env50_min).

---

# gam_plot_species *Plot GAM-Based Niche Curve for a Single OTU*

---

## Description

Fits a GAM response curve for a specific OTU and plots the results. Includes options for confidence intervals and different color palettes.

## Usage

```
gam_plot_species(
  otu,
  env,
  env_var,
  otu_id,
  data_type = c("auto", "count", "relative"),
  count_family = c("nb", "poisson"),
  use_offset = TRUE,
  lib_size = NULL,
  min_mean = 1e-05,
  min_prev = 0.1,
  k_spline = 5,
  n_grid = 200,
  add_ci = TRUE,
  palette = c("blue", "orange", "green", "purple", "viridis"),
  point_alpha = 0.85
)
```

## Arguments

| | |
|---|---|
| otu | An OTU-by-sample matrix or data frame. |
| env | A sample-by-environment data frame. |
| env_var | Character string; the environmental variable to use as the gradient. |
| otu_id | Character string; the ID of the OTU to plot (must exist in otu). |
| data_type | Character string; "auto", "count", or "relative". |
| count_family | Character string; "nb" or "poisson". |
| use_offset | Logical; whether to use library size offset for count data. |
| lib_size | Optional named vector of library sizes. |
| min_mean | Minimum mean abundance filter (for relative mode). |
| min_prev | Minimum prevalence filter. |
| k_spline | Spline basis dimension. |
| n_grid | Number of grid points for prediction. |
| add_ci | Logical; whether to plot the 95% confidence interval. |
| palette | Character string; color palette name ("blue", "orange", "green", "purple", or "viridis"). |
| point_alpha | Numeric; transparency alpha for observed data points. |

## Value

A list containing:

**data** Data frame of observed values (env, y).

**grid** Data frame of fitted values along the gradient.

**fit** The GAM model object.

**optimum_env** Estimated niche optimum.

**env50_min** Lower bound of 50% niche breadth.

**env50_max** Upper bound of 50% niche breadth.

**breadth50** Niche breadth.

**plot** The ggplot object.

---

`levins_calc_binned` *Calculate Levins Niche Width Along a Gradient (Binned States)*

---

## Description

Computes OTU-level Levins niche breadth along a continuous composite axis (e.g., CCA1) by discretizing the axis into bins (states). Abundance is first aggregated within bins (mean or sum), converted to within-bin proportions $p_{ij}$, and then used to compute:

$$B_i = 1/\sum_j p_{ij}^2$$

A standardized breadth is also reported:

$$B_i' = (B_i - 1)/(K - 1)$$

where $K$ is the number of bins.

## Usage

```
levins_calc_binned(
  otu,
  env,
  axis_var,
  nbin = 8L,
  bin_method = c("equal_freq", "equal_width"),
  agg_fun = c("mean", "sum"),
  otu_mode = c("auto", "count", "relative"),
  min_occ = 3L,
  min_abund = 5
)
```

## Arguments

| | |
|---|---|
| `otu` | An OTU-by-sample matrix or data frame. |
| `env` | A sample-by-environment data frame with row names as sample IDs. |
| `axis_var` | Character string; column name in `env` containing the axis values. |
| `nbin` | Integer; number of bins used to discretize the axis. Default is 8. |

| | |
|---|---|
| bin_method | Character string; binning strategy. "equal_freq" (quantile-based) or "equal_width". |
| agg_fun | Character string; aggregation method within bins. "mean" (recommended) or "sum". |
| otu_mode | Character string; input data type. "auto" (detect), "count", or "relative". If "count", columns are normalized to relative abundance before aggregation to reduce sequencing depth bias. |
| min_occ | Minimum number of samples with abundance > 0 required to keep an OTU. |
| min_abund | Minimum total abundance required to keep an OTU. |

## Value

A data frame with one row per OTU and the following columns:

**OTU** OTU identifier.

**axis** Name of the gradient axis used.

**n_states** Number of bins (K).

**levins_B** Raw Levins niche breadth.

**levins_Bstd** Standardized Levins niche breadth.

**n_samples** Number of samples where the OTU is present.

**total_abund** Total abundance of the OTU.

---

# levins_calc_group *Calculate Community-Mean Levins Niche Width Along a Gradient*

---

## Description

Computes the abundance-weighted mean Levins width for each sample, using OTU-level Levins widths and sample-level OTU abundances. Optionally plots the relationship between this community metric and an environmental gradient.

## Usage

```
levins_calc_group(
  otu,
  env,
  levins_df,
  grad,
  width_col = "levins_Bstd",
  method = c("lm", "loess"),
  make_plot = TRUE
)
```

### Arguments

| | |
|---|---|
| otu | An OTU-by-sample matrix or data frame. |
| env | A sample-by-environment data frame. |
| levins_df | A data frame of OTU-level Levins results containing at least columns OTU and width_col. |
| grad | The environmental gradient to plot against (column name or index in env). |
| width_col | Character string; column name in levins_df to use as the width metric. Default is "levins_Bstd". |
| method | Character string; smoothing method for the trend line. "lm" or "loess". |
| make_plot | Logical; whether to return a ggplot object. |

### Details

The community-mean width for sample $s$ is:

$$B_s = \sum_i Q_{is} B'_i$$

where $Q_{is}$ is the relative abundance of OTU $i$ in sample $s$ and $B'_i$ is the standardized Levins width (levins_Bstd).

### Value

A list containing:

**data** Data frame with columns Sample, ENV, and CommLevinsWidth.

**plot** The ggplot object (if make_plot=TRUE).

---

## levins_plot_pos_width *Plot Species Niche Position vs Levins Width*

---

### Description

Merges species niche optima (from any method such as CCA, GAM, or weighted average) with standardized Levins niche width, and plots the position-width relationship showing the correlation (R-squared) and P-value.

### Usage

```
levins_plot_pos_width(
  pos_df,
  levins_df,
  id_col,
  pos_col = "NichePosition",
  width_col = "levins_Bstd",
  method = c("lm", "loess"),
  make_plot = TRUE
)
```

### Arguments

| | |
|---|---|
| pos_df | A data frame containing species niche positions. Must include id_col and pos_col. |
| levins_df | A data frame containing Levins widths. Must include id_col and width_col. |
| id_col | Character string; column name of the species/OTU ID shared by both tables (e.g., "OTU"). |
| pos_col | Character string; column name of the niche position in pos_df. Default "NichePosition". |
| width_col | Character string; column name of the Levins width in levins_df. Default "levins_Bstd". |
| method | Character string; fitting method for the trend line. "lm" or "loess". |
| make_plot | Logical; whether to return a ggplot object. |

### Value

A list containing:

**data** Merged data frame used for plotting.

**plot** The ggplot object (if make_plot=TRUE).

---

## niche_width_calc *Calculate Niche Width (Samples as States)*

---

### Description

Computes OTU-level niche breadth by treating each sample as a discrete state. For OTU $i$, let $p_{is}$ be the proportional abundance of OTU $i$ in sample $s$ (i.e., abundance of OTU $i$ normalized to sum to 1 across all samples).

### Usage

```
niche_width_calc(
  otu,
  env = NULL,
  min_occ = 3L,
  min_abund = 5,
  standardize = TRUE,
  method = c("levins", "shannon", "both")
)
```

## Arguments

| | |
|---|---|
| otu | An OTU-by-sample matrix or data frame (rows = OTUs, columns = samples). Values should be non-negative (counts or relative abundance). |
| env | Optional sample metadata data frame with row names as sample IDs. If provided, samples are aligned by the intersection of colnames(otu) and rownames(env). |
| min_occ | Minimum number of samples with abundance > 0 required to keep an OTU. Default is 3. |
| min_abund | Minimum total abundance required to keep an OTU (sum across samples). Default is 5. |
| standardize | Logical; if TRUE and method includes "levins", returns the standardized breadth levins_Bstd in addition to the raw Levins breadth. Default is TRUE. |
| method | Character; which niche width index to compute. One of "levins", "shannon", or "both". Default is "levins". |

## Details

Two niche-width indices are supported via method:

- **Levins breadth (Levins)**:

$$B_i = 1/\sum_s p_{is}^2$$

  Optionally, a standardized breadth ranging from 0 to 1 is returned:

$$B_i' = (B_i - 1)/(K - 1)$$

  where $K$ is the total number of samples (states).

- **Shannon breadth (Shannon)**:

$$H_i = -\sum_s p_{is} \log(p_{is})$$

  Terms with $p_{is} = 0$ are excluded from the sum to avoid $\log(0)$.

## Value

A data frame with one row per OTU. Columns include:

**OTU** OTU identifier.

**n_states** Number of states (samples) used in calculation.

**n_samples** Number of samples where the OTU is present.

**total_abund** Total abundance of the OTU across samples.

**levins_B** Raw Levins niche breadth. Present if method is "levins" or "both".

**levins_Bstd** Standardized Levins niche breadth (if standardize=TRUE). Present if method is "levins" or "both".

**shannon_H** Shannon niche breadth (entropy). Present if method is "shannon" or "both".

## plant_env *Example plant environmental table*

### Description

Environmental variables for the samples in plant_otu. Rows are samples and columns are environmental variables.

### Usage

```
plant_env
```

### Format

A data frame with 290 rows and 11 variables:

**TN** Total nitrogen.

**TK** Total potassium.

**TP** Total phosphorus.

**SOC** Soil organic carbon.

**AP** Available phosphorus.

**AN** Available nitrogen.

**Moisture** Soil moisture.

**AMT** Annual mean temperature.

**AMP** Annual mean precipitation.

**pH** Soil pH.

**abspH** Absolute deviation from neutral pH.

### See Also

plant_otu, plant_group

## plant_group *Example plant sample grouping*

### Description

Group labels for the samples in plant_otu and plant_env. The vector names are sample IDs.

### Usage

```
plant_group
```

## Format

A named character vector with one group label per sample.

## See Also

plant_otu, plant_env

---

plant_otu *Example plant community abundance table*

---

## Description

A forest plant community dataset for demonstrating the EcoNiche workflow. Rows are plant species and columns are samples. Values are non-negative abundance counts.

## Usage

```
plant_otu
```

## Format

A numeric matrix with 1038 plant species in rows and 290 samples in columns.

## See Also

plant_env, plant_group

# Index

∗ **datasets**
plant_env, 20
plant_group, 20
plant_otu, 21

cca_calc_gradient, 2
cca_calc_group, 3, *6*
cca_calc_species, 4
cca_fit_ordination, 5
cca_plot_group_env, 6
cca_prep_env, 7
cca_workflow, 8
cca_workflow_gradient, *8*, 9
cca_workflow_group, *8*, 10

gam_calc_sitewidth, 11
gam_fit_model, 12
gam_plot_species, 13

levins_calc_binned, 15
levins_calc_group, 16
levins_plot_pos_width, 17

niche_width_calc, 18

plant_env, 20, *20, 21*
plant_group, 20, *20, 21*
plant_otu, *20*, 21, *21*
prcomp, *7*