

Package ‘ccva’

October 22, 2026

Type Package

Title Climate Change Vulnerability Assessment with Constrained Latent Models

Version 0.1.0

Description A general framework for climate change vulnerability assessment (CCVA) of any species and any dataset. Variables are tagged as exposure, sensitivity or adaptive capacity; each component is computed from its own predictors by a chosen method; and the three components are combined into vulnerability. Methods range from the conventional weighted index of Ofori et al. (2017) <[doi:10.1038/s41598-017-13245-y](https://doi.org/10.1038/s41598-017-13245-y)> to shape-constrained additive models ('scam'), latent-variable structural equation models ('lavaan') in which vulnerability is inferred from observable proxies, and penalised regression ('glmnet') for high-dimensional predictors. Ecological knowledge enters as explicit constraints on predictors, on each component's role and on the links between components. Diagnostics, validation against observed outcomes, bootstrap and weight-based uncertainty, calibrated risk predictions and plots for every step are included.

License GPL (>= 3)

URL <https://github.com/DoostiH/ccva>

BugReports <https://github.com/DoostiH/ccva/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), lavaan (>= 0.6-12), stats, utils

Suggests scam, glmnet, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Hassan Doosti [aut, cre, cph] (affiliation: Macquarie University)

Maintainer Hassan Doosti <hassan.doosti@mq.edu.au>

Repository CRAN

Date/Publication 2026-10-22 16:00:02 UTC

Contents

as_ccva	2
ccva	4
ccva_adjust_method	5
ccva_aggregate	6
ccva_calibrate	7
ccva_check	8
ccva_classify	9
ccva_combine	10
ccva_compare	12
ccva_component	13
ccva_constraints	15
ccva_methods	17
ccva_plot	17
ccva_rank	18
ccva_recommend	19
ccva_register	20
ccva_report	21
ccva_sdm_exposure	22
ccva_sim	23
ccva_simulate	24
ccva_sim_data	25
ccva_thermal_margin	26
ccva_uncertainty	27
ccva_validate	28
predict.ccva_fit	29
theme_ccva	29
Index	31

as_ccva	<i>Build a CCVA dataset</i>
---------	-----------------------------

Description

Tags the columns of any data frame as indicators of the three CCVA components, gives each indicator a direction, and names the observable outcomes (vulnerability proxies such as IUCN status, population trend or range change). Every other function in the package works on the resulting ‘ccva_data‘ object, so this is the package’s data contract: nothing about a particular taxon or study is assumed.

Usage

```
as_ccva(
  data,
  exposure,
  sensitivity,
  adaptive = NULL,
  outcomes = NULL,
  id = NULL,
  direction = NULL,
  coords = NULL,
  formative = NULL,
  na_action = c("omit", "fail")
)
```

Arguments

<code>data</code>	A data frame with one row per assessed unit (species or population).
<code>exposure, sensitivity, adaptive</code>	Character vectors naming the indicator columns of each component. Each component may use one column or many, and the sets may differ in size. ‘adaptive’ may be ‘NULL’ for an exposure-sensitivity (ES) assessment.
<code>outcomes</code>	Optional character vector naming observable vulnerability proxies. Needed by the latent, shape-constrained and penalised models, and by <code>[ccva_validate()]</code> .
<code>id</code>	Optional name of the column holding unit identifiers. If ‘NULL’, rows are labelled ‘unit_1’, ‘unit_2’, ...
<code>direction</code>	Optional named vector of ‘+1’ / ‘-1’ (or “+” / “-”) giving, for each indicator, whether a higher value means <i>more</i> of its component (more exposure, more sensitivity, more adaptive capacity), and for each outcome whether a higher value means <i>more vulnerable</i> . Columns not listed default to ‘+1’. For example a thermal tolerance ratio, where higher means less sensitive, takes ‘-1’; range change, where a negative value is range loss, takes ‘-1’.
<code>coords</code>	Optional character vector of two column names (longitude, latitude) used by <code>[ccva_plot()]</code> for maps.
<code>formative</code>	Optional components whose indicators are complementary <i>facets</i> rather than interchangeable measures of one construct – e.g. adaptive capacity from dispersal ability and genetic diversity, which act through distinct mechanisms. Their indicators are not expected to covary, so <code>[ccva_check()]</code> does not flag a low Cronbach’s alpha, and the latent model enters them as a composite (built by the component scoring method) instead of a reflective latent factor. Components not listed are reflective: their indicators are treated as interchangeable measures.
<code>na_action</code>	“omit” (default) drops rows with missing values in any used column, with a message; “fail” stops instead.

Value

An object of class ‘ccva_data’.

See Also

[ccva_simulate()] for data with a known latent structure.

Examples

```
d <- as_ccva(ccva_sim,
  exposure   = c("E1_risk", "E2_risk", "E3_risk"),
  sensitivity = grep("^s[0-9]+_", names(ccva_sim), value = TRUE),
  adaptive    = grep("^a[0-9]+_", names(ccva_sim), value = TRUE),
  outcomes    = c("iucn", "pop_trend", "range_change"),
  direction   = c(pop_trend = -1, range_change = -1),
  id = "species")
d
```

ccva

Run a complete CCVA

Description

Convenience wrapper: builds the ‘ccva_data’ (if given a data frame) and calls [ccva_combine()].

Usage

```
ccva(data, method = "latent", ...)
```

Arguments

data	A ‘ccva_data’ object, or a data frame together with the component arguments of [as_ccva()].
method	Combination method, see [ccva_combine()].
...	Arguments for [as_ccva()] (when ‘data’ is a data frame) and [ccva_combine()].

Value

A ‘ccva_fit’ object.

Examples

```
fit <- ccva(ccva_sim_data(), method = "weighted",
  component_method = "ordinal")
head(ccva_rank(fit))
```

ccva_adjust_method *Adjust trait measurements for assay methodology*

Description

Physiological sensitivity traits such as heat tolerance (CTmax) are measured with different protocols – static or dynamic assays, different ramping rates, starting temperatures and acclimation regimes – and the protocol shifts the value obtained. Before species are compared, the methodological effect should be removed. This fits a linear model of the measurements on the method variables (with a unit effect when units are measured more than once) and returns every measurement adjusted to a common reference protocol, plus a per-unit summary ready to use as an indicator in [as_ccva()].

Usage

```
ccva_adjust_method(
  data,
  value,
  method_vars,
  unit = NULL,
  reference = NULL,
  weights = NULL
)
```

Arguments

data	A data frame of measurements (one row per measurement).
value	Name of the measured trait column.
method_vars	Names of the method columns (numeric, e.g. ramp rate; or categorical, e.g. static / dynamic).
unit	Optional name of the unit column (species or population). Including it lets method effects be estimated within units.
reference	Optional named list of reference values for the method variables. Defaults: the median of numeric variables, the most common level of categorical ones.
weights	Optional name of a column of weights (e.g. sample size).

Details

Method effects can only be separated from true differences between units when protocols vary within units, or are not confounded with them across units; aliased effects are reported.

Value

An object of class ‘ccva_adjustment’ with ‘data’ (including the adjusted column ‘<value>_adj’), ‘by_unit’ (mean adjusted value per unit), ‘effects’ (method coefficients), ‘reference’ and ‘model’.

Examples

```
set.seed(1)
d <- data.frame(species = rep(paste0("sp", 1:20), each = 3),
                ramp = rep(c(0.1, 0.25, 0.5), 20),
                assay = rep(c("dynamic", "dynamic", "static"), 20))
d$ctmax <- 40 + rep(rnorm(20), each = 3) + 4 * d$ramp + rnorm(60, 0, 0.3)
adj <- ccva_adjust_method(d, "ctmax", c("ramp", "assay"), unit = "species")
adj
head(adj$by_unit)
```

ccva_aggregate

Aggregate species vulnerability to sites or communities

Description

For spatial prioritisation: given which species occur where, summarise the vulnerability of each site's community – mean and maximum vulnerability, the number and share of highly vulnerable species, and the mean exposure, sensitivity and adaptive capacity of its species. With a [ccva_uncertainty()] result, the site means get intervals too.

Usage

```
ccva_aggregate(
  fit,
  occurrence,
  site = "site",
  unit = "species",
  uncertainty = NULL,
  coords = NULL,
  level = 0.95
)
```

Arguments

fit	A 'ccva_fit'.
occurrence	Either a data frame with one row per species presence at a site (columns named by 'site' and 'unit'), or a sites x species 0/1 matrix with dimnames.
site, unit	Column names in 'occurrence'.
uncertainty	Optional [ccva_uncertainty()] result.
coords	Optional data frame with columns 'site', 'x', 'y' for maps.
level	Interval coverage.

Value

An object of class 'ccva_sites': a data frame per site, ordered from most to least vulnerable community.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted")
set.seed(1)
occ <- data.frame(site = rep(paste0("site", 1:8), each = 25),
                  species = sample(names(fit$score), 200, replace = TRUE))
occ <- unique(occ)
ccva_aggregate(fit, occ)
```

ccva_calibrate	<i>Calibrated risk predictions</i>
----------------	------------------------------------

Description

Rankings say which species are *more* vulnerable than others; managers need to know how *likely* a species is to decline. This calibrates a fit's vulnerability scores against an observed outcome (for example IUCN status of Vulnerable or worse, a declining population trend, or range loss) by logistic regression, and returns for every unit the probability of that outcome with a confidence interval.

Usage

```
ccva_calibrate(
  fit,
  outcome = NULL,
  threshold = NULL,
  event = NULL,
  uncertainty = NULL,
  R = 500,
  level = 0.95,
  seed = NULL
)
```

Arguments

fit	A 'ccva_fit'.
outcome	Outcome column (default: the first outcome). Oriented so higher = more vulnerable.
threshold	Units with an oriented outcome at or above this value count as an event. Default: for an ordinal outcome (10 or fewer distinct values) the second-highest value, e.g. IUCN status Vulnerable or worse on a 0–3 scale; otherwise the median.
event	Optional logical vector of events, overriding 'outcome'.
uncertainty	Optional [ccva_uncertainty()] result.
R	Number of bootstrap replicates.
level	Interval coverage.
seed	Optional random seed.

Details

Intervals come from a bootstrap over units that refits the calibration model. If a `[ccva_uncertainty()]` result is supplied, each replicate also uses a different draw of the vulnerability scores, so uncertainty in the scores themselves is propagated into the risk.

When the outcome was also used to fit the vulnerability model (latent, shape and penalised methods use the outcomes), this calibration is in-sample and optimistic. For an honest assessment, calibrate against an outcome the model did not see.

Value

An object of class `'ccva_calibration'` with `'units'` (probability and interval per unit), `'curve'` (probability across the score range), `'bins'` (observed vs predicted), `'metrics'` (AUC, Brier score) and the `'model'`.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted",
                   component_method = "ordinal")
cal <- ccva_calibrate(fit, outcome = "iucn", R = 200, seed = 1)
cal
```

ccva_check

Diagnose the indicators before combining them

Description

Summing indicators into a component assumes they measure one coherent construct; combining components assumes they are not redundant. Neither assumption is usually checked. This function checks both:

Usage

```
ccva_check(x, redundancy = 0.9, component_method = "mean")
```

Arguments

<code>x</code>	A <code>'ccva_data'</code> object.
<code>redundancy</code>	Absolute correlation above which a pair is flagged.
<code>component_method</code>	Method used to score components for the cross-component correlations.

Details

* **internal consistency** of each component – Cronbach’s alpha (raw and standardised), the mean inter-item correlation (which, unlike alpha, does not rise with the number of indicators) and corrected item-total correlations of its oriented indicators. Alpha is only expected to be high for *reflective* components, whose indicators are interchangeable measures; for components declared ‘formative’ in [as_ccva()] (distinct, complementary facets) a low alpha is expected and is not flagged; * **cross-component correlation** of the component scores (Spearman); * **redundancy** – indicator pairs, within or across components, with absolute Spearman correlation above ‘redundancy’; * **dimensionality** – predictors per unit, flagging blocks where predictors approach or exceed the number of units (use a penalised method there).

Value

An object of class ‘ccva_check’.

Examples

```
chk <- ccva_check(ccva_sim_data())
chk
```

ccva_classify

Classify vulnerability scores into Low / Moderate / High

Description

Conventional CCVAs cut scores at 1/3 and 2/3, which silently assumes the scores are uniformly distributed and has no link to any observed outcome. This function offers that convention and three alternatives.

Usage

```
ccva_classify(
  score,
  cutoffs = "uniform",
  data = NULL,
  roc_outcome = NULL,
  roc_levels = NULL,
  labels = c("Low", "Moderate", "High")
)
```

Arguments

score	Numeric vector of vulnerability scores.
cutoffs	One of * "uniform" – fixed cuts at 1/3 and 2/3 (Ofori et al. 2017); * "quantile" – sample tertiles (equal-sized classes); * "mixture" – natural clusters from a three-component Gaussian mixture fitted by EM; * "roc" – outcome-calibrated cuts maximising Youden’s J against an ordinal outcome (see ‘roc_outcome’, ‘roc_levels’); * a numeric vector of two cut points.

data	A 'ccva_data' object; needed for "roc".
roc_outcome	Outcome column used for "roc" (default: the first outcome). It is oriented so higher = more vulnerable.
roc_levels	Two outcome levels on the oriented scale: units at or above the first are "at least Moderate", at or above the second are "High". Defaults to the second and third distinct values.
labels	Class labels.

Value

An ordered factor with attribute 'cutoffs'.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted")
table(ccva_classify(fit$score, "uniform"))
table(ccva_classify(fit$score, "roc", data = fit$data))
```

ccva_combine	<i>Combine the components into vulnerability</i>
--------------	--

Description

Scores the three components and combines them into a vulnerability score for every unit, under a chosen model. Constraints from [ccva_constraints()] are imposed on predictors and on each component's role in every model that can express them.

Usage

```
ccva_combine(
  x,
  method = c("weighted", "shape", "latent", "penalised"),
  component_method = "mean",
  component_args = list(),
  weights = NULL,
  constraints = NULL,
  response = NULL,
  shared_error = NULL,
  max_indicators = 30,
  k = NULL,
  alpha = 1,
  cutoffs = "uniform",
  ...
)
```

Arguments

x	A 'ccva_data' object.
method	Combination method; see Methods.
component_method	Scoring method for the components: one string for all, or a named list / vector per component (see [ccva_component()]).
component_args	Optional named list (per component) of extra arguments for [ccva_component()], e.g. 'weights' or 'breaks'.
weights	Component weights for "weighted", named 'exposure', 'sensitivity', 'adaptive'. Default equal.
constraints	A [ccva_constraints()] object. Defaults to [ccva_default_constraints()]; use [ccva_unconstrained()] for free estimation.
response	For "shape": an outcome column, a numeric vector (higher = more vulnerable), or 'NULL' to use the first principal component of all outcomes.
shared_error	For "latent": a list of character vectors, each a group of indicators with a common error source.
max_indicators	For "latent": a component with more indicators than this (or with only one) enters as its observed component score instead of a latent factor. Default 30.
k	Basis dimension for "shape" splines; chosen from 'n' if 'NULL'.
alpha	Elastic-net mixing for "penalised" (1 = LASSO).
cutoffs	Classification of scores into Low / Moderate / High; see [ccva_classify()].
...	Passed to registered combine methods.

Value

An object of class 'ccva_fit': 'score' (vulnerability in $\setminus[0, 1]$, higher = more vulnerable), 'class', 'components' (fitted [ccva_component()] objects), 'component_scores', 'model', 'details', 'notes' (any constraint a method could not express) and 'predict'.

Methods

"weighted" The conventional index (Ofori et al. 2017) with the corrected normalisation $V = (w_E E + w_S S - w_A A + w_A) / (w_E + w_S + w_A)$, which equals $(E + S - A + 1) / 3$ for equal weights and maps the full range onto $\setminus[0, 1]$ without clipping. Without adaptive-capacity indicators it is $(E + S) / 2$. With [constrain_moderator()] adaptive capacity has no main effect: $V \propto w_E E(1 - \gamma_E A) + w_S S(1 - \gamma_S A)$. Use 'component_method = "ordinal"' to reproduce Ofori's 1–3 scoring.

"shape" Shape-constrained additive model (Pya and Wood 2015, package **scam**) of an observed 'response' on the component scores. Component signs become monotone increasing / decreasing splines, so the *shape* is learned from data while the *direction* is fixed by ecology. With [constrain_moderator()] it fits the interaction surfaces $f_{EA}(E, A) + f_{SA}(S, A)$, monotone in each argument (adaptive capacity as moderator only).

“latent” Structural equation model (**lavaan**) in which vulnerability V^* is a latent variable measured by the outcome proxies and caused by latent exposure, sensitivity and adaptive capacity, each measured by its own indicators (a full MIMIC model). Every constraint type is imposed as a linear (in)equality on loadings and paths. Indicators sharing an error source (e.g. SDM metrics with a common denominator) are declared in ‘shared_error’ and modelled with an orthogonal method factor.

“penalised” Two-stage model for high-dimensional predictors: (1) latent vulnerability is estimated from the outcome proxies alone; (2) it is regressed on **all** indicators with a LASSO / elastic net (**glmnet**) whose coefficient signs follow the component and indicator constraints. Selects which predictors drive vulnerability.

Methods added with [ccva_register()] are also accepted.

References

Ofori, B. Y. et al. (2017) **Scientific Reports** 7, 12979. Pya, N. and Wood, S. N. (2015) **Statistics and Computing** 25, 543–559. Rosseel, Y. (2012) **Journal of Statistical Software** 48(2).

See Also

[ccva()], [ccva_compare()], [ccva_uncertainty()], [ccva_plot()]

Examples

```
d <- ccva_sim_data()

# Conventional Ofori-style weighted index
w <- ccva_combine(d, "weighted", component_method = "ordinal")
w

# Constrained latent model (the default constraints)
l <- ccva_combine(d, "latent",
                 shared_error = list(c("E1_risk", "E2_risk", "E3_risk")))
l
cor(l$score, d$truth$V_star, method = "spearman")
```

ccva_compare

Compare several CCVA fits

Description

Compare several CCVA fits

Usage

```
ccva_compare(..., truth = NULL)
```

Arguments

... Named 'ccva_fit' objects (or a single named list of them), all fitted to the same units.

truth Optional true vulnerability; defaults to the simulated truth when present.

Value

An object of class 'ccva_comparison' with a 'table' (validation correlations per method, class agreement with the first fit), 'rank_cor' (Spearman correlations between methods' scores) and the fits.

Examples

```
d <- ccva_sim_data()
cmp <- ccva_compare(
  ofori = ccva_combine(d, "weighted", component_method = "ordinal"),
  latent = ccva_combine(d, "latent",
    shared_error = list(c("E1_risk", "E2_risk", "E3_risk"))))
cmp
```

ccva_component

Score the exposure, sensitivity and adaptive-capacity components

Description

Each component is computed from its own indicators (its predictors) by a chosen method. Indicators are first oriented with the 'direction' given in [as_ccva()], so that a higher value always means more of the component.

Usage

```
ccva_component(
  x,
  component = c("exposure", "sensitivity", "adaptive"),
  method = "mean",
  weights = NULL,
  breaks = NULL,
  anchor = NULL,
  alpha = 1,
  constraints = NULL,
  ...
)

ccva_exposure(x, method = "mean", ...)

ccva_sensitivity(x, method = "mean", ...)

ccva_adaptive(x, method = "mean", ...)
```

Arguments

<code>x</code>	A 'ccva_data' object.
<code>component</code>	One of "exposure", "sensitivity", "adaptive".
<code>method</code>	Scoring method; see Details.
<code>weights</code>	Optional named numeric vector of indicator weights (for "ordinal": maximum scores, e.g. Ofori's 11, 10, 9, ... for sensitivity). Missing indicators get weight 1.
<code>breaks</code>	For "ordinal", an optional named list giving, per indicator, two cut points on the raw scale. Values above the upper cut score 3 (more of the component) when the direction is '+1', and 1 when it is '-1'.
<code>anchor</code>	For "penalised": an outcome column name, a numeric vector (higher = more vulnerable), or 'NULL' to use the first principal component of all outcomes.
<code>alpha</code>	Elastic-net mixing for "penalised": 1 = LASSO, 0 = ridge.
<code>constraints</code>	A [ccva_constraints()] object. Predictor-level rules are honoured: fixed-at-zero indicators are dropped; signs and bounds restrict the penalised coefficients.
<code>...</code>	Passed to registered methods.

Details

Methods:

"ordinal" The Ofori et al. (2017) scheme: each indicator is scored 1–3 (by 'breaks', or by sample tertiles when 'breaks' is 'NULL'), and the scores are summed with optional 'weights' (the maximum score of each indicator) and divided by the maximum. Scores therefore lie in $\backslash[1/3, 1]$, exactly as in the original study. Use it to reproduce conventional assessments and to measure what discretisation loses.

"mean" Continuous: each indicator rescaled to $\backslash[0, 1]$, then a weighted mean. No information is discarded.

"pca" First principal component of the standardised indicators, signed so loadings are positive on average, rescaled to $\backslash[0, 1]$. Data-driven weights.

"penalised" Penalised regression (LASSO by default; set 'alpha') of an 'anchor' on the indicators, with coefficient signs constrained to the ecological direction. Designed for high-dimensional blocks where predictors outnumber units, such as genomic markers. Needs the **glmnet** package.

Methods added with [ccva_register()] are also accepted.

Value

An object of class 'ccva_component' with elements 'score' (named numeric vector), 'importance' (per-indicator weight, loading or coefficient), 'method', 'component' and 'predict' (a function scoring new data on the same scale).

See Also

[ccva_combine()], which calls these for you.

Examples

```
d <- ccva_sim_data()
e_ord <- ccva_exposure(d, method = "ordinal")
e_cont <- ccva_exposure(d, method = "mean")
cor(e_ord$score, e_cont$score, method = "spearman")

s <- ccva_sensitivity(d, method = "pca")
head(sort(s$importance, decreasing = TRUE))
```

ccva_constraints	<i>Constraints on predictors and components</i>
------------------	---

Description

Ecological knowledge enters a CCVA as explicit constraints, at two levels:

Usage

```
ccva_constraints(..., defaults = TRUE)

ccva_default_constraints()

ccva_unconstrained()

constrain_sign(target, sign = c("+", "-", "direction"))

constrain_bounds(target, lower = -Inf, upper = Inf)

constrain_equal(targets)

constrain_fixed(target, value)

constrain_link(from, to, link_sign = c("free", "+", "-"))

constrain_shape(target, shape = c("concave", "convex"))

constrain_moderator(gamma = c(0.5, 0.4))
```

Arguments

...	‘ccva_rule’ objects from the ‘constrain_*()’ helpers.
defaults	Logical. If ‘TRUE’ (default), start from [ccva_default_constraints()] (component signs plus indicator signs taken from the data’s ‘direction’) and add ‘...’ on top; later rules for the same target override earlier ones.
target	For component-level rules, one of “exposure”, “sensitivity”, “adaptive”. For predictor-level rules, an indicator or outcome column name. “.indicators” applies to every indicator and outcome.

sign	"+" , "-" , or for ".indicators" the special value "direction" (use each column's direction from [as_ccva()]).
lower, upper	Bounds on the effect (component level) or loading / weight (predictor level). Use '-Inf' / 'Inf' for one-sided bounds.
targets	Character vector of two or more indicators (within the same component) whose loadings / weights are constrained equal.
value	Fixed value. '0' removes the indicator or component effect.
from, to	Components for a directed structural link: 'from' affects 'to' (e.g. adaptive capacity through dispersal shapes the SDM-derived exposure a species will face).
link_sign	Sign of the link: "free" , "+" or "-" .
shape	For the shape-constrained model: "concave" (e.g. an exposure effect that rises steeply then plateaus) or "convex" (accelerating, e.g. highly sensitive species disproportionately affected). Combined with the component's sign this gives monotone concave / convex splines. Unimodal shapes are not available.
gamma	Moderation strengths 'c(exposure, sensitivity)' used by the weighted method: adaptive capacity reduces the exposure and sensitivity effects by up to these proportions.

Details

*****Predictor level**** – the role of an indicator (or outcome proxy) within its component: the sign of its loading or weight, bounds, equality with other indicators, or a fixed value (0 drops it). *****Component level**** – how a component enters vulnerability: exposure and sensitivity raise it, adaptive capacity lowers it; bounds on the effects; or adaptive capacity acting only as a **moderator** of exposure and sensitivity, with no main effect. *****Between components**** – the three components are not independent. [constrain_link()] declares a directed dependency, e.g. dispersal-driven adaptive capacity shaping SDM-derived exposure. In the latent SEM it is a structural path, and the indirect and total effects of the source component on vulnerability are reported; in the other models it is resolved in two stages, by removing from the target component the part explained by the source, so it is not counted twice.

Constraints are built from the 'constrain_*()' helpers and collected with 'ccva_constraints()'. The same object is honoured by every combination method of [ccva_combine()] that can express it: sign, bound, equality and fixed constraints become inequality / equality restrictions on loadings and paths in the latent SEM; signs become monotone-increasing or -decreasing splines in the shape-constrained model and coefficient limits in the penalised model; the moderator role becomes an interaction surface in the shape and weighted models. A constraint a method cannot express is reported, never silently ignored.

Sign constraints on loadings also solve the sign-indeterminacy of latent factors: with every loading pinned to its ecological direction, the latent scores cannot come out reversed.

Value

An object of class 'ccva_constraints'.

See Also

[ccva_combine()], [ccva_register()]

Examples

```
# Default ecological constraints
ccva_constraints()

# Adaptive capacity as moderator only, thermal tolerance loading fixed
# equal to range size, and generation length dropped
ccva_constraints(
  constrain_moderator(),
  constrain_equal(c("s1_thermal", "s2_range")),
  constrain_fixed("s10_genlen", 0)
)

# No constraints at all (for comparison)
ccva_unconstrained()
```

ccva_methods

List available methods

Description

List available methods

Usage

```
ccva_methods()
```

Value

A data frame of built-in and registered methods.

Examples

```
ccva_methods()
```

ccva_plot

Plot any step of a CCVA

Description

One entry point for the figures of every stage. ‘ccva_plot()’ dispatches on the class of ‘object’; ‘type’ picks the figure. Every plot is a ‘ggplot’ object, so it can be themed, combined or saved with ‘ggplot2::ggsave()’.

Usage

```
ccva_plot(object, type = NULL, n = 15, ...)
```

Arguments

object	A 'ccva_data', 'ccva_check', 'ccva_component', 'ccva_fit', 'ccva_comparison' or 'ccva_uncertainty' object.
type	Figure type (see table). 'NULL' gives the first listed.
n	For rankings and interval plots, the number of units shown at each end ("ranking") or in total ("intervals", "pairwise").
...	Unused.

Details

| object | 'type' | shows | |—|—|—| | 'ccva_data' | "indicators" | distribution of every indicator, grouped by component | | "correlation" | indicator correlation matrix, ordered by component | | 'ccva_check' | "alpha" | Cronbach's alpha per component against thresholds | | "correlation" | cross-component correlations | | 'ccva_component' | "importance" | weight / loading / coefficient of each indicator | | "distribution" | distribution of the component score | | 'ccva_fit' | "structure" | path diagram: components to vulnerability, with effects | | "components" | the three component scores side by side | | "scores" | vulnerability distribution with class cut-offs | | "ranking" | most and least vulnerable units with their components | | "contributions" | how each component pushes each unit's score (weighted) | | "effects" | shape curves, latent loadings or penalised coefficients | | "map" | vulnerability over coordinates | | 'ccva_comparison' | "truth" | each method against the true vulnerability | | "ranks" | rank of each unit under two methods | | "validation" | correlation of each method with each target | | 'ccva_uncertainty' | "intervals" | score intervals, ranked | | "pairwise" | probability one unit is more vulnerable than another |

Value

A 'ggplot' object.

Examples

```
d <- ccva_sim_data()
ccva_plot(d, "indicators")
fit <- ccva_combine(d, "weighted", component_method = "ordinal")
ccva_plot(fit, "structure")
ccva_plot(fit, "ranking", n = 8)
```

ccva_rank	<i>Rank units by vulnerability</i>
-----------	------------------------------------

Description

Rank units by vulnerability

Usage

```
ccva_rank(fit, uncertainty = NULL)
```

Arguments

`fit` A 'ccva_fit'.

`uncertainty` Optional [ccva_uncertainty()] result to add intervals.

Value

Data frame ordered from most to least vulnerable.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted")
head(ccva_rank(fit))
```

ccva_recommend	<i>Match methods to the data available</i>
----------------	--

Description

Turns general guidance – shape-constrained parametric models for small samples, regularised regression for regional compilations, flexible non-parametric methods for global databases, with internal consistency checks, structural or two-stage handling of dependencies among components, and honest uncertainty in every case – into concrete advice for a given dataset, based on the number of units and the number of predictors per component.

Usage

```
ccva_recommend(x)
```

Arguments

`x` A 'ccva_data' object.

Details

The sample-size thresholds (50 and 150 units) reflect rough feasibility: correlated additive latent models need about 50 units, interaction surfaces about 150. They are guides, not rules.

Value

An object of class 'ccva_recommendation' with the 'tier', a 'profile' of the data and a table of 'steps'.

Examples

```
ccva_recommend(ccva_sim_data())
ccva_recommend(ccva_simulate(40, seed = 1))
```

ccva_register	<i>Register a new component or combination method</i>
---------------	---

Description

The framework grows by contribution. A registered method is used exactly like a built-in one, through the ‘method’ argument of the component scorers or of [ccva_combine()].

Usage

```
ccva_register(name, fun, type = c("component", "combine"), description = "")
ccva_unregister(name, type = c("component", "combine"))
```

Arguments

name	Method name.
fun	The method function.
type	“component” or “combine”.
description	One-line description shown by [ccva_methods()].

Details

A **component** method is a function ‘function(X, anchor, signs, ...)’ where ‘X’ is the oriented indicator matrix (higher = more of the component), ‘anchor’ an optional numeric vector (oriented vulnerability proxy) and ‘signs’ the resolved indicator sign constraints. It must return a list with ‘score’ (numeric, one per row) and ‘predict’, a function of a new ‘X’ returning scores on the same scale.

A **combine** method is a function ‘function(x, components, constraints, ...)’ receiving the ‘ccva_data’, the fitted component scores (data frame) and the resolved constraints. It must return a list with ‘score’ (vulnerability, higher = more vulnerable) and optionally ‘model’, ‘details’ and ‘predict’ (a function of new ‘ccva_data’).

Value

Invisibly, the updated registry listing.

Examples

```
# A median-based component score
ccva_register("median", function(X, anchor, signs, ...) {
  z <- scale(X)
  list(score = apply(z, 1, stats::median),
        predict = function(newX) apply(scale(newX,
        attr(z, "scaled:center"), attr(z, "scaled:scale")), 1,
        stats::median))
}, type = "component", description = "Median of standardised indicators")
```

```
ccva_methods()
ccva_unregister("median", "component")
```

ccva_report

Write a reproducible CCVA report

Description

Renders an HTML report of a fit: the data and diagnostics, every component, the combination model and its constraints, the ranking, and optionally uncertainty and a method comparison. Needs **rmarkdown** (and pandoc).

Usage

```
ccva_report(
  fit,
  file = file.path(tempdir(), "ccva_report.html"),
  uncertainty = NULL,
  comparison = NULL,
  title = "Climate change vulnerability assessment",
  open = interactive()
)
```

Arguments

fit	A 'ccva_fit'.
file	Output HTML file. Defaults to a file in the session's temporary directory; give a path to keep the report.
uncertainty	Optional [ccva_uncertainty()] result.
comparison	Optional [ccva_compare()] result.
title	Report title.
open	Open the report in a browser when done.

Value

The path to the report, invisibly.

Examples

```
if (requireNamespace("rmarkdown", quietly = TRUE) &&
    rmarkdown::pandoc_available()) {
  fit <- ccva_combine(ccva_sim_data(), "weighted")
  path <- ccva_report(fit, open = FALSE)
  file.exists(path)
}
```

ccva_sdm_exposure	<i>Compute exposure predictors from species distribution model output</i>
-------------------	---

Description

Turns the cell counts of thresholded current and future SDM habitat maps into the three exposure indices of Ofori et al. (2017), and recodes them so that a higher value always means more exposure (risk direction), ready to be used as exposure indicators in [as_ccva()].

Usage

```
ccva_sdm_exposure(n_current, n_future, n_overlap, n_protected = NULL)
```

Arguments

n_current	Number of suitable cells under current climate.
n_future	Number of suitable cells under future climate.
n_overlap	Number of cells suitable under both.
n_protected	Optional number of currently suitable cells inside protected areas.

Details

$$E_1 = (N_{future} - N_{current})/N_{current}$$

$$E_2 = N_{overlap}/N_{current}$$

$$E_3 = N_{protected}/N_{current}$$

with risk recoding $E_{1,risk} = -E_1$, $E_{2,risk} = 1 - E_2$, $E_{3,risk} = 1 - E_3$. The three share the denominator $N_{current}$, so their residuals are correlated; declare them in ‘shared_error’ of [ccva_combine()] when using the latent model.

The SDM itself (MaxEnt, BIOCLIM, an ensemble, ...) is fitted with the user’s preferred package; this function only needs the cell counts.

Value

A data frame with the raw indices (‘E1_change’, ‘E2_overlap’, ‘E3_protected’) and their risk-coded versions (‘E1_risk’, ‘E2_risk’, ‘E3_risk’).

Examples

```
ccva_sdm_exposure(n_current = c(120, 80), n_future = c(60, 95),
                  n_overlap = c(50, 70), n_protected = c(30, 10))
```

ccva_sim

*Simulated CCVA dataset***Description**

300 simulated species generated by [ccva_simulate()] with ‘seed = 99’ and the linear structural model. Contains the observed columns a modeller would see plus the latent truth for validation. All values are simulated and have no biological meaning.

Usage

ccva_sim

Format

A data frame with 300 rows and 26 columns:

species Unit identifier.

E1_risk Exposure: habitat loss risk (SDM-derived, higher = more exposed).

E2_risk Exposure: lack of overlap between current and future habitat.

E3_risk Exposure: lack of protection of current habitat. The three exposure risks share a denominator, hence correlated residuals.

s1_thermal Sensitivity: thermal tolerance (higher = more sensitive).

s2_range Sensitivity: small geographic range.

s3_climzones Sensitivity: few climate zones occupied.

s4_habitat Sensitivity: habitat specialisation.

s5_diet Sensitivity: diet specialisation.

s6_reprorate Sensitivity: low reproductive rate.

s7_repromode Sensitivity: reproductive mode.

s8_clutch Sensitivity: small clutch size.

s9_survival Sensitivity: low offspring survival.

s10_genlen Sensitivity: long generation length.

s11_lifespan Sensitivity: lifespan.

a1_genetic Adaptive capacity: genetic diversity (higher = more capacity).

a2_bodysize Adaptive capacity: body size as a dispersal proxy.

a3_lowfrag Adaptive capacity: low habitat fragmentation.

a4_microhab Adaptive capacity: microhabitat buffering.

iucn IUCN-like status, 0 (least concern) to 3 (endangered).

pop_trend Population trend: -1 declining, 0 stable, 1 increasing.

range_change Proportional range change; negative = loss.

E_star Latent exposure (simulation truth).

S_star Latent sensitivity (simulation truth).

A_star Latent adaptive capacity (simulation truth).

V_star Latent vulnerability (simulation truth).

Source

Simulated with [ccva_simulate()]; see ‘data-raw/ccva_sim.R’.

Examples

```
head(ccva_sim[, 1:6])
```

ccva_simulate

Simulate CCVA data with a known latent structure

Description

Generates data from a three-layer data-generating process, so that every method can be checked against the truth:

Usage

```
ccva_simulate(
  n = 300,
  nonlinear = FALSE,
  phi = matrix(c(1, 0.3, -0.2, 0.3, 1, -0.25, -0.2, -0.25, 1), 3, 3),
  b = c(E = 0.5, S = 0.35, A = 0.25),
  sd_zeta = 0.3,
  n_genomic = 0,
  n_causal = 10,
  link_AE = 0,
  seed = NULL
)
```

Arguments

n	Number of units (species). The full latent model has 21 indicators and four latent factors; use a few hundred for stable fits.
nonlinear	Logical; use the nonlinear structural model for V^* .
phi	3 x 3 correlation matrix of (E^*, S^*, A^*) .
b	Structural coefficients ‘c(E, S, A)’; adaptive capacity enters with a negative sign.
sd_zeta	Standard deviation of the structural disturbance.
n_genomic	Number of genomic markers (coded 0/1/2) to add to the adaptive-capacity block, for testing penalised methods when predictors outnumber units. The first ‘n_causal’ markers carry signal from A^* ; the rest are noise.
n_causal	Number of informative genomic markers.
link_AE	Strength of a directed dependency of exposure on adaptive capacity, $E^* \leftarrow E^* + link_{AE}A^*$. A negative value mimics dispersal: species with more capacity to move face less of the SDM-derived exposure. Default 0 (no link).
seed	Optional random seed.

Details

***Layer 1** – correlated latent exposure E^* , sensitivity S^* and adaptive capacity A^* . ***Layer 2** – latent vulnerability $V^* = b_E E^* + b_S S^* - b_A A^* + \zeta$ (linear), or a nonlinear version with an exposure plateau, convex sensitivity and adaptive capacity acting only as a moderator. ***Layer 3** – observed indicators: three SDM-derived exposure risks with a shared-denominator nuisance, eleven sensitivity traits, four adaptive-capacity traits, optional genomic markers, and three vulnerability proxies (IUCN-like status, population trend, range change).

All indicators are continuous except the proxies ‘iucn’ (0–3) and ‘pop_trend’ (-1, 0, 1), which are ordinal by nature. Ofori-style 1–3 scores are produced later by ‘method = "ordinal"’ in the component scorers, so the information lost by discretising can be measured.

Value

A ‘ccva_data’ object (see [as_ccva()]) with an extra element ‘truth’, a data frame of the latent values ‘E_star’, ‘S_star’, ‘A_star’, ‘V_star’. Use ‘x\$data’ for the raw data frame.

See Also

[ccva_sim] for a bundled simulated dataset.

Examples

```
sim <- ccva_simulate(n = 200, seed = 1)
sim
head(sim$truth)

# High-dimensional adaptive-capacity block (p > n)
sim_g <- ccva_simulate(n = 150, n_genomic = 400, seed = 1)
length(sim_g$indicators$adaptive)
```

ccva_sim_data

Specification of the bundled simulated data

Description

Convenience wrapper that turns [ccva_sim] into a ‘ccva_data’ object with the correct component tags and directions, and attaches the latent truth.

Usage

```
ccva_sim_data(data = ccva::ccva_sim)
```

Arguments

data Defaults to [ccva_sim].

Value

A ‘ccva_data’ object with a ‘truth’ element.

Examples

```
d <- ccva_sim_data()
d
```

ccva_thermal_margin	<i>Compute thermal sensitivity predictors</i>
---------------------	---

Description

Mechanistic sensitivity predictors from measured thermal limits: warming tolerance $WT = CT_{max} - T_{hab}$ and thermal safety margin $TSM = T_{opt} - T_{hab}$. Smaller (or negative) values mean the organism is closer to its limit, i.e. more sensitive, so give these columns direction ‘-1’ in [as_ccva()].

Usage

```
ccva_thermal_margin(ctmax, t_habitat, topt = NULL)
```

Arguments

ctmax	Critical thermal maximum.
t_habitat	Habitat (or operative body) temperature experienced by the unit, e.g. extracted from a climate layer at its coordinates.
topt	Optional thermal optimum.

Value

A data frame with ‘warming_tolerance’ and, if ‘topt’ is given, ‘safety_margin’.

Examples

```
ccva_thermal_margin(ctmax = c(42, 45), t_habitat = c(31, 28),
  topt = c(35, 36))
```

Description

Published CCVAs usually report a sorted list of point estimates, which implies a precision the data rarely support. Two complementary sources of uncertainty are available:

Usage

```
ccva_uncertainty(
  fit,
  type = c("bootstrap", "weights"),
  R = 200,
  level = 0.95,
  robust = 0.05,
  score_error = TRUE,
  seed = NULL,
  progress = interactive()
)
```

Arguments

fit	A ‘ccva_fit’.
type	“bootstrap” or “weights”.
R	Number of replicates.
level	Interval coverage.
robust	Threshold for calling a pairwise ranking robust.
score_error	For the latent model, add each unit’s factor-score prediction error to the bootstrap draws (recommended).
seed	Optional random seed.
progress	Show a progress bar.

Details

* ‘type = “bootstrap” – **sampling uncertainty**: units are resampled with replacement, the **whole** pipeline (component scoring, constraints, combination model) is refitted on each resample, and the original units are scored with the refitted model. For the latent model each unit’s score is a prediction of an unobserved quantity, so (with ‘score_error = TRUE’) its factor-score standard error is added to every replicate; without it, intervals reflect parameter uncertainty only and are far too narrow. * ‘type = “weights” – **weighting uncertainty** for the weighted index (a partial-identification view): component weights are drawn uniformly from the simplex, so the intervals show how much a ranking depends on the arbitrary choice of weights.

For each unit the result gives an interval for its score and for its rank; for each pair of units, the probability that one is more vulnerable than the other. A ranking of two units is called **robust** when that probability is above ‘1 - robust’ or below ‘robust’.

Value

An object of class 'ccva_uncertainty' with 'summary' (one row per unit: estimate, interval, rank interval), 'pairwise' (matrix of P(row unit more vulnerable than column unit)), 'robust_share', 'overlap_share' and the raw 'draws'.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted")
u <- ccva_uncertainty(fit, "weights", R = 500, seed = 1)
u

ub <- ccva_uncertainty(fit, "bootstrap", R = 50, seed = 1)
head(ub$summary)
```

ccva_validate

Validate vulnerability scores against observed outcomes

Description

A CCVA score that has never been compared with any observable outcome is unvalidated. This reports the rank correlation of a fit's scores with every outcome proxy (oriented so a positive correlation is good) and, for simulated data, with the latent truth.

Usage

```
ccva_validate(fit, truth = NULL)
```

Arguments

fit	A 'ccva_fit'.
truth	Optional numeric vector of true vulnerability. If 'NULL', 'fit\$data\$truth\$V_star' is used when present.

Details

Note: when the outcomes were also used to fit the model (latent, shape and penalised methods), correlation with them measures fit, not out-of-sample validity; use held-out outcomes or the truth for that.

Value

A data frame with one row per target: Spearman and Kendall correlations.

Examples

```
fit <- ccva_combine(ccva_sim_data(), "weighted")
ccva_validate(fit)
```

predict.ccva_fit	<i>Predict vulnerability for new units</i>
------------------	--

Description

Predict vulnerability for new units

Usage

```
## S3 method for class 'ccva_fit'
predict(object, newdata, ...)
```

Arguments

object	A 'ccva_fit'.
newdata	A 'ccva_data' object (or data frame with the same columns) for the new units.
...	Unused.

Value

Numeric vector of vulnerability on the scale of the training fit (values outside $[0, 1]$ indicate units beyond the training range).

theme_ccva	<i>Colours and theme used by ccva plots</i>
------------	---

Description

'ccva_colours' maps each component (and vulnerability) to a fixed colour, so a component looks the same in every figure. The palette was checked for colour-vision-deficiency separation; facet and direct labels carry identity too, so colour is never the only cue.

Usage

```
theme_ccva(base_size = 11)

ccva_colours
```

Arguments

base_size	Base font size.
-----------	-----------------

Format

'ccva_colours': a named character vector.

Value

`'theme_ccva()'` returns a ggplot2 theme.

Examples

```
ccva_colours
```

Index

- * **datasets**
 - ccva_sim, [23](#)
 - theme_ccva, [29](#)
- as_ccva, [2](#)
- ccva, [4](#)
- ccva_adaptive (ccva_component), [13](#)
- ccva_adjust_method, [5](#)
- ccva_aggregate, [6](#)
- ccva_calibrate, [7](#)
- ccva_check, [8](#)
- ccva_classify, [9](#)
- ccva_colours (theme_ccva), [29](#)
- ccva_combine, [10](#)
- ccva_compare, [12](#)
- ccva_component, [13](#)
- ccva_constraints, [15](#)
- ccva_default_constraints (ccva_constraints), [15](#)
- ccva_exposure (ccva_component), [13](#)
- ccva_methods, [17](#)
- ccva_plot, [17](#)
- ccva_rank, [18](#)
- ccva_recommend, [19](#)
- ccva_register, [20](#)
- ccva_report, [21](#)
- ccva_sdm_exposure, [22](#)
- ccva_sensitivity (ccva_component), [13](#)
- ccva_sim, [23](#)
- ccva_sim_data, [25](#)
- ccva_simulate, [24](#)
- ccva_thermal_margin, [26](#)
- ccva_uncertainty, [27](#)
- ccva_unconstrained (ccva_constraints), [15](#)
- ccva_unregister (ccva_register), [20](#)
- ccva_validate, [28](#)
- constrain_bounds (ccva_constraints), [15](#)
- constrain_equal (ccva_constraints), [15](#)
- constrain_fixed (ccva_constraints), [15](#)
- constrain_link (ccva_constraints), [15](#)
- constrain_moderator (ccva_constraints), [15](#)
- constrain_shape (ccva_constraints), [15](#)
- constrain_sign (ccva_constraints), [15](#)
- predict.ccva_fit, [29](#)
- theme_ccva, [29](#)