Package: EpiExposure (via r-universe)

November 6, 2026

Type Package

Title Exposure-Lag-Response Modeling for Plant Disease Epidemiology

Version 0.1.0

Description Supports modeling and interpretation of environmental exposure-lag-response relationships in plant disease epidemiology using distributed lag non-linear models (DLNMs). Provides functions for defining complete exposure profiles, evaluating model identifiability, summarizing lag-specific and period-specific effects, performing sensitivity analyses, simulating environmental exposure scenarios, predicting plant disease outcomes, quantifying yield and economic losses, and evaluating individual and ensemble models. Supports harmonized frequentist and Bayesian modeling workflows. Builds on the framework implemented in the 'dlnm' package and described by Gasparrini (2011) <doi:10.18637/jss.v043.i08>.

Depends R (>= 4.1)

Imports dlnm, stats, glmmTMB, brms, spaMM, rlang, mgcv, nlme, ggplot2, scales

Suggests dplyr, tidyr, purrr, tidyselect, INLA, knitr, rmarkdown, testthat (>= 3.0.0), pkgdown, readxl, patchwork, metR, bdlnm, cowplot, future, sn, future.apply

Additional_repositories https://inla.r-inla-download.org/R/stable

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

URL https://github.com/tomazrg/EpiExposure, https://tomazrg.github.io/EpiExposure/

BugReports https://github.com/tomazrg/EpiExposure/issues

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Ricardo Gomes Tomáz [aut, cre] (ORCID: <https://orcid.org/0000-0001-7708-4263>), Emerson M. Del Ponte [aut] (ORCID: <https://orcid.org/0000-0003-4398-409X>), Felipe Dalla Lana [aut] (ORCID: <https://orcid.org/0000-0002-3112-6197>)

Maintainer Ricardo Gomes Tomáz <ricardogomesagro@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-06 18:06:31 UTC

RemoteUrl https://github.com/cran/EpiExposure

RemoteRef HEAD

RemoteSha 58dc552d55160d328e6b2caa62b6f5bad6a0e901

Contents

binomial_data . . . 3
build_design . . . 4
check_identifiability . . . 6
compare_exposures . . . 10
compare_periods . . . 14
compare_predictions . . . 18
compute_eci . . . 22
compute_ecilag . . . 29
define_exposures . . . 33
define_periods . . . 36
ensemble_bestfit . . . 38
epi_data . . . 43
epi_sensitivity . . . 44
find_bestfit . . . 49
fit_epidlnm . . . 56
gamma_data . . . 61
gaussian_data . . . 62
nb2_data . . . 63
plot_eci . . . 64
plot_effects . . . 66
plot_losses . . . 70
plot_performance . . . 75
plot_scenarios . . . 79
poisson_data . . . 87
predict_outcomes . . . 88
prepare_response . . . 91
reduce_effects . . . 93
simulate_exposures . . . 98

simulate_losses 101
simulate_ranges 106
simulate_scenarios 108
spatial_replicate 112
spatial_year 113
st_poisson 114
summarise_effects 115

Index **120**

`binomial_data` *Simulated binomial epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and a binary plant disease outcome. The dataset is intended for demonstrating binomial models in EpiExposure.

Usage

```
binomial_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

rain Daily rainfall in millimeters.

wetness Daily leaf wetness duration in hours.

y Binary disease outcome coded as 0 or 1 and repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("binomial_data", package = "EpiExposure")
head(binomial_data)
```

build_design *Build an epidemic-level DLNM design matrix*

Description

Converts complete long-format exposure profiles into one epidemic-level design row per ‘epi_id‘. For each exposure, the function reconstructs a new DLNM cross-basis from that epidemic’s observed exposure profile using the **effective basis parameterization stored in the supplied training template**, then retains the final cross-basis row after chronological ordering.

Usage

```
build_design(
  data,
  cb_templates,
  max_lag = NULL,
  include_response = TRUE,
  groups = "epi_id"
)
```

Arguments

data	Non-empty long-format data frame containing ‘epi_id‘, ‘time‘, and every exposure represented in ‘cb_templates‘. Rows are ordered internally by ‘epi_id‘ and ‘time‘. Under the EpiExposure exact-profile contract, every epidemic must contain exactly the common fitted ‘max_lag + 1‘ time points. Because all exposure variables are finite columns of those same rows, all fitted exposures necessarily use the same temporal support and profile length. If a column ‘y‘ is present and ‘include_response = TRUE‘, ‘y‘ must contain exactly one finite numeric outcome value per epidemic.
cb_templates	Named list of ‘dlnm::crossbasis‘ objects, normally returned by ‘define_exposures()‘. Every element must contain valid effective ‘argvar‘, ‘arglag‘, and ‘lag‘ attributes. The cross-basis objects themselves are not reused numerically for new epidemics. Their stored effective parameterization is reused to transform each new exposure profile.
max_lag	Optional lag validation argument retained for backward compatibility. ‘cb_templates‘ are the authoritative source of the fitted lag definition and ‘max_lag‘ never overrides them. All templates must share one common maximum lag. If ‘NULL‘, that common lag is read directly from the templates. If supplied, ‘max_lag‘ may be one non-negative integer scalar or the legacy two-element form ‘c(0, L)‘; after normalization its maximum must equal the common template lag. Exposure-specific/named lag windows are not supported by the current EpiExposure exact-profile contract.

include_response
: Logical. If ‘TRUE‘ and ‘data‘ contains ‘y‘, append one response value per epidemic. If ‘FALSE‘, ‘y‘ is ignored.

groups
: Non-empty character vector of identifier columns to preserve in the final design. Must include ‘"epi_id"‘; the default is ‘"epi_id"‘. Additional identifiers, such as ‘"block"‘, ‘"site"‘, or ‘"year"‘, must exist in ‘data‘, contain no missing values, and be constant within each ‘epi_id‘. They do not redefine the exposure-profile unit, which remains exclusively ‘epi_id‘.

Details

‘build_design()‘ never re-estimates spline knots, boundary knots, degrees of freedom, or lag-basis parameters from the input ‘data‘. The supplied ‘cb_templates‘ are authoritative for basis transport.

Temporal requirements

A vector supplied to ‘dlnm::crossbasis()‘ is interpreted as one complete, equally spaced exposure profile. Accordingly, ‘build_design()‘ requires finite numeric time, unique times within epidemics, constant spacing within each epidemic when more than one time point is present, and the same spacing across epidemics.

If the common fitted maximum lag is ‘L‘, every epidemic must contain exactly

$$L + 1$$

observations. Profiles with fewer or more observations are rejected. ‘build_design()‘ never truncates an older profile, selects a trailing window, pads a shorter profile, or silently aligns exposures with different lag windows. Since every exposure is evaluated on the same rows of ‘data‘, all fitted exposures have identical temporal length and support.

For ‘max_lag = 0‘, the exact profile length is one observation. In that special case no within-profile time interval exists, so ‘epiexposure_time_step‘ is stored as ‘NA‘.

Training-template transport

For each exposure ‘x‘, reconstruction is conceptually:

“‘ dlnm::crossbasis(x, lag = attr(training_template, "lag"), argvar = attr(training_template, "argvar"), arglag = attr(training_template, "arglag")) “‘

The numerical cross-basis values change with the new exposure profile while the training basis definition remains fixed. Reconstructed bases must retain the training template’s number of columns and native column names/order before canonical EpiExposure names are assigned.

Template storage

‘attr(result, "cb_templates")‘ stores the **original supplied templates**, not a cross-basis reconstructed from the first epidemic. This preserves the original training basis definition for later prediction and validation.

Value

A data frame with one row per epidemic containing the identifier columns specified in ‘groups‘, canonical cross-basis columns named ‘cb_<exposure>_<column_index>‘, and, optionally, ‘y‘.

The returned data frame stores the following attributes:

- ‘"cb_templates"‘: the original supplied cross-basis templates; - ‘"epiexposure_spec"‘: effective per-exposure basis specification synchronized to the template attributes; - ‘"epiexposure_cb_cols"‘: canonical cross-basis column order; - ‘"epiexposure_vars"‘: exposure names; - ‘"epiexposure_max_lag"‘: common fitted maximum lag; - ‘"epiexposure_history_length"‘: exact required profile length, ‘max_lag + 1‘; - ‘"epiexposure_history_contract"‘: ‘"all_fitted_exposures_same_exact_max_lag_plus_one"‘; - ‘"epiexposure_time_step"‘: common time-series spacing, or ‘NA‘ when ‘max_lag = 0‘ because a one-point profile has no estimable spacing; - ‘"epiexposure_design_contract"‘: ‘"final_crossbasis_row_per_group"‘; - ‘"epiexposure_group_cols"‘: identifier columns preserved in the design.

`check_identifiability` *Diagnose identifiability and numerical stability of DLNM cross-basis designs*

Description

Diagnoses the epidemic-level DLNM design that EpiExposure would construct for model fitting from one or more exposure variables. The function deliberately reuses the canonical EpiExposure design path:

Usage

```
check_identifiability(
  data,
  vars,
  max_lag,
  df_var = 4,
  df_lag = 4,
  fun_var = "ns",
  fun_lag = "ns",
  include_intercept = TRUE,
  corr_threshold = 0.9,
  condition_warn = 30,
  condition_severe = 100,
  vif_threshold = 10,
  n_per_parameter_warn = 10,
  tol = 1e-07,
  keep_design = FALSE
)
```

Arguments

`data`	Non-empty long-format exposure data frame containing ‘epi_id‘, ‘time‘, and every exposure named in ‘vars‘. EpiExposure v1 uses a strict exact-profile contract. If ‘max_lag = L‘, every epidemic must contain exactly ‘L + 1‘ rows. Profiles with fewer or more rows are rejected; they are never truncated, padded, or silently realigned. All requested

exposure variables occupy those same validated rows and therefore necessarily have the same temporal support and profile length.

'epi_id' must not contain missing values. 'time' and all exposure variables must contain only finite numeric values. When 'max_lag > 0', time must be strictly increasing after ordering, equally spaced within each epidemic, and use the same spacing across epidemics. Missing or non-finite exposure values are not dropped in EpiExposure v1; they are errors.

`vars` Character vector with one or more unique exposure-variable names. The reserved EpiExposure names '"epi_id"', '"time"', and '"y"' cannot be used as exposure names.

`max_lag` Non-negative integer maximum lag. The legacy form 'c(0, L)' is accepted for backward compatibility. EpiExposure v1 requires the fitted lag range to begin at lag 0 and all fitted exposures to use the same 'max_lag'.

`df_var` Positive integer controlling the exposure-response basis when 'fun_var' is '"ns"' or '"bs"'. For 'fun_var = "poly"', it is used as the polynomial degree. It is ignored by the linear basis.

`df_lag` Positive integer controlling the lag-response basis when 'fun_lag' is '"ns"' or '"bs"'. For 'fun_lag = "poly"', it is used as the polynomial degree. It is ignored by the linear basis.

`fun_var` Character exposure-basis function. Supported unpenalized EpiExposure v1 choices are '"ns"', '"bs"', '"poly"', and '"lin"'.

`fun_lag` Character lag-basis function. Supported unpenalized EpiExposure v1 choices are '"ns"', '"bs"', '"poly"', and '"lin"'.

Penalized dlnm basis functions '"ps"' and '"cr"' are deliberately not supported in EpiExposure v1. A penalized spline transformation alone is not a penalized DLNM: the associated penalty matrices and smoothing parameters must also be propagated to model fitting. The current engine-agnostic fitting contract therefore stops explicitly instead of diagnosing a basis that EpiExposure would later fit without its penalty.

`include_intercept`
Logical scalar. If 'TRUE' (default), include a model intercept when evaluating numerical rank of the combined epidemic-level design. This matches the standard EpiExposure fitting parameterization. 'FALSE' is retained as an advanced diagnostic option.

`corr_threshold` Finite number in '(0, 1]'. Heuristic absolute correlation threshold used to flag potentially important between-exposure cross-basis collinearity and to report high same-time raw-exposure correlation. Default is '0.90'.

`condition_warn` Positive finite number. Heuristic scaled condition number at or above which a numerical warning is reported. Default is '30'.

`condition_severe`
Positive finite number strictly larger than 'condition_warn'. At or above this value the design is classified as numerically unstable. Default is '100'.

`vif_threshold` Positive finite number. Threshold for supplementary VIF reporting. Default is '10'. VIFs for individual spline columns can be high by construction and do not independently determine the global status.

`n_per_parameter_warn`
: Positive finite number. Heuristic minimum ratio of complete epidemics to columns in the rank design. Default is ‘10‘. This is a design-complexity diagnostic, not a universal sample-size rule.

`tol`
: Positive finite number used for numerical QR-rank decisions, near-zero-variance checks, and numerical zero checks in the supplementary VIF calculation. Default is ‘1e-7‘. Temporal validation itself follows the canonical tolerances in ‘define_exposures()‘ and ‘build_design()‘.

`keep_design`
: Logical scalar. If ‘TRUE‘, include the complete epidemic-level numerical matrix used for the rank diagnostic in the returned object. Default is ‘FALSE‘.

Details

“‘ templates <- define_exposures(...) design <- build_design(data, templates, include_response = FALSE) “‘

This avoids maintaining a second, potentially divergent implementation of cross-basis construction inside the diagnostic function. Consequently, the basis definitions, training-data-dependent knots, lag convention, marginal intercept rules, canonical cross-basis column order, and final-row design used here are the same as those used by the current EpiExposure fitting workflow.

What is diagnosed

EpiExposure fits one epidemic-level row after transforming each complete exposure profile through its training cross-basis and retaining the final cross-basis row. ‘check_identifiability()‘ diagnoses that same numerical design. It does not diagnose every intermediate row returned by ‘dlnm::crossbasis()‘ within a profile.

The function first calls ‘define_exposures()‘ to estimate the training basis definitions jointly from all supplied epidemics, then calls ‘build_design()‘ to reconstruct one final cross-basis row per epidemic using those fixed training definitions. This is the authoritative EpiExposure v1 design path.

Exact-profile and common-window contract

If the fitted maximum lag is ‘L‘, each epidemic must contain exactly

$$L + 1$$

chronological exposure observations, corresponding to lag ‘L‘ through lag ‘0‘. All fitted exposures use the same ‘L‘. A longer profile is not silently reduced to its last ‘L + 1‘ rows, and a shorter profile is not padded.

For ‘max_lag = 0‘, each epidemic contains exactly one observation and the lag basis must be linear. A one-point profile has no estimable temporal step, so ‘time_step‘ is stored as ‘NA‘.

Identifiability versus numerical stability

‘identifiable‘ is based on the **numerical rank** of the epidemic-level design at the user-selected tolerance; it is not an algebraic symbolic-rank proof. When ‘include_intercept = TRUE‘, the intercept is included in this rank calculation.

Numerical stability is assessed separately. The condition number is computed from usable cross-basis predictor columns after centering and scaling each column to unit standard deviation. The intercept is not included in this scaled condition number because centering makes it orthogonal to

the scaled predictor columns. If the scaled predictor matrix is rank deficient, or if the number of usable columns cannot be supported by its rows, the condition number is reported as infinite.

Condition-number cutoffs are heuristics, not universal inferential laws. Similarly, the epidemics-per-column ratio is a descriptive warning about design complexity and should not be interpreted as a formal sample-size calculation.

Correlations and VIFs

Correlations among columns from the same spline basis are expected and are not, by themselves, evidence that the model is invalid. The combined rank and scaled condition number are the primary numerical diagnostics.

‘pairwise_crossbasis_correlation‘ examines **different exposure blocks** and can therefore highlight two exposures that contribute highly redundant transformed temporal information. It reports the maximum pairwise column correlation; this is not a canonical-correlation analysis.

Raw-exposure correlations are calculated on the long-format rows and are purely descriptive. Repeated/serial observations are not treated as independent observations for inferential testing, and the function does not report correlation p-values. High raw correlation alone does not change the global ‘status‘ when the fitted cross-basis design remains well behaved.

VIFs are also supplementary because spline-expanded columns are commonly correlated by construction. A high VIF produces a recommendation but, on its own, does not upgrade an otherwise acceptable design to ‘"warning"‘.

Penalized DLNMs

EpiExposure v1 is intentionally unpenalized. ‘"ps"‘ and ‘"cr"‘ are rejected here for the same reason they are rejected by ‘define_exposures()‘: the future penalized framework must carry the basis, penalty matrices, smoothing parameters, fitting engine, prediction design, and uncertainty contract as a coherent model. Diagnosing ‘"ps"‘ columns while fitting them later without their penalty would be misleading.

Value

An object of class ‘epiexposure_identifiability‘, a list containing:

- ‘identifiable‘: whether the combined rank design has full numerical rank at ‘tol‘; - ‘numerically_stable‘: whether it is full rank, has no near-zero-variance cross-basis columns, has a scaled condition number below ‘condition_severe‘, and contains more epidemics than rank-design columns; - ‘status‘: ‘"ok"‘, ‘"warning"‘, or ‘"problem"‘; - ‘overall‘: one-row summary of the combined design; - ‘by_variable‘: diagnostics for each exposure’s cross-basis block; - ‘pairwise_exposure_correlation‘: descriptive row-level Pearson and Spearman correlations between raw exposures; - ‘pairwise_crossbasis_correlation‘: maximum and mean absolute pairwise correlations between columns belonging to different exposure blocks; - ‘vif‘: supplementary column-wise VIF diagnostics; - ‘singular_values‘: singular values of the centered/scaled usable cross-basis predictor matrix; - ‘dependent_columns‘: pivot-based set of columns not needed to span a rank-deficient combined design; - ‘near_zero_variance_columns‘: flagged cross-basis columns; - ‘complete_epi_id‘: epidemic IDs represented in the design; - ‘basis_specification‘: effective basis metadata transported from ‘define_exposures()‘/‘build_design()‘; - ‘diagnostic_flags‘: named logical flags used to construct the status; - ‘recommendations‘: concise interpretation and follow-up suggestions; - ‘settings‘: normalized diagnostic settings and temporal metadata; - ‘design_matrix‘: included only when ‘keep_design = TRUE‘.

compare_exposures *Compare chronological exposure profiles*

Description

Compares two or more exposure profiles, simulated scenarios, or epidemic-specific exposure profiles while preserving the EpiExposure chronological profile convention.

Usage

```
compare_exposures(
  exposures = NULL,
  exposure1 = NULL,
  exposure2 = NULL,
  time = NULL,
  group = NULL,
  agg_fun = c("median", "mean"),
  mode = c("global", "timewise"),
  cor_method = c("pearson", "spearman", "kendall"),
  q = 0.8,
  eps = 1e-12
)
```

Arguments

exposures	Named list of numeric exposure profiles. Every element must represent the same exposure variable, in the same units, over the same chronological positions, and all profiles must have equal length. Profiles must be stored from the earliest/oldest observation to the most recent observation. A single numeric vector is still accepted for backward compatibility, but at least two profiles or aggregated groups are required for an actual comparison.
exposure1	Legacy numeric exposure vector. It must be supplied together with ‘exposure2‘ when ‘exposures = NULL‘.
exposure2	Legacy numeric exposure vector. It must be supplied together with ‘exposure1‘ when ‘exposures = NULL‘. If ‘exposures‘ is supplied, ‘exposure1‘ and ‘exposure2‘ must both be ‘NULL‘; legacy arguments are not silently ignored.
time	Optional numeric vector of chronological time positions. It must have the same length as every exposure profile, contain unique finite values, be strictly increasing, and be equally spaced. Equal spacing is required because EpiExposure profiles represent complete regular exposure profiles and several global metrics treat each temporal position equally. If ‘NULL‘, ‘0:(n - 1)‘ is used.

`group` Optional group membership for the supplied profiles. When provided, profiles are first aggregated within each group at every time position using ‘agg_fun‘, and pairwise comparisons are made between the resulting group-level profiles.

An unnamed vector is matched by profile position. A named vector is safer: its names must match ‘names(exposures)‘ exactly and are used to align group membership to profiles before aggregation.

`agg_fun` Character. ‘"median"‘ or ‘"mean"‘. Used only when ‘group‘ is supplied. The default is ‘"median"‘.

`mode` Character. ‘"global"‘ or ‘"timewise"‘.

‘"global"‘ Returns one row per pairwise profile comparison with distance, magnitude, correlation, peak-timing, center-of-mass, and high-exposure overlap summaries.

‘"timewise"‘ Returns one row per chronological position and pairwise comparison, retaining the complete exposure profiles.

`cor_method` Character correlation method used in ‘mode = "global"‘. Available options are:

- ‘"pearson"‘ (default): linear association between the original exposure values; - ‘"spearman"‘: monotonic association calculated from exposure ranks; - ‘"kendall"‘: Kendall’s rank correlation (tau), also measuring monotonic association and often useful for small samples or many non-linear monotonic relationships.

The selected method changes only the ‘corr‘ summary. All distance, magnitude, timing, center-of-mass, and overlap metrics are unchanged.

Correlation is returned as ‘NA‘ when either profile is constant because none of the supported correlation coefficients is defined in that case.

`q` Finite numeric scalar between 0 and 1. Each profile’s own ‘q‘ quantile defines its high-exposure positions for the Jaccard-style ‘overlap_above_q‘ metric. The default is ‘0.8‘.

`eps` Positive finite numeric tolerance. It is used only to decide when a time-wise ratio denominator or a non-negative profile’s total center-of-mass weight is numerically indistinguishable from zero.

‘eps‘ is **not added to exposure values** and therefore never modifies a profile before comparison.

Details

Each profile must be ordered from the **oldest/earliest** exposure observation to the **most recent** observation. Consequently, for a profile of length ‘n‘, the final element corresponds to retrospective lag 0 and the first element corresponds to positional lag ‘n - 1‘.

‘compare_exposures()‘ is descriptive. It compares exposure profiles themselves and does not estimate DLNM effects, disease risks, or model predictions.

Comparison direction

Pairwise differences are directional:

$$difference = exposure1 - exposure2.$$

Therefore positive ‘diff‘, ‘diff_sum‘, or ‘diff_mean‘ values indicate larger exposure in the first named profile than in the second.

‘timing_shift_peak‘ and ‘center_of_mass_shift‘ follow the same direction:

$$time_1 - time_2.$$

Positive values therefore mean that the corresponding timing summary occurs later in ‘exposure1‘.

Discrete distance metrics

For paired profile values x_{1t} and x_{2t},

$$L1 = \sum_t |x_{1t} - x_{2t}|,$$

$$L2 = \sqrt{\sum_t (x_{1t} - x_{2t})^2},$$

and

$$Linf = \max_t |x_{1t} - x_{2t}|.$$

‘L1‘ and ‘L2‘ increase with the number of represented time positions. ‘MAE‘ and ‘RMSE‘ are therefore also returned when comparisons across equal-unit profiles of different lengths are needed.

These distances retain the units/scaling of the exposure profile and should only be compared across profiles of the same exposure variable and units.

Time-wise ratios

Earlier EpiExposure code calculated

$$x_1/(x_2 + eps),$$

which changed every denominator, including perfectly valid non-zero values. The current implementation never alters observed exposure values. It returns

$$x_1/x_2$$

only when ‘abs(x2) > eps‘; otherwise the ratio is ‘NA‘.

A numerical ratio is not automatically scientifically meaningful. It should only be interpreted as a relative exposure contrast when the exposure is measured on a ratio scale with a meaningful zero. For example, ratios of temperatures expressed in degrees Celsius generally should not be interpreted as relative temperature effects.

Correlation method

‘cor_method‘ controls only the global ‘corr‘ column:

- Pearson measures linear co-variation of the original profile values; - Spearman is Pearson correlation of the ranks and measures monotonic association; - Kendall returns Kendall’s tau, based on concordant and discordant pairs.

Correlation describes similarity in temporal pattern, not similarity in absolute exposure magnitude. For example, two profiles can have correlation close to 1 while one is systematically much larger than the other. Distance and magnitude metrics should therefore be interpreted alongside ‘corr‘.

No correlation is causal and no p-value is computed by this descriptive function.

Peak timing and ties

‘peak_time1‘ and ‘peak_time2‘ retain the first chronological occurrence of the maximum for backward compatibility. The function additionally returns first and last peak times, the midpoint of their temporal range, the number of maximum positions, and indicators of tied maxima. Thus a plateau or repeated equal maximum is never hidden by ‘which.max()‘.

‘timing_shift_peak‘ uses the first peak times for backward compatibility. ‘timing_shift_peak_midpoint‘ compares the midpoints of the full peak-time ranges and is usually more informative when maxima are tied.

Temporal center of mass

For a non-negative profile with positive total exposure, the temporal center of mass is

$$COM = \frac{\sum_t time_t x_t}{\sum_t x_t}.$$

Earlier versions silently replaced negative exposure values by zero before computing COM. That changes the supplied profile and is no longer done.

If a profile contains any negative value, or its non-negative total mass is not greater than ‘eps‘, its COM is returned as ‘NA‘ and ‘com_defined‘ is ‘FALSE‘. Even for non-negative data, COM is scientifically interpretable only when zero and exposure magnitude have a meaningful weighting interpretation.

High-exposure overlap

Each profile obtains its own threshold:

$$Q_j(q) = quantile(x_j, q).$$

High-exposure positions satisfy $x_{jt} >= Q_j(q)$. The returned ‘overlap_above_q‘ is the Jaccard index

$$\frac{|H_1 \cap H_2|}{|H_1 \cup H_2|}.$$

This measures whether the two profiles experience their **own relatively high exposure** at the same times. It is not an overlap above one common absolute exposure threshold. Ties at the quantile threshold can make the number of high positions larger than exactly ‘(1 - q) * n‘; counts and thresholds are returned explicitly.

Group aggregation

With ‘group‘, aggregation occurs independently at every chronological position before any pairwise metric is calculated. The resulting comparison therefore describes representative group-level profiles, not a distribution of pairwise epidemic-level differences. No uncertainty interval or inferential test is produced by this function.

Value

A data frame.

With ‘mode = "timewise"‘, the output contains:

- ‘exposure1‘, ‘exposure2‘: compared profile/group names; - ‘position‘: chronological position ‘1, ..., n‘; - ‘time‘: chronological time supplied by the user; - ‘lag‘: positional retrospective lag, with lag 0 at the most recent observation; - ‘value1‘, ‘value2‘; - ‘diff = value1 - value2‘; - ‘abs_diff = abs(diff)‘; - ‘ratio = value1 / value2‘ when ‘abs(value2) > eps‘, otherwise ‘NA‘; - ‘ratio_defined‘: whether that numerical ratio was computed.

With ‘mode = "global"‘, the output contains one row per pair and includes:

- discrete distances ‘L1‘, ‘L2‘, and ‘Linf‘; - ‘MAE‘ and ‘RMSE‘, which are length-normalized profile discrepancies; - individual and between-profile sums, means, and SDs; - the selected Pearson, Spearman, or Kendall correlation when both profiles vary; - peak values and peak timing; - explicit information about tied maxima; - temporal centers of mass when the corresponding profile is non-negative and has positive total mass; - profile-specific high-exposure thresholds and Jaccard overlap.

Output attributes record chronological order, common time step, comparison direction, grouping information, and the ratio/center-of-mass contracts.

`compare_periods` *Compare period-specific DLNM effects*

Description

Compares period-specific DLNM associations returned by ‘summarise_effects(scale = "period")‘ against one selected reference period.

Usage

```
compare_periods(
  period_df,
  period_ref = 1,
  periods = NULL,
  output = c("summary", "samples"),
  interval_probs = NULL,
  eps = 1e-10
)
```

Arguments

`period_df`	Output from the current ‘summarise_effects(scale = "period")‘. A data frame is expected. If an ‘"epiexposure_effects"‘ object produced with diagnostics is supplied, its ‘$effects‘ component is used. The object must carry the current EpiExposure temporal metadata: one common fitted ‘max_lag‘, exact ‘history_length = max_lag + 1‘, and the exact-profile contract.

Current EpiExposure period output must contain ‘var‘, ‘value‘, ‘period‘, ‘scale‘, ‘eta‘, ‘effect‘, ‘baseline‘, ‘predicted‘, and ‘delta‘.

If ‘summarise_effects(..., uncertainty = TRUE)‘ was used, ‘period_df‘ must be the **sample-level** output produced with ‘output = "samples"‘. Comparing already-summarized medians and interval bounds would not preserve covariance among period effects.

`period_ref` Reference period. Either:

- one positive integer selecting the period by its defined order; or - one exact non-empty character period label, for example ‘"W1"‘.

Numeric references do not require period labels to end in numbers. If ‘periods‘ is supplied, its row order defines the numeric period order. Otherwise the first-occurrence order in ‘period_df$period‘ is used.

`periods` Optional period-definition data frame, typically returned by ‘define_periods()‘, containing ‘period‘, ‘lag_start‘, and ‘lag_end‘.

When supplied, period labels are validated against ‘period_df‘, every period boundary must lie within the inherited common fitted lag range ‘0:max_lag‘, and ‘lag_start‘, ‘lag_end‘, and ‘n_lags‘ are appended to the comparison output. The number of lags is

‘n_lags = lag_end - lag_start + 1‘.

Supplying ‘periods‘ is recommended when the compared periods have unequal widths because cumulative DLNM effects are sums across included lags.

`output` Character. ‘"summary"‘ or ‘"samples"‘.

For deterministic input (‘uncertainty = FALSE‘), only ‘"summary"‘ is available and contains the direct period comparisons.

For sample-level uncertain input, ‘"samples"‘ returns draw-by-draw comparisons and ‘"summary"‘ returns medians, SDs, and empirical intervals calculated **after** the draw-by-draw comparison.

`interval_probs` Optional numeric vector of length two defining the empirical interval for uncertain comparisons. If ‘NULL‘, the interval stored by ‘summarise_effects()‘ is reused; otherwise the default ‘c(0.025, 0.975)‘ is used when no stored interval is available.

`eps` Positive finite numerical tolerance. It is used only for consistency checks of values that should be equal under the EpiExposure contract; it is not used to alter effects or to force a denominator away from zero.

Details

The comparison preserves the EpiExposure effect contract:

- ‘eta‘ is the fundamental additive DLNM contrast on the fitted link scale; - ‘effect‘ is the user-requested representation of ‘eta‘ (‘"linear"‘, ‘"exponentiated"‘, or ‘"percent"‘); - ‘predicted‘ is the population-level expected outcome for that period; - ‘delta = predicted - baseline‘ is the absolute response-scale change from the joint exposure reference profile.

Period comparisons are always made at the same exposure variable and the same exposure evaluation value.

Exact common lag/profile contract

‘compare_periods()‘ does not receive the original long-format exposure histories, so it cannot recount time rows itself. Instead, it requires the temporal metadata propagated by the current ‘summarise_effects()‘ output:

$$profile_length = max_lag + 1.$$

The stored contract must state that all fitted exposure variables used the same exact lag window. This prevents period comparisons from silently mixing effect objects created under incompatible temporal histories.

When ‘periods‘ is supplied, its lag boundaries are additionally checked against the inherited common fitted ‘max_lag‘. ‘compare_periods()‘ does not require a manually supplied ‘periods‘ object to cover every fitted lag, because analysts may intentionally compare a subset of defined periods; it only requires every supplied period to remain inside the fitted lag window.

What is being compared?

‘summarise_effects(scale = "period")‘ first sums lag-specific DLNM contributions on the additive linear-predictor scale within each requested period. ‘compare_periods()‘ then compares those period-specific cumulative contrasts at the **same exposure value**.

If period j has contrast $\eta_j(x)$ and the selected reference period has contrast $\eta_r(x)$, the fundamental comparison is

$$\Delta\eta_{j:r}(x) = \eta_j(x) - \eta_r(x).$$

Under a log link,

$$\exp\{\Delta\eta_{j:r}(x)\} = RR_j(x)/RR_r(x),$$

while under a logit link,

$$\exp\{\Delta\eta_{j:r}(x)\} = OR_j(x)/OR_r(x).$$

Therefore the returned multiplicative ‘ratio‘ is derived from ‘eta_diff‘, not from blindly dividing whichever transformed ‘effect‘ happens to be displayed.

This distinction matters for ‘effect_measure = "percent"‘. For example, a +20 percent relative effect and a +10 percent relative effect do **not** imply a period ratio of ‘20 / 10 = 2‘. Their underlying multiplicative effects are 1.20 and 1.10, and the appropriate relative ratio is ‘1.20 / 1.10‘.

Response-scale comparison

‘predicted_diff‘ compares expected outcomes under the same joint reference profile while changing the focal exposure over different lag periods. For Beta and Binomial outcomes, multiplying ‘predicted_diff‘ by 100 gives the difference in percentage points between the two period-specific expected proportions/probabilities.

Since ‘delta = predicted - baseline‘ and the same baseline is used for every period comparison at a given parameter draw, ‘delta_diff‘ must equal ‘predicted_diff‘ up to numerical tolerance. The function validates this identity.

Unequal period widths

Period effects are cumulative sums over lags. Consequently, comparing a 21-lag period with a 10-lag period compares the cumulative associations as defined; the contrast can reflect both the lag-response pattern and the different number of included lags. ‘compare_periods()‘ does not divide cumulative effects by period width. Use period definitions with the scientific interpretation intended by the analysis.

Uncertainty

When ‘summarise_effects()‘ uses uncertainty, all requested effects are based on the same joint parameter draw. ‘compare_periods()‘ preserves this covariance by matching the target and reference periods **within the same draw**, calculating ‘eta_diff‘, ‘diff‘, ‘ratio‘, ‘predicted_diff‘, and ‘delta_diff‘ draw by draw, and only then summarizing those comparisons.

Already summarized uncertain output is therefore rejected explicitly. Subtracting medians or combining marginal interval endpoints would not recover the distribution of a difference or ratio.

Value

A data frame containing the original period-specific quantities and the corresponding reference-period quantities. Principal comparison columns are:

‘eta_diff‘ Difference in additive DLNM contrasts:

‘eta - ref_eta‘.

This is the canonical comparison on the fitted link scale.

‘diff‘ Arithmetic difference in the displayed ‘effect‘:

‘effect - ref_effect‘.

Its units depend on the ‘effect_measure‘ used by ‘summarise_effects()‘.

‘ratio‘ For ‘log‘ links, the ratio of the two period-specific response ratios. For ‘logit‘ links, the ratio of the two period-specific odds ratios:

‘exp(eta - ref_eta)‘.

For other links, no general multiplicative interpretation is defined, so ‘ratio‘ and ‘ratio_percent‘ are ‘NA‘.

‘ratio_percent‘ ‘100 * (ratio - 1)‘ when ‘ratio‘ is defined.

‘predicted_diff‘ Difference between the period-specific population expected responses:

‘predicted - ref_predicted‘.

‘delta_diff‘ Difference between period-specific absolute response-scale changes:

‘delta - ref_delta‘.

Because the joint ‘baseline‘ is identical across compared periods, ‘delta_diff‘ and ‘predicted_diff‘ should be numerically equal.

With uncertain sample-level input and ‘output = "summary"‘, each numeric quantity is summarized by its median plus ‘_sd‘, ‘_lower‘, and ‘_upper‘ columns.

Output attributes record the reference period, fitted link, effect measure, prediction contract, uncertainty contract, ratio interpretation, common fitted ‘max_lag‘, exact profile length, and the inherited EpiExposure exact-profile contract.

compare_predictions *Compare predicted outcomes between exposure scenarios*

Description

Compares population-level expected predictions from two or more explicit exposure scenarios using ‘predict_outcomes()‘ as the computational backend.

Usage

```
compare_predictions(
  fit,
  profiles = NULL,
  profiles1 = NULL,
  profiles2 = NULL,
  type = c("response", "link"),
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  probs = c(0.025, 0.975),
  seed = NULL,
  extrapolation = c("warn", "error", "allow"),
  eps = 1e-12
)
```

Arguments

fit	Fitted model returned by the current ‘fit_epidlnm()‘.
profiles	Named list containing at least two scenarios. Top-level names are scenario labels. Each scenario must contain the complete fitted exposure profile structure expected by ‘predict_outcomes()‘. For each fitted exposure, the scenario may supply: • one finite numeric scalar, interpreted as a constant profile and expanded internally to that exposure’s fitted ‘max_lag + 1‘ positions; • one explicit numeric chronological profile; • a matrix/data frame whose rows are multiple chronological profiles; • a list of multiple numeric chronological profiles; • an object returned directly by the audited ‘simulate_exposures()‘. For an audited ‘simulate_exposures()‘ object, all ‘n‘ simulated exposure profiles in its canonical ‘$profiles‘ collection are used directly. Thus, for example: “‘ scenario_A <- list(tmean = 25, rain = simulate_exposures(..., n = 100), wetness = simulate_exposures(..., n = 100)) “‘

creates 100 multivariable profiles: the constant temperature profile is recycled, while rain profile 1 is paired with wetness profile 1, rain profile 2 with wetness profile 2, and so forth.

Within a scenario, fitted exposures are paired by profile index exactly as in ‘predict_outcomes()‘. Across scenarios, the number of matched profile sets must be either one or a common maximum. A scenario containing one profile set is recycled across that common profile index when another scenario contains multiple matched profile sets.

Thus multiple profiles are compared by matched index; a Cartesian product of scenario profiles is never created.

Explicit non-scalar profile must contain exactly the fitted ‘max_lag + 1‘ positions for their exposure. A ‘simulate_exposures()‘ object must have been generated with the same ‘max_lag‘ stored for that fitted exposure. This strict check prevents extra or missing positions in the profile from being silently ignored.

`profiles1` Legacy first scenario profile. Used only with ‘profiles2‘ when ‘profiles = NULL‘.

`profiles2` Legacy second scenario profile. Used only with ‘profiles1‘ when ‘profiles = NULL‘.

`type` Prediction scale:

- ‘"response"‘ (default): population-level expected outcome on its natural response scale; - ‘"link"‘: population-level linear predictor.

‘diff = pred2 - pred1‘ is available on either scale. Relative metrics (‘ratio‘ and ‘percent_change‘) are calculated only for response-scale predictions.

`uncertainty` Logical. If ‘FALSE‘, comparisons use deterministic predictions from the central fixed/population parameter estimate. If ‘TRUE‘, comparisons are calculated draw by draw from joint parameter uncertainty.

`output` Character. ‘"summary"‘ or ‘"samples"‘.

With ‘uncertainty = FALSE‘, only ‘"summary"‘ is valid.

With ‘uncertainty = TRUE‘, ‘"samples"‘ returns draw-level pairwise comparisons and ‘"summary"‘ summarizes the **draw-level comparisons** by median, SD, and empirical interval.

`n_samples` Positive integer number of fixed/population parameter draws used when ‘uncertainty = TRUE‘. At least two are required.

`probs` Numeric vector of length two defining the lower and upper empirical uncertainty probabilities. The default ‘c(0.025, 0.975)‘ gives a 95 percent interval.

`seed` Optional finite integer seed passed once to the joint ‘predict_outcomes()‘ call. It controls **model-parameter draws only** and does not resimulate exposure profiles already produced by ‘simulate_exposures()‘. The prediction backend restores the caller’s random-number state after seeded sampling.

`extrapolation` Behavior when any scenario contains exposure values outside the fitted exposure range: ‘"warn"‘ (default), ‘"error"‘, or ‘"allow"‘. The fitted DLNM basis is never re-estimated from scenario data.

`eps` Positive finite tolerance used only to decide whether a response-scale reference prediction is sufficiently above zero for a relative comparison.

No epsilon is added to a prediction. When ‘pred1 <= eps‘, ‘ratio‘ and ‘percent_change‘ are returned as ‘NA‘ rather than modifying the denominator.

Details

All scenarios are submitted to ‘predict_outcomes()‘ in **one joint batch**. This is essential when ‘uncertainty = TRUE‘: the same fixed/population parameter draw is applied to every scenario before differences, ratios, or other contrasts are calculated.

Exposure profiles follow the EpiExposure chronological convention: each profile is supplied from the oldest/earliest exposure observation to the most recent observation, with the final value corresponding to lag 0.

EpiExposure v1 prediction target

‘compare_predictions()‘ compares the same prediction estimand used throughout the current package:

$$E(Y \mid X, \theta)$$

using the fixed/population component with fitted random effects excluded. Models may have been fitted with random effects, but this function does not provide conditional/group-specific prediction.

For deterministic prediction, each scenario is evaluated with the central parameter estimate:

$$\hat{\mu}_j = g^{-1}(X_j \hat{\beta})$$

for ‘type = "response"‘, or $X_j \hat{\beta}$ for ‘type = "link"‘.

No residual, observation, process, dispersion, or posterior-predictive noise is added.

Pairwise direction

For every ordered pair generated from the input scenario order,

$$diff = pred_2 - pred_1.$$

A positive difference therefore means that scenario 2 has the larger predicted expected outcome on the selected prediction scale.

Relative response-scale contrasts

When ‘type = "response"‘ and ‘pred1 > eps‘,

$$ratio = pred_2 / pred_1$$

and

$$percent_change = 100(ratio - 1).$$

Earlier EpiExposure code used ‘pred2 / (pred1 + eps)‘ and divided differences by ‘abs(pred1) + eps‘. Those formulas changed valid predictions and gave unusual behavior for non-positive denominators. The current implementation never modifies predicted values.

A numerical response ratio is scientifically interpretable as a relative outcome change only when the outcome scale has an appropriate meaningful zero. This is usually natural for expected counts, positive means, and probabilities, but may not be meaningful for every Gaussian response.

With ‘type = "link"‘, ‘ratio‘ and ‘percent_change‘ are intentionally ‘NA‘. Dividing two linear predictors generally does not define a meaningful model contrast. The valid link-scale contrast is ‘diff‘.

Probability and proportion outcomes

For Beta and Binomial response-scale predictions,

$$percentage_point_change = 100(pred_2 - pred_1).$$

This is distinct from ‘percent_change‘. For example, a probability increase from 0.20 to 0.30 is +10 percentage points but +50 percent relative to 0.20.

Joint uncertainty across scenarios

A central requirement of scenario comparison is that scenario contrasts use the same parameter draw. For draw s,

$$diff^{(s)} = \mu_2^{(s)} - \mu_1^{(s)}.$$

‘compare_predictions()‘ therefore constructs one combined prediction batch containing every scenario and calls ‘predict_outcomes()‘ only once. The prediction backend applies one joint $\beta^{(s)}$ to every row in that batch.

The function then matches scenario 1 and scenario 2 within the same ‘profile‘ and ‘sample‘, calculates all contrasts draw by draw, and only then summarizes them. This preserves covariance among scenario predictions.

Calling ‘predict_outcomes()‘ independently for each scenario and pairing rows with the same integer sample label would not establish that those rows came from the same parameter draw and is therefore not used.

Direct ‘simulate_exposures()‘ integration

An audited ‘simulate_exposures()‘ object contains the canonical ‘$profiles‘ collection and metadata describing chronological order and ‘max_lag‘. ‘compare_predictions()‘ recognizes this object directly and uses all ‘n‘ simulated exposure profiles.

For example:

“‘ scenario_A <- list(tmean = simulate_exposures(max_lag = 85, n = 100, mode = "profile", background = list(dist = "normal", mean = 25, sd = 4)), rain = 5, wetness = simulate_exposures(max_lag = 85, n = 100, mode = "profile", background = list(dist = "normal", mean = 10, sd = 2))) “‘

The 100 tmean and wetness profiles are paired by index; the constant rain profile is recycled across them.

‘n‘ in ‘simulate_exposures()‘ controls exposure-profile variability only. Model-parameter uncertainty requested through ‘uncertainty = TRUE‘ in this function is a separate layer. Thus 100 simulated exposure profiles and 1000 coefficient draws represent different sources of variation.

The direct integration also supports the immediately preceding ‘simulate_exposures()‘ return structure for backward compatibility, provided its metadata explicitly identify chronological profile order. New package code should use the standardized audited object.

Multiple profile sets within scenarios

If all scenarios contain one profile set, the output has one row per scenario pair.

If at least one scenario contains multiple matched profile sets, comparisons are made by profile index. A scenario with one profile set is recycled to the common number of profile sets; any other non-singleton scenario must contain that same common number.

Value

A data frame containing pairwise scenario comparisons.

Deterministic output contains:

- ‘scenario1‘, ‘scenario2‘; - ‘profile‘ when more than one matched profile set is compared; - ‘pred1‘, ‘pred2‘; - ‘diff = pred2 - pred1‘; - ‘abs_diff = abs(diff)‘; - ‘ratio = pred2 / pred1‘ when defined on the response scale; - ‘percent_change = 100 * (ratio - 1)‘ when defined; - ‘relative_change_defined‘; - ‘percentage_point_change = 100 * diff‘ for Beta and Binomial response-scale predictions, otherwise ‘NA‘.

With ‘uncertainty = TRUE‘ and ‘output = "samples"‘, the same quantities are returned for every matched parameter draw, together with ‘sample‘.

With ‘uncertainty = TRUE‘ and ‘output = "summary"‘, ‘pred1‘, ‘pred2‘, ‘diff‘, ‘abs_diff‘, and any applicable relative/probability contrasts are summarized by their median plus ‘_sd‘, ‘_lower‘, and ‘_upper‘ columns.

For relative metrics, ‘relative_defined_fraction‘ gives the fraction of draws in which ‘pred1 > eps‘. A posterior relative summary is returned only when the relative metric is defined for **all** matched draws; otherwise its median and interval are ‘NA‘ rather than being calculated from a truncated subset of draws.

Output attributes record the population expected-response prediction contract, family, link, prediction scale, scenario order, profile-input sources, uncertainty contract, and comparison direction.

`compute_eci` *Compute Exposure Cumulative Impact*

Description

Computes a descriptive raw cumulative exposure (‘ECI_raw‘) and, when a fitted EpiExposure model is supplied, a model-weighted cumulative exposure impact (‘ECI_weighted‘) for one or more exposure variables.

Usage

```
compute_eci(
  profile = NULL,
  data = NULL,
  group = NULL,
  time = "time",
  group_level = NULL,
  fit = NULL,
  vars = NULL,
  ref = list(method = "median", value = NULL),
```

```
  scale = c("link", "response", "percent"),
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  interval_probs = c(0.025, 0.975),
  seed = NULL,
  extrapolation = c("error", "warn", "allow")
)
```

Arguments

profile	Optional exposure profile input. Supply exactly one of 'profile' or 'data'. When one exposure variable is evaluated, 'profile' may be a finite numeric vector. For multiple variables, supply a list with one numeric vector per requested variable. A named list is recommended. If the list is unnamed, its order must match 'vars'. Profiles must be chronological from oldest to most recent. When 'fit' is supplied, every focal exposure profile must contain **exactly** 'max_lag + 1' observations for that fitted exposure. Profiles that are shorter or longer are rejected; 'compute_eci()' never selects a temporal window silently from a longer profile. When 'fit = NULL', no fitted 'max_lag' exists. In that raw-only mode, 'profile' must be one finite numeric vector and the function simply returns its descriptive cumulative sum.
data	Optional non-empty long-format data frame containing observed chronological exposure profiles. Supply exactly one of 'profile' or 'data'. 'data' must contain the grouping column named by 'group', the time column named by 'time', and every exposure requested in 'vars'. Every evaluated group must contain exactly 'max_lag + 1' equally spaced observations for the requested focal exposure(s). Groups with either fewer or more rows are rejected. If a method-based reference is used, 'data' must additionally contain every exposure in the fitted model because those variables are needed to construct the joint reference profile.
group	Character scalar naming the independent-profile grouping column in 'data'. Required when 'data' is supplied.
time	Character scalar naming the chronological numeric time column in 'data'. Default is '"time"'. Times must be unique and equally spaced within each evaluated group, and the temporal spacing must be common across groups.
group_level	Optional grouping value or vector of values to evaluate. If 'NULL', all group levels are evaluated in first-occurrence order.
fit	Optional fitted model returned by the current 'fit_epidlnm()'. If 'NULL', only 'ECI_raw' is calculated and model-weighted ECI, uncertainty, and response-scale quantities are unavailable.
vars	Optional character vector naming fitted exposure variables to evaluate. If 'fit' contains one exposure and 'vars = NULL', that exposure is used. For a multivariable fit, 'vars' must be supplied explicitly. Each requested variable is evaluated separately against the same joint reference profile; the function does not sum impacts from different exposure variables.

`ref` Reference exposure specification used when ‘fit‘ is supplied.

Two forms are supported:

- Method-based: ‘list(method = "median", value = NULL)‘, ‘list(method = "percentile", value = p)‘, or ‘list(method = "fixed", value = x)‘.
- Exposure-specific: a named list containing exactly one finite numeric reference value for **every fitted exposure**, for example ‘list(tmean = 25, rain = 5, wetness = 10)‘.

Method-based references require ‘data‘ because the reference values are derived from the supplied exposure data. ‘"median"‘ uses each fitted exposure’s own median. ‘"percentile"‘ uses the same percentile ‘p‘ for every fitted exposure. ‘"fixed"‘ applies the same numeric value to all fitted exposures and should be used cautiously when variables have different units.

For direct ‘profile‘ input, use the exposure-specific named-list form.

The default is ‘list(method = "median", value = NULL)‘.

`scale` Character defining the interpretation and units of the single returned model-weighted impact column, ‘ECI_weighted‘:

- ‘"link"‘: the centered cumulative DLNM contrast $\Delta\eta$; - ‘"response"‘: the absolute response-scale impact $P - B$, where ‘P‘ is the population expected response for the focal profile and ‘B‘ is the population expected response for the joint reference profile; - ‘"percent"‘: ‘100 * (exp(Delta eta) - 1)‘, available only for log and logit links.

Internally, ‘compute_eci()‘ calculates both the centered link-scale contrast $\Delta\eta$ and the response-scale difference $\Delta = P - B$. These internal quantities are intentionally **not returned as separate ‘eta‘ and ‘delta‘ columns** because they duplicate ‘ECI_weighted‘ on the corresponding scale: with ‘scale = "link"‘, ‘ECI_weighted = Delta eta‘; with ‘scale = "response"‘, ‘ECI_weighted = P - B‘. The internal quantities are still retained during computation because they are required for scale transformations and draw-by-draw uncertainty propagation.

For a log link, ‘"percent"‘ is the percent relative change in the expected response. For a logit link, it is the percent change in odds, not the percentage-point change in probability.

`uncertainty` Logical. If ‘FALSE‘, use the harmonized central fixed/population parameter estimate. If ‘TRUE‘, propagate joint fixed/population parameter uncertainty draw by draw.

`output` Character. ‘"summary"‘ returns deterministic values when ‘uncertainty = FALSE‘, or medians, SDs, and empirical intervals when ‘uncertainty = TRUE‘. ‘"samples"‘ returns one row per parameter draw and requires ‘uncertainty = TRUE‘.

`n_samples` Positive integer number of parameter draws used when ‘uncertainty = TRUE‘. At least two are required. Default is 1000.

`interval_probs` Numeric vector of length two defining the empirical uncertainty interval. Default ‘c(0.025, 0.975)‘ gives a 95 percent interval.

`seed` Optional finite integer used for parameter-draw sampling. The caller’s global random-number state is restored when the function exits.

`extrapolation` Character controlling exposure values outside the fitted cross-basis exposure range: ‘"error"‘ (default), ‘"warn"‘, or ‘"allow"‘. The fitted basis is never re-estimated from ECI input data.

Details

Exposure profiles are interpreted chronologically, from the oldest observation to the most recent observation. The most recent value is associated internally with lag 0.

The model-weighted ECI is a **contrast relative to an explicit joint exposure reference profile**. It is not the uncentered cross-basis contribution ‘cb(x) is not, by itself, an interpretable exposure effect.

Raw ECI

For the discrete exposure profile used by the fitted lag window,

$$ECI_{raw} = \sum_{j=0}^{L} x_j.$$

This is a discrete cumulative exposure sum, not a continuous-time integral. It retains the units of the exposure multiplied by the number of discrete lag positions. Comparisons are meaningful only for the same exposure variable, units, lag window, and temporal resolution.

The centered descriptive counterpart is

$$ECI_{raw,centered} = \sum_{j=0}^{L} (x_j - x_{ref}).$$

Neither raw quantity incorporates the fitted exposure-response or lag-response shape.

Exact exposure-profile length

When a fitted model is supplied, ‘compute_eci()‘ uses the EpiExposure exact temporal-profile contract. For a fitted maximum lag ‘L‘, every evaluated focal exposure profile must contain exactly

$$L + 1$$

chronological observations: the first value represents lag ‘L‘ and the last value represents lag 0. All fitted exposures share the same ‘max_lag‘. Longer profiles are not truncated with ‘tail()‘ and shorter profiles are not padded. This prevents ambiguity about which exposure window generated the cumulative impact and keeps ECI calculations comparable across profiles, groups, and downstream lag decompositions.

In raw-only mode (‘fit = NULL‘), this restriction cannot be inferred because no fitted ‘max_lag‘ is available; the supplied vector is summed as given.

Centered weighted ECI

A DLNM cross-basis is a basis expansion. The uncentered quantity

$$cb(x)^T \beta$$

is a contribution to the linear predictor under the chosen basis parameterization, but it is not an invariant exposure effect. In particular, applying the inverse link to this contribution alone does not produce a valid expected response because the model intercept and the reference contribution of the other fitted exposures are absent.

EpiExposure therefore defines the model-weighted cumulative impact as a contrast between the focal profile and an explicit joint reference profile:

$$\Delta\eta = \{cb(x) - cb(x_{ref})\}^T \beta_{focal}.$$

Every non-focal fitted exposure is held at its own reference value throughout its complete lag window.

This centering makes ‘ECI_weighted‘ consistent with the DLNM contrast logic used by ‘summarise_effects()‘.

Why only ‘ECI_weighted‘ is returned

Three mathematical quantities are required internally:

$$\Delta\eta = \eta_{target} - \eta_{reference},$$

$$B = g^{-1}(\eta_{reference}), \qquad P = g^{-1}(\eta_{target}),$$

and

$$\Delta = P - B.$$

Earlier output designs could expose ‘eta‘, ‘delta‘, and ‘ECI_weighted‘ simultaneously. This is redundant because the selected scale already defines which model-weighted impact is being reported. The current output therefore uses one canonical column:

- ‘scale = "link"‘: ‘ECI_weighted‘ is $\Delta\eta$;
- ‘scale = "response"‘: ‘ECI_weighted‘ is $\Delta = P - B$;
- ‘scale = "percent"‘: ‘ECI_weighted‘ is the permitted percent transformation of $\Delta\eta$.

‘baseline‘ and ‘predicted‘ remain in the output because they are distinct expected-response quantities that provide the response-scale context for the contrast. The intermediate ‘eta‘ and ‘delta‘ quantities continue to be computed internally, including for every uncertainty draw, but are not duplicated as public output columns.

Response-scale ECI

Let X_{ref} denote the complete fixed/population design under the joint reference exposure profile. The population expected-response baseline is

$$B = g^{-1}(X_{ref}\beta).$$

Let X_{target} replace only the focal exposure profile by the evaluated profile. Then

$$P = g^{-1}(X_{target}\beta)$$

and

$$\Delta = P - B.$$

With ‘scale = "response"‘, ‘ECI_weighted = Delta‘.

Percent scale

With a log link,

$$100\{\exp(\Delta\eta) - 1\}$$

is the percent relative change in the population expected response.

With a logit link, the same mathematical transformation is the percent change in odds. It is not the percent change or percentage-point change in predicted probability.

Other links do not receive this exponential interpretation, so ‘scale = "percent"‘ is rejected explicitly.

Population-level prediction contract

Weighted ECI follows the same EpiExposure v1 prediction target as ‘predict_outcomes()‘ and ‘summarise_effects()‘:

$$E(Y \mid X, \theta)$$

using only the fixed/population component. Fitted group-specific random effects are set to zero/excluded. No residual, observation, process, dispersion, or posterior-predictive noise is added.

With ‘uncertainty = FALSE‘, frequentist fits use fitted fixed coefficients and Bayesian fits use posterior-mean fixed/population coefficients.

Uncertainty

With ‘uncertainty = TRUE‘, one coherent fixed/population parameter draw is applied jointly to the reference design and every target ECI design. For draw s,

$$\Delta\eta_i^{(s)} = (X_i - X_{ref})\beta^{(s)},$$

$$B^{(s)} = g^{-1}(X_{ref}\beta^{(s)}),$$

$$P_i^{(s)} = g^{-1}(X_i\beta^{(s)}),$$

and

$$\Delta_i^{(s)} = P_i^{(s)} - B^{(s)}.$$

The requested ‘ECI_weighted‘ transformation is then applied **within each draw** before medians, SDs, and empirical quantiles are calculated. The same parameter draw determines the baseline and every target row, preserving covariance among variables, groups, baseline, target predictions, and the weighted ECI.

Frequentist engines use the joint asymptotic fixed-effect covariance matrix. Bayesian engines use joint posterior or approximate-posterior fixed-effect draws through the centralized EpiExposure prediction helpers. The function does not independently reconstruct engine-specific uncertainty and does not add residual or posterior-predictive outcome noise.

Method-based reference values

Method-based references are resolved once from the complete supplied ‘data‘, not independently within each group. This ensures every evaluated group is compared with the same joint exposure reference condition.

Value

A data frame.

With a fitted model, principal columns are:

‘var‘ Focal exposure variable.

‘reference_value‘ Reference exposure value for the focal variable.

‘max_lag‘ Common fitted maximum lag.

‘n_exposure_values‘ Number of exposure values used; always ‘max_lag + 1‘.

‘ECI_raw‘ Descriptive sum of the focal exposure values over the fitted lag window.

‘ECI_raw_centered‘ Descriptive sum of exposure deviations from the focal reference value over the same lag window.

‘baseline‘ Population-level expected response under the joint reference exposure profile, with fitted random effects excluded.

‘predicted‘ Population-level expected response when the focal exposure follows the evaluated profile and every other fitted exposure remains at its reference value.

‘ECI_weighted‘ The model-weighted exposure impact on the scale requested by ‘scale‘. It equals the internally calculated $\Delta\eta$ on the link scale, the internally calculated $P - B$ on the response scale, or the permitted exponential percent transformation on the percent scale.

‘scale‘ Requested weighted-ECI scale.

‘eta‘ and ‘delta‘ are not returned as separate columns. Their information is represented by ‘ECI_weighted‘ when ‘scale = "link"‘ and ‘scale = "response"‘, respectively.

With ‘uncertainty = TRUE‘ and ‘output = "summary"‘, ‘baseline‘, ‘predicted‘, and ‘ECI_weighted‘ are each summarized by their median plus ‘_sd‘, ‘_lower‘, and ‘_upper‘ columns. ‘ECI_raw‘ and ‘ECI_raw_centered‘ are exposure-profile descriptors and therefore remain deterministic.

With ‘uncertainty = TRUE‘ and ‘output = "samples"‘, ‘baseline‘, ‘predicted‘, and ‘ECI_weighted‘ are returned draw by draw with a ‘sample‘ column. The internal link-scale and response-scale contrasts are calculated for every draw but are not duplicated as output columns.

When ‘fit = NULL‘, the result contains only ‘ECI_raw‘ and the number of exposure values used.

The output attribute ‘epiexposure_eci_history_contract‘ records ‘"exact_max_lag_plus_one"‘ whenever a fitted model is used.

compute_ecilag *Compute lag-specific decomposition of Exposure Cumulative Impact (ECI)*

Description

Decomposes the model-weighted Exposure Cumulative Impact (ECI) of one or more exposure profiles into exact lag-specific contributions on the linear-predictor scale.

Usage

```
compute_ecilag(
  profile = NULL,
  data = NULL,
  group = NULL,
  time = "time",
  group_level = NULL,
  fit,
  vars = NULL,
  ref = list(method = "median", value = NULL),
  absolute = TRUE,
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  interval_probs = c(0.025, 0.975),
  seed = NULL,
  extrapolation = c("error", "warn", "allow")
)
```

Arguments

profile	Optional exposure-profile input. Supply exactly one of ‘profile‘ or ‘data‘. For a single focal exposure, ‘profile‘ may be one finite numeric vector. For multiple focal exposures, supply a list containing one numeric vector per variable. A named list is recommended; if unnamed, its order must match ‘vars‘. Exposure profiles must be chronological from the oldest observation to the most recent observation. Under the EpiExposure exact-profile contract, every supplied profile must contain exactly ‘max_lag + 1‘ observations. The first value corresponds to lag ‘max_lag‘ and the final value to lag 0. Profiles are never truncated, padded, or silently realigned.
data	Optional non-empty long-format data frame containing observed exposure profiles. Supply exactly one of ‘profile‘ or ‘data‘. ‘data‘ must contain ‘group‘, ‘time‘, and every exposure requested in ‘vars‘. Each evaluated group must contain exactly ‘max_lag + 1‘ observations, with unique and equally spaced time values. All evaluated groups must use the same temporal spacing.

When a method-based reference is requested, ‘data‘ must additionally contain every exposure included in the fitted model because the complete joint reference profile is constructed from those variables.

`group` Character scalar naming the independent-profile grouping column in ‘data‘. Required when ‘data‘ is supplied.

`time` Character scalar naming the chronological numeric time column in ‘data‘. Default is ‘"time"‘.

`group_level` Optional grouping value or vector of grouping values to evaluate. If ‘NULL‘, all groups are evaluated in first-occurrence order.

`fit` Fitted EpiExposure model returned by the current ‘fit_epidlnm()‘ implementation. The fitted model must satisfy the current population-level expected-response, central-parameter, uncertainty, and exact-profile metadata contracts.

`vars` Optional character vector naming fitted exposure variables to decompose.

If the fitted model contains only one exposure, ‘vars = NULL‘ uses that exposure automatically. For multivariable fits, ‘vars‘ must be supplied explicitly.

Each requested variable is decomposed separately. When one focal variable is evaluated, all other fitted exposures remain fixed at their joint reference values; contributions from different exposure variables are not pooled into a single lag decomposition.

`ref` Reference exposure specification.

Two forms are supported:

- Method-based: ‘list(method = "median", value = NULL)‘, ‘list(method = "percentile", value = p)‘, or ‘list(method = "fixed", value = x)‘.
- Exposure-specific: a named list containing exactly one finite reference value for every fitted exposure, for example ‘list(tmean = 25, rain = 5, wetness = 70)‘.

Method-based references require ‘data‘. ‘"median"‘ uses each fitted exposure’s median, ‘"percentile"‘ applies the same percentile probability to each fitted exposure, and ‘"fixed"‘ applies the same numeric value to every fitted exposure and should therefore be used cautiously when exposures have different units.

The default is ‘list(method = "median", value = NULL)‘.

`absolute` Logical. If ‘TRUE‘, add the absolute lag contribution and, when uncertainty is requested, its uncertainty summary. The signed ‘ECI_weighted‘ column is always retained.

‘ECI_percent‘ is always based on the absolute magnitude of each lag contribution:

$$100\frac{|C_l|}{\sum_j |C_j|}.$$

Consequently, positive and negative lag effects do not cancel when relative lag importance is calculated. If the total absolute contribution is zero, percentage contributions are undefined and returned as ‘NA‘.

`uncertainty` Logical. If ‘FALSE‘, lag contributions are calculated from the harmonized central fixed/population parameter estimate. If ‘TRUE‘, joint parameter uncertainty is propagated draw by draw through the complete lag decomposition.

`output`	Character. ‘"summary"‘ returns deterministic lag contributions when ‘uncertainty = FALSE‘, or medians, standard deviations, and empirical intervals when ‘uncertainty = TRUE‘. ‘"samples"‘ additionally returns the lag-specific and total weighted ECI values for every parameter draw and requires ‘uncertainty = TRUE‘.
`n_samples`	Positive integer number of parameter draws used when ‘uncertainty = TRUE‘. At least two are required. Default is 1000.
`interval_probs`	Numeric vector of length two specifying the empirical uncertainty interval. The default ‘c(0.025, 0.975)‘ gives a 95 percent interval.
`seed`	Optional finite integer used for parameter-draw generation or sampling. When supplied, the caller’s random-number state is restored when the function exits.
`extrapolation`	Character controlling exposure values outside the range recorded by the fitted cross-basis: ‘"error"‘ (default), ‘"warn"‘, or ‘"allow"‘. The fitted basis specification is never re-estimated from the ECI-lag input data.

Details

Each evaluated exposure profile is compared with an explicit joint reference profile. The focal exposure is allowed to vary through time while every other fitted exposure remains fixed at its reference value. Lag-specific contributions are obtained by changing one chronological exposure position at a time relative to that same reference profile and reconstructing the corresponding fitted cross-basis through the standard EpiExposure prediction pipeline.

The resulting lag contributions are additive: their sum is required to equal the centered cumulative ECI for the complete focal exposure trajectory. ‘compute_ecilag()‘ therefore performs an exact decomposition of the fitted DLNM contrast. It does not calculate numerical derivatives, marginal effects, or local sensitivity measures.

Centered cumulative impact

Let X_{ref} denote the fixed/population design generated when every fitted exposure follows its reference profile. For focal exposure v, let X_v denote the design in which only that exposure follows the evaluated complete trajectory. The model-weighted cumulative ECI is

$$ECI_{weighted,v} = (X_v - X_{ref})\beta.$$

This quantity is a centered contrast on the linear-predictor scale. The intercept and all non-focal exposure terms cancel because they are identical in the target and reference designs.

Exact lag decomposition

For each chronological position corresponding to lag l, the function creates an isolated target profile in which only that exposure value differs from the reference trajectory. The resulting design contrast is

$$D_l = X_l - X_{ref}.$$

Its lag-specific contribution is

$$C_l = D_l\beta.$$

The implementation explicitly verifies

$$\sum_{l=0}^{L} D_l = X_v - X_{ref}$$

and therefore

$$\sum_{l=0}^{L} C_l = ECI_{weighted,v}.$$

If this additivity condition is not satisfied within numerical tolerance, the function stops rather than returning an approximate decomposition.

This is fundamentally different from a derivative-based sensitivity analysis. ‘compute_ecilag()‘ assigns the complete centered fitted contrast to exact lag positions; it does not estimate $\partial\eta/\partial x_l$.

Chronological and lag indexing

Input profiles are chronological. For fitted maximum lag L, a profile of length $L + 1$ is interpreted as

$$(x_L, x_{L-1}, \ldots, x_1, x_0),$$

where the first supplied observation corresponds to lag L and the last to lag 0. Returned ‘by_lag‘ tables are ordered by increasing lag.

Joint reference profile

Even when only one exposure is decomposed, the reference condition is defined jointly for all fitted exposures. This keeps the decomposition consistent with the fitted multivariable model. Non-focal exposures remain fixed at their own reference values throughout the entire lag window.

Parameter uncertainty

With ‘uncertainty = TRUE‘, the same joint fixed/population parameter draw is used for all lag contributions within a trajectory. For draw s,

$$C_l^{(s)} = D_l \beta^{(s)}$$

and

$$ECI_{weighted}^{(s)} = \sum_l C_l^{(s)}.$$

Additivity is verified for every draw. Summary statistics are calculated only after the draw-specific contributions and percentage contributions have been obtained.

Frequentist engines use joint draws derived from the fitted fixed-effect covariance matrix. Bayesian engines use joint posterior or approximate-posterior fixed/population coefficient draws through the centralized EpiExposure prediction helpers.

No residual, observation, process, dispersion, or posterior-predictive noise is added. Uncertainty therefore represents parameter uncertainty in the fitted exposure-lag association.

Value

A list containing cumulative and lag-specific ECI results.

For a single exposure profile, the principal components are:

‘ECI_raw‘ Descriptive sum of the focal exposure profile.

‘ECI_raw_centered‘ Descriptive sum of exposure deviations from the focal reference value.

‘ECI_weighted‘ Centered cumulative exposure contribution on the linear-predictor scale.

‘reference_value‘ Reference value for the focal exposure.

‘max_lag‘ Common fitted maximum lag.

‘n_exposure_values‘ Number of observations in the profile; always ‘max_lag + 1‘.

‘by_lag‘ Data frame containing the exact contribution assigned to each lag.

With ‘uncertainty = TRUE‘, ‘ECI_weighted‘ is summarized by its median, standard deviation, and empirical lower and upper quantiles. The ‘by_lag‘ table similarly contains draw-based summaries for signed contributions, percentage contributions, and, when ‘absolute = TRUE‘, absolute contributions.

With ‘output = "samples"‘, ‘by_lag_samples‘ and ‘ECI_weighted_samples‘ contain the corresponding draw-level quantities.

When ‘data‘ or multiple focal exposure profiles are evaluated, cumulative results are returned in ‘eci_summary‘ and lag-specific results in ‘by_lag‘, with the grouping column included when applicable.

The result also stores attributes describing the exact ECI-lag decomposition, common fitted maximum lag, profile length, reference values, and uncertainty setting.

See Also

‘compute_eci()‘, ‘summarise_effects()‘, ‘predict_outcomes()‘

`define_exposures` *Define DLNM exposure templates*

Description

Defines the training cross-basis parameterization for one or more exposure variables from long-format epidemic profiles.

Usage

```
define_exposures(
  data,
  vars,
  max_lag,
  df_var = 4,
  df_lag = 4,
  fun_var = "ns",
  fun_lag = "ns"
)
```

Arguments

data	Non-empty long-format data frame containing ‘epi_id‘, ‘time‘, and every exposure named in ‘vars‘. Rows do not need to be pre-sorted. They are ordered internally by ‘epi_id‘ and ‘time‘. Each epidemic must form one complete exposure profile containing exactly ‘max_lag + 1‘ time points. When more than one time point is present, spacing must be constant within epidemics and identical across epidemics. Missing exposure values are not allowed in EpiExposure v1. Because all exposures are finite columns of the same validated rows, every fitted exposure uses the same temporal support and profile length.
vars	Unique character vector naming exposure variables in ‘data‘.
max_lag	Non-negative integer maximum lag. The legacy form ‘c(0, L)‘ is also accepted, but EpiExposure v1 requires the lag range to start at 0.
df_var	Positive integer controlling the exposure-response basis when ‘fun_var‘ is ‘"ns"‘ or ‘"bs"‘. For ‘fun_var = "poly"‘, ‘df_var‘ is used as the polynomial degree. It is ignored for ‘fun_var = "lin"‘.
df_lag	Positive integer controlling the lag-response basis when ‘fun_lag‘ is ‘"ns"‘ or ‘"bs"‘. For ‘fun_lag = "poly"‘, ‘df_lag‘ is used as the polynomial degree. It is ignored for ‘fun_lag = "lin"‘.
fun_var	Character exposure-basis function. Supported unpenalized EpiExposure v1 choices are ‘"ns"‘, ‘"bs"‘, ‘"poly"‘, and ‘"lin"‘.
fun_lag	Character lag-basis function. Supported unpenalized EpiExposure v1 choices are ‘"ns"‘, ‘"bs"‘, ‘"poly"‘, and ‘"lin"‘. Penalized dlnm bases such as ‘"ps"‘ are not supported by the current engine-agnostic EpiExposure fitting contract. A P-spline transformation is not, by itself, a penalized DLNM: the corresponding penalty must also be supplied during model fitting (for example through ‘dlnm::cbPen()‘ and ‘mgcv::gam(..., paraPen = ...)‘, or through a dedicated cross-basis smooth). EpiExposure v1 therefore stops explicitly rather than fitting an unintentionally unpenalized ‘"ps"‘ basis.

Details

‘define_exposures()‘ is the first step in the standard EpiExposure workflow:

“‘ templates <- define_exposures(...) design <- build_design(data, templates) fit <- fit_epidlnm(...) “‘

The function estimates **basis definitions**, not disease effects. For each exposure it uses all training epidemics jointly to determine any data-dependent exposure-basis features (for example spline knots and boundary knots), while ‘dlnm::crossbasis(..., group = ...)‘ keeps the individual epidemic profiles as independent time series so lags never cross from one epidemic into another.

Independent epidemic series

Earlier EpiExposure code concatenated epidemic profiles and inserted ‘max_lag‘ missing values between them before calling ‘crossbasis()‘. The current implementation uses the native ‘group‘ argument of ‘dlnm::crossbasis()‘ instead.

This is the intended dlnm representation for several independent time series. After internal sorting, every epidemic remains consecutive and complete, while exposure values from all training epidemics still contribute to the common exposure-basis definition. Thus, for a spline exposure basis, data-dependent knots are estimated from the pooled **training exposure distribution**, but lagged profiles never cross epidemic boundaries.

Temporal requirements

A vector passed to ‘dlnm::crossbasis()‘ represents one complete exposure profile. ‘define_exposures()‘ therefore rejects:

- duplicated time values within an epidemic; - irregular spacing within an epidemic when more than one time point exists; - different time steps across epidemics; - epidemics with either fewer **or more** than ‘max_lag + 1‘ observations.

If the common maximum lag is ‘L‘, every epidemic must contain exactly

$$L + 1$$

observations. Profiles are never truncated to a trailing window, padded, or silently realigned. Because all fitted exposures are columns of the same long-format rows, all variables necessarily have the same temporal length and support within each epidemic.

For ‘max_lag = 0‘, the exact profile length is one observation. In that special case no time interval exists to estimate, so ‘epiexposure_time_step‘ is stored as ‘NA‘.

These are the same temporal assumptions enforced downstream by ‘build_design()‘.

Exposure and lag intercepts

The exposure basis is constructed without an intercept. This follows the standard DLNM identifiability requirement and avoids rank deficiency when a model intercept is present.

The lag basis includes an intercept. This is the standard cross-basis parameterization for the lag dimension and allows an effect distributed over the full fitted lag window.

Effective versus requested basis arguments

‘dlnm::crossbasis()‘ calls ‘onebasis()‘ internally and may normalize or modify basis arguments. Accordingly, EpiExposure stores the **effective** ‘argvar‘, ‘arglag‘, and lag attributes returned by the completed cross-basis, rather than assuming that the original request is the final parameterization.

These effective attributes are the authoritative training template reused by ‘build_design()‘, ‘predict_outcomes()‘, cross-validation, scenario functions, and effect summaries. Prediction data never redefine knots or boundary knots.

Scope of ‘df_var‘ and ‘df_lag‘

For spline bases, the user-supplied degrees of freedom determine the requested flexibility, but the returned ‘spec‘ records the effective basis arguments. For ‘"poly"‘, the corresponding ‘df_*‘ argument is interpreted as polynomial degree. For ‘"lin"‘, the corresponding ‘df_*‘ value has no effect.

For predictors with highly skewed distributions or many repeated values, ‘splines::ns()‘ may issue a knot-placement warning when interior knots coincide with boundary values. This adjustment is handled automatically and does not prevent model fitting.

Value

A named list of ‘dlnm::crossbasis‘ training templates, one per exposure. The list carries:

- ‘attr(x, "spec")‘: effective per-exposure basis specification; - ‘attr(x, "epiexposure_vars")‘: exposure names; - ‘attr(x, "epiexposure_max_lag")‘: the common fitted maximum lag; - ‘attr(x, "epiexposure_history_length")‘: the exact required exposure profile length, ‘max_lag + 1‘; - ‘attr(x, "epiexposure_history_contract")‘: ‘"all_fitted_exposures_same_exact_max_lag_plus_one"‘; - ‘attr(x, "epiexposure_time_step")‘: the common temporal spacing, or ‘NA‘ when ‘max_lag = 0‘ because one observation has no estimable spacing; - ‘attr(x, "epiexposure_template_contract")‘: ‘"training_grouped_crossbasis"‘; - ‘attr(x, "epiexposure_profile_order")‘: ‘"chronological"‘.

Each individual cross-basis also stores:

- ‘attr(cb, "cb_colnames")‘: native dlnm cross-basis column names; - ‘attr(cb, "epiexposure_canonical_cb_colnames")‘: the canonical future design names ‘cb_<variable>_<index>‘; - ‘attr(cb, "epiexposure_variable")‘: exposure name.

`define_periods` *Define epidemiological lag periods*

Description

Creates contiguous, non-overlapping lag periods for period-specific DLNM summaries and scenario definitions.

Usage

```
define_periods(max_lag, cuts = NULL, prefix = "W")
```

Arguments

`max_lag`	Non-negative integer maximum lag. The standard form is a scalar such as ‘85‘. For compatibility with lag-range metadata, ‘c(0, 85)‘ is also accepted. EpiExposure v1 requires lag histories to start at 0.
`cuts`	Optional numeric vector of **inclusive period-end lags**. For example, with ‘max_lag = 20‘: “‘ cuts = c(7, 14) “‘ creates: “‘ 0–7 8–14 15–20 “‘ Every cut must therefore be a unique finite integer strictly between ‘0‘ and ‘max_lag‘. Cut order in the input is not meaningful; valid cuts are arranged in increasing lag order before periods are constructed. ‘NULL‘ or ‘numeric(0)‘ requests one cumulative period from lag 0 through ‘max_lag‘.
`prefix`	Non-empty character scalar used to label periods. With the default ‘prefix = "W"‘, periods are labeled ‘"W1"‘, ‘"W2"‘, and so on.

Details

Periods are defined on the EpiExposure retrospective lag scale:

- lag 0 = the most recent exposure observation; - increasing lag = progressively older exposure observations; - ‘max_lag‘ = the oldest exposure observation included in the fitted profile.

Inclusive cut semantics

‘cuts‘ are not arbitrary break coordinates and are not the first lag of the following period. Each cut is the **inclusive final lag of the current period**.

Thus:

“‘ define_periods(max_lag = 20, cuts = c(7, 14)) “‘

returns:

“‘ period lag_start lag_end W1 0 7 W2 8 14 W3 15 20 “‘

Every lag from ‘0‘ through ‘max_lag‘ appears exactly once. There are no gaps and no overlaps.

Single cumulative period

When ‘cuts = NULL‘ or ‘cuts = numeric(0)‘, the full lag profile is represented by one period:

“‘ period lag_start lag_end W1 0 max_lag “‘

This also supports ‘max_lag = 0‘, in which case the single valid period is ‘0–0‘.

Validation philosophy

Earlier versions silently converted cut points with ‘as.integer()‘ and discarded cuts outside the fitted lag range. The current EpiExposure contract does not alter or ignore invalid period definitions silently.

Non-integer, duplicated, non-finite, boundary (‘0‘ or ‘max_lag‘), or out-of-range cut points produce explicit errors. This prevents downstream functions such as ‘summarise_effects()‘ and scenario simulation from using a period definition different from the one requested by the analyst.

Relationship with ‘summarise_effects()‘

For period-specific effects, ‘summarise_effects()‘ treats both ‘lag_start‘ and ‘lag_end‘ as inclusive. Therefore a period ‘8–14‘ represents exactly the lags:

“‘ 8, 9, 10, 11, 12, 13, 14 “‘

‘define_periods()‘ constructs exactly that contract and orders periods by increasing retrospective lag.

Value

A data frame with exactly three columns:

- ‘period‘: unique period label; - ‘lag_start‘: first lag included in the period; - ‘lag_end‘: last lag included in the period.

Period bounds are integer and inclusive. Rows are ordered from the most recent lag interval to the oldest.

The returned object also stores:

- ‘attr(x, "epiexposure_max_lag")‘: validated maximum lag; - ‘attr(x, "epiexposure_cuts")‘: validated internal cut points; - ‘attr(x, "epiexposure_n_lags")‘: total number of represented lags, ‘max_lag + 1‘; - ‘attr(x, "epiexposure_period_contract")‘: ‘"contiguous_inclusive_retrospective_lags"‘;

- ‘attr(x, "epiexposure_lag_scale")‘: ‘"lag_0_most_recent"‘; - ‘attr(x, "epiexposure_full_coverage")‘: ‘TRUE‘.

`ensemble_bestfit` *Create model and lag-contribution ensembles from best-fit DLNM models*

Description

Combines out-of-fold (OOF) predictions produced by the current ‘find_bestfit()‘ into one or more model-level ensembles and, optionally, combines lag-specific contributions from retained full-data model refits.

Usage

```
ensemble_bestfit(
  bestfit,
  predictions = NULL,
  lag_data = NULL,
  fit_list = NULL,
  data = NULL,
  group = "epi_id",
  var = NULL,
  compute_ecilag_args = list(),
  ensemble_scope = c("model", "lag", "both"),
  method = "unweighted",
  top_n = 3,
  model_id = NULL,
  weight_metric = NULL,
  threshold = NULL,
  weight_transform = c("softmax", "positive", "rank_inverse", "uniform"),
  stacking_model = c("ridge", "lm"),
  stack_objective = c("regularized", "min_error", "constrained"),
  lambda = 1e-06,
  stack_intercept = TRUE,
  model_col = "model_id",
  id_cols = c("group", "fold"),
  lag_group_cols = c("var", "lag"),
  lg_strategy = c("requested_available", "model", "strict"),
  rl_weights = TRUE,
  verbose = TRUE
)
```

Arguments

`bestfit`	Non-empty data frame returned by the current ‘find_bestfit()‘ using ‘validation_method = "LOOCV"‘ or ‘validation_method = "k-fold"‘. The function

requires the standardized metadata stored by ‘find_bestfit()‘, including ‘family‘, ‘outcome_type‘, ‘rank_metric‘, ‘threshold‘, the prediction contract, ‘validation_method‘, ‘validation_scheme‘, ‘validation_assignments‘, the common fitted ‘max_lag‘, the expected profile length (‘max_lag + 1‘), and the EpiExposure exact-profile contract. Results produced with ‘validation_method = "holdout"‘ are not supported.

`predictions` Optional data frame of OOF predictions. If ‘NULL‘, ‘attr(bestfit, "predictions")‘ is used. Under the current ‘find_bestfit()‘ contract, required columns include ‘model_col‘, ‘group‘, ‘fold‘, ‘observed‘, and ‘predicted‘, in addition to any columns listed in ‘id_cols‘.

All selected models must contain predictions for the same OOF identifiers. The function deliberately does not silently reduce the ensemble to an intersection of different validation supports, because model weights and ensemble metrics would otherwise be evaluated on a different sample from the individual-model ranking.

`lag_data` Optional data frame of deterministic lag-specific ECI decompositions. Required columns are ‘model_col‘, all ‘lag_group_cols‘, and ‘ECI_weighted‘. If ‘NULL‘ and a lag ensemble is requested, contributions are generated with ‘compute_ecilag()‘ from retained full-data fits.

`fit_list` Optional named list of fitted models used for automatic lag decomposition. If ‘NULL‘, ‘attr(bestfit, "fits")‘ is used.

`data` Optional long-format exposure data used only for automatic lag decomposition, while ignored/not required for ensemble_scope = "model". It must contain ‘group‘, a column named ‘time‘, and the exposures requested from the retained fitted models. Rows are ordered internally by ‘group‘ and ‘time‘, and time must satisfy the regular-series requirements used elsewhere in EpiExposure. Every group must contain exactly the common fitted ‘max_lag + 1‘ observations. Longer profiles are not truncated and shorter profiles are not padded.

`group` Character scalar naming the grouping column in ‘data‘ for automatic lag decomposition.

`var` Optional unique character vector of exposure variables requested for automatic lag decomposition.

`compute_ecilag_args`
Named list of additional arguments passed to ‘compute_ecilag()‘. It cannot override ‘data‘, ‘group‘, ‘fit‘, ‘var‘, or ‘profile‘. ‘uncertainty = TRUE‘ is not supported by ‘ensemble_bestfit()‘ v1 for lag ensembles because parameter draws from separately fitted models do not form a defined joint draw distribution.

`ensemble_scope` Character. One of ‘"model"‘, ‘"lag"‘, or ‘"both"‘.

`method` One or more methods selected from ‘"best"‘, ‘"hard_voting"‘, ‘"unweighted"‘, ‘"weighted"‘, and ‘"stacked"‘. Multiple methods may be evaluated in one call, for example ‘c("unweighted", "weighted", "stacked")‘.

‘"stacked"‘ and ‘"hard_voting"‘ are model-level methods only. ‘"hard_voting"‘ additionally requires a binomial outcome.

`top_n` Positive integer number of top-ranked models selected when ‘model_id = NULL‘.

`model_id` Optional vector of unique model identifiers. When supplied, ‘top_n‘ is ignored. Every requested model must have a finite value for ‘weight_metric‘.

`weight_metric` Optional performance metric used to rank/select models. When ‘model_id = NULL‘, the ‘top_n‘ models are selected according to this metric. For ‘method = "weighted"‘, it is also used to construct model weights. For ‘method = "stacked"‘, it determines which models are selected but is not transformed directly into stacking weights; those weights are estimated from the aligned OOF predictions by the level-2 model. If ‘NULL‘, the function inherits ‘attr(bestfit, "rank_metric")‘. Metric direction is determined by ‘.metric_direction()‘.

`threshold` Optional probability threshold strictly between 0 and 1 for binomial classification metrics and ‘hard_voting‘. If ‘NULL‘, ‘attr(bestfit, "threshold")‘ is used.

When ‘weight_metric‘ is threshold-dependent (‘Accuracy‘, ‘Balanced_Accuracy‘, ‘Sensitivity‘, ‘Specificity‘, ‘F1‘, ‘MCC‘, or ‘Precision‘), a user-supplied threshold must equal the threshold used by ‘find_bestfit()‘. Otherwise the stored model-ranking metric and the ensemble threshold would refer to different classification rules, so the function stops explicitly.

`weight_transform`

Character. Transformation used only by ‘method = "weighted"‘:

‘"softmax"‘ Stable softmax of the direction-adjusted metric.

‘"positive"‘ For maximize metrics, negative values are truncated to zero and positive values are normalized. For minimize metrics, reciprocal loss is normalized. If no positive utility exists, the function stops instead of silently switching to uniform weights.

‘"rank_inverse"‘ Weights proportional to inverse selected-model rank.

‘"uniform"‘ Equal weights.

`stacking_model` Character. ‘"ridge"‘ or ‘"lm"‘. It must be coherent with ‘stack_objective‘: ‘"regularized"‘ requires ‘"ridge"‘ and ‘"min_error"‘ requires ‘"lm"‘. With ‘"constrained"‘, ‘"ridge"‘ adds the penalty ‘lambda‘ whereas ‘"lm"‘ uses no ridge penalty.

`stack_objective`

Character. ‘"regularized"‘, ‘"min_error"‘, or ‘"constrained"‘.

‘"regularized"‘ fits ridge stacking. ‘"min_error"‘ fits ordinary least-squares stacking. ‘"constrained"‘ minimizes squared error subject to non-negative model weights summing to one, using projected-gradient optimization on the simplex.

`lambda` Non-negative ridge penalty used by regularized stacking and by constrained stacking when ‘stacking_model = "ridge"‘.

`stack_intercept`

Logical. Include an intercept for unconstrained stacking. ‘stack_objective = "constrained"‘ requires ‘stack_intercept = FALSE‘, because an unconstrained intercept would destroy the convex-combination interpretation.

`model_col` Character scalar naming the model identifier column.

`id_cols` Unique character vector identifying an OOF prediction row. Under the current ‘find_bestfit()‘ output the default ‘c("group", "fold")‘ is appropriate for both LOOCV and grouped k-fold.

`lag_group_cols` Unique character vector defining one lag-ensemble unit. It must include ‘"lag"‘. Include the epidemic/group column as well when lag contributions are group-specific, for example ‘c("epi_id", "var", "lag")‘.

lg_strategy	Character. Controls automatic lag-variable availability: ‘"requested_available"‘ uses requested variables available in each model, ‘"model"‘ uses every exposure fitted in each model, and ‘"strict"‘ requires every requested variable in every selected model. For lag aggregation, ‘"strict"‘ additionally requires every lag unit to be available from all selected models.
rl_weights	Logical. For lag ensembles, renormalize model weights among models actually available within each lag unit. If ‘FALSE‘, missing models retain their absent weight mass and the lag contribution can therefore shrink toward zero.
verbose	Logical. Print concise information.

Details

The function inherits the EpiExposure v1 prediction contract from ‘find_bestfit()‘: model-level ensemble predictions represent population/fixed-component **expected responses**, with fitted random effects set to zero. The OOF predictions used here are deterministic predictions from the harmonized central parameter estimate; coefficient/posterior uncertainty, residual noise, future-observation noise, and group-specific random effects are not added.

Model-level OOF ensemble contract

The current ‘find_bestfit()‘ produces one deterministic OOF expected-response prediction per successfully validated group and candidate model. This function combines those predictions; it does not call the original model prediction methods again.

Selected models must share the same OOF support. This strict alignment is intentional. Silently using only complete rows across models could make an ensemble appear better simply because difficult groups were removed.

For non-binary outcomes, ensemble performance uses CCC, Cb, Pearson correlation, RMSE, and MAE. For binomial outcomes, probability-based ensembles retain probabilities for ROC AUC, Brier score, and Log Loss and apply ‘threshold‘ only for classification metrics.

‘hard_voting‘ is different: each individual predicted probability is thresholded first, and the final class is the majority vote. Exact ties are resolved with the class produced by the highest-ranked selected model. ‘vote_fraction‘ is reported for interpretation but is not treated as a calibrated probability, so ROC AUC, Brier score, and Log Loss are returned as ‘NA‘ for hard voting.

Supported validation methods and holdout exclusion

‘ensemble_bestfit()‘ accepts results produced by ‘find_bestfit()‘ with ‘validation_method = "LOOCV"‘ or ‘validation_method = "k-fold"‘. Under these validation schemes, every successfully evaluated group receives one out-of-fold prediction, and the selected models are required to share the same complete OOF support.

Results produced with ‘validation_method = "holdout"‘ are intentionally not supported. Grouped holdout generates evaluation predictions for only one reserved subset of groups. If that same subset were used to select models, estimate ensemble weights, and report ensemble performance, the resulting performance would not represent an independent final assessment.

Numerically, simple combinations such as unweighted averaging could be calculated on holdout predictions. However, supporting them alongside weighted and stacked ensembles would create different validation supports and inferential contracts across methods. EpiExposure therefore applies one consistent rule to all model-level ensemble methods: ensemble construction and evaluation require aligned predictions from at least two validation folds.

A holdout-based ensemble workflow would require separate data roles, such as a training set for fitting the base models, a validation set for selecting models and estimating ensemble weights, and an independent test set for final evaluation. That three-way partition is outside the current ‘find_bestfit()‘ validation contract.

Stacking and validation folds

Stacking has two separate outputs that should not be confused.

Ensemble performance is calculated from **fold-cross-fitted level-2 predictions**. For OOF fold f, stacking coefficients are fitted using the stored OOF rows from folds other than f, and those coefficients are then applied to fold f. With LOOCV this reduces to the historical leave-one-row-out level-2 calculation. With grouped k-fold, every epidemic in the same validation fold is held out from level-2 fitting together.

In ‘method = "stacked"‘, ‘weight_metric‘ is used to rank and select the base models when ‘model_id = NULL‘; it is not transformed directly into stacking weights. Stacking weights are level-2 regression coefficients estimated from the aligned OOF predictions of the selected models. Therefore, a stacking weight represents the contribution of one model conditional on the other selected models and does not necessarily follow the individual-model ranking.

During cross-fitted performance evaluation, these coefficients are re-estimated separately using the OOF rows outside each validation fold. After this evaluation, a final stacking model is fitted to the complete OOF prediction matrix. The resulting coefficients are reported in ‘model_weights‘ and ‘stacking_final_coefficients‘ and are intended for combining the selected base models after they have been refitted to all available data.

For unconstrained ridge or least-squares stacking, these coefficients may be negative and are not required to sum to one. Only ‘stack_objective = "constrained"‘ enforces non-negative weights that sum to one, with no stacking intercept.

This is a level-2 cross-fitting procedure based on the OOF prediction matrix already produced by ‘find_bestfit()‘. It is not a fully nested re-fitting of every base learner inside an additional outer validation loop; therefore ensemble performance should be interpreted as OOF ensemble performance for model development rather than as an independent external-validation estimate.

Exact common lag/profile contract

‘ensemble_bestfit()‘ inherits the exact-profile contract established by the current ‘find_bestfit()‘: all candidate/fitted exposures use one common non-negative integer ‘max_lag‘, and every original exposure profile contains exactly ‘max_lag + 1‘ equally spaced observations.

For automatic lag decomposition, the retained fitted models must report the same ‘max_lag‘, ‘history_length‘, and ‘history_contract‘ as ‘bestfit‘, and every group supplied in ‘data‘ must contain exactly that profile length. Profiles are never truncated, padded, or silently realigned.

If precomputed ‘lag_data‘ is supplied, the original long-format exposure profiles are no longer available to this function. In that route, ‘ensemble_bestfit()‘ validates the lag values against the inherited common ‘max_lag‘ but cannot reconstruct the original profile row counts.

Lag ensembles

Lag ensembles combine lag-specific ECI contributions, not raw DLNM coefficients:

$$ECI_{ens,l} = \sum_m \alpha_m ECI_{m,l}.$$

The model weights α_m come from OOF model performance, while lag-specific ECI values are obtained from ‘compute_ecilag()‘ applied to the selected models refitted on the complete dataset.

Percent contribution is based on the absolute contribution magnitude within each non-lag grouping unit:

$$100|ECI_{ens,l}|/\sum_{l}|ECI_{ens,l}|.$$

Ensemble uncertainty for lag contributions is deliberately not produced in EpiExposure v1. Independently sampled coefficient/posterior draws from separately fitted models do not automatically form a valid joint draw distribution, so arbitrary draw-index pairing is not performed.

Value

A list containing:

‘ensemble_summary‘ Model-level performance, one row per requested method.

‘ensemble_predictions‘ OOF ensemble predictions with an explicit ‘method‘ column and the aligned individual-model predictions.

‘ensemble_by_lag‘ Ensemble lag-specific ECI decomposition.

‘lag_data‘ Lag data used for aggregation.

‘model_weights‘ Method-specific model weights. For stacking, these are the **final** level-2 weights fitted to all available OOF rows after cross-fitted ensemble performance has been evaluated.

‘selected_models‘ Selected rows from ‘bestfit‘.

‘stacking_coefficients_by_fold‘ For stacked methods, level-2 coefficients fitted without the current OOF fold and used to predict that fold.

‘stacking_final_coefficients‘ Final stacking coefficients fitted to the complete OOF matrix for downstream use with full-data base-model refits.

‘metric_warnings‘ Warnings captured while evaluating ensemble performance metrics.

The returned list also carries the inherited family, threshold, population/expected-response prediction contract, validation metadata (‘validation_method‘, ‘validation_scheme‘, ‘k‘, ‘validation_seed‘, ‘validation_stratified‘, ‘validation_assignments‘, and ‘validation_balance‘), and temporal metadata (‘max_lag‘, ‘history_length‘, ‘history_contract‘, and ‘time_step‘).

---

epi_data *Simulated beta epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and a continuous plant disease outcome expressed as a proportion. The dataset is intended for demonstrating beta regression and the main modeling, interpretation, simulation, and prediction workflows available in EpiExposure.

Usage

```
epi_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

rain Daily rainfall in millimeters.

wetness Daily leaf wetness duration in hours.

y Continuous disease response expressed as a proportion strictly between 0 and 1 and repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("epi_data", package = "EpiExposure")
head(epi_data)
```

---

epi_sensitivity *Compute sensitivity of exposure-response curves*

Description

Estimates the local derivative of an already calculated exposure-response curve. The function can work directly with supplied curve values or first smooth the curve with a GAM and differentiate the fitted smooth.

Usage

```
epi_sensitivity(
  data,
  x = "value",
  y = "prediction",
  scenario_var = "scenario",
  k = 10,
  method = c("analytical", "finite"),
  smooth_basis = c("cs", "cr", "tp", "ts"),
  elasticity = TRUE,
  critical = TRUE,
  eps = 1e-06
)
```

Arguments

data	Non-empty data frame containing the curve to differentiate. At minimum, it must contain the columns selected by ‘x‘ and ‘y‘. Within each group defined by ‘scenario_var‘, ‘x‘ must contain unique finite numeric values. Duplicate ‘x‘ values are rejected rather than aggregated silently because a derivative requires one unambiguous curve value at each evaluation position.
x	Character scalar naming the predictor/evaluation coordinate. Default is ‘"value"‘.
y	Character scalar naming the curve value to differentiate. Default is ‘"prediction"‘.
scenario_var	‘NULL‘ or a character vector naming one or more grouping columns. The derivative is computed independently within every unique combination of these columns. The default profile identifier is ‘"scenario"‘. Set ‘scenario_var = NULL‘ to differentiate the entire supplied data set as one curve. If several exposure variables, scenarios, or parameter draws are present, include all columns needed to identify one unique curve, for example ‘scenario_var = c("var", "scenario")‘ or ‘scenario_var = c("scenario", "sample")‘.
k	Positive integer GAM basis dimension requested when ‘method = "analytical"‘. Default is ‘10‘. The effective value is reduced, when necessary, to the number of distinct ‘x‘ values in the current curve. It is never increased silently. A non-constant curve analyzed with ‘method = "analytical"‘ requires at least three distinct ‘x‘ values and ‘k >= 3‘.
method	Character. One of: - ‘"analytical"‘: EpiExposure name for a profile-based GAM smooth derivative. A univariate GAM is fitted with ‘mgcv::gam(..., method = "REML")‘, using the auxiliary smooth basis selected by ‘smooth_basis‘, then its linear-predictor matrix is differentiated numerically using a small within-range perturbation; - ‘"finite"‘: differentiates the supplied curve directly using local finite-difference

formulas. With at least three points, second-order three-point formulas are used and support irregular ‘x‘ spacing.

The label ‘"analytical"‘ is retained for backward compatibility, but this method is not a symbolic analytical derivative. It is a derivative of the fitted GAM smooth obtained by finite differencing its ‘lpmatrix‘.

`smooth_basis` Character. Auxiliary ‘mgcv‘ smooth basis used only when ‘method = "analytical"‘. One of:

- ‘"cs"‘ (default): shrinkage cubic regression spline; - ‘"cr"‘: cubic regression spline without the additional shrinkage modification; - ‘"tp"‘: thin plate regression spline; - ‘"ts"‘: shrinkage thin plate regression spline.

This choice belongs only to the **post-processing GAM fitted inside ‘epi_sensitivity()‘**. It does **not** redefine, replace, inherit, or need to match the exposure-response or lag-response basis used by ‘define_exposures()‘, ‘fit_epidlnm()‘, or ‘find_bestfit()‘. The DLNM has already been fitted before this function is called.

‘"cs"‘ remains the default because it provides a low-rank univariate cubic smooth with shrinkage, which is a useful conservative default when differentiating an already estimated curve. The alternatives are exposed to support transparent sensitivity/robustness analyses of this secondary smoothing step.

‘smooth_basis‘ is validated even when ‘method = "finite"‘ for API consistency, but it has no computational effect in that method and the output metadata records it as unused.

`elasticity` Logical scalar. If ‘TRUE‘, compute local elasticity

$$E(x) = \frac{dy}{dx}\frac{x}{y}.$$

For ‘method = "analytical"‘, the denominator is the GAM-fitted curve value, so the derivative and elasticity refer to the same fitted function. For ‘method = "finite"‘, the denominator is the supplied ‘y‘.

Elasticity is returned as ‘NA‘ when its denominator is numerically indistinguishable from zero.

`critical` Logical scalar. If ‘TRUE‘, detect derivative sign changes. A sign change from positive to negative is classified as a local maximum; a change from negative to positive is classified as a local minimum.

Because the true zero crossing can lie between evaluated ‘x‘ values, the nearest supplied/evaluated row is flagged and the interpolated crossing is returned in ‘critical_x‘.

`eps` Positive finite numeric scalar smaller than 1. Default is ‘1e-6‘.

It is used as a relative numerical tolerance. For the GAM derivative, the perturbation is proportional to the observed ‘x‘ span and is constrained to remain inside that span. For elasticity and critical-point detection, scale-aware tolerances are derived from the corresponding curve/derivative magnitudes.

Details

‘epi_sensitivity()‘ is a **post-processing** function. It does not fit an EpiExposure DLNM, reconstruct exposure profiles, or redefine the fitted lag window. Consequently, rows supplied to this

function are curve-evaluation points, not necessarily the ‘max_lag + 1‘ observations that constituted an original exposure profile.

GAM-based derivative

For ‘method = "analytical"‘, EpiExposure fits the descriptive curve

$$y = s(x) + \epsilon$$

with REML smoothing-parameter estimation and the auxiliary ‘mgcv‘ basis selected through ‘smooth_basis‘.

The supported bases are deliberately restricted to four standard one-dimensional choices:

- ‘"cs"‘: shrinkage cubic regression spline; - ‘"cr"‘: cubic regression spline; - ‘"tp"‘: thin plate regression spline; - ‘"ts"‘: shrinkage thin plate regression spline.

This auxiliary spline is **not the DLNM spline**. It is fitted only after an exposure-response curve has already been produced. For example, a DLNM may have been fitted with a natural spline exposure basis and the resulting curve may still be differentiated here with ‘smooth_basis = "cs"‘ or ‘"tp"‘. There is no requirement that the two bases match because they serve different statistical roles.

The DLNM basis determines the epidemiological model itself. In contrast, ‘smooth_basis‘ controls only how the already evaluated curve is optionally smoothed before numerical differentiation. Users interested in robustness can therefore compare ‘"cs"‘, ‘"cr"‘, ‘"tp"‘, and ‘"ts"‘ without changing the previously fitted DLNM.

Let $X_p(x)$ denote the GAM linear-predictor matrix and $\hat{\beta}$ its fitted coefficient vector. The fitted curve is

$$\hat{f}(x) = X_p(x)\hat{\beta}.$$

The derivative mapping is approximated from two nearby prediction matrices:

$$X_p'(x) \approx \frac{X_p(x_+) - X_p(x_-)}{x_+ - x_-},$$

and

$$\hat{f}'(x) = X_p'(x)\hat{\beta}.$$

Interior points use a centered perturbation. Boundary points use a one-sided perturbation that remains inside the observed ‘x‘ range; the function does not intentionally extrapolate beyond the supplied curve.

The external method name ‘"analytical"‘ is retained because it was part of the previous EpiExposure API. Methodologically, however, this is a GAM-smoothed numerical derivative, not symbolic differentiation.

Why ‘"cs"‘ remains the default

Derivatives can amplify small-scale wiggles in an estimated curve. A shrinkage cubic regression spline is therefore retained as the default auxiliary smoother because it offers a simple one-dimensional cubic basis while allowing stronger penalization of weak structure. This is a post-processing default, not a statement that ‘"cs"‘ is universally optimal.

‘smooth_basis = "cr"‘ is useful when the analyst wants the corresponding cubic regression spline without the extra shrinkage modification. ‘"tp"‘ provides the general thin plate regression spline commonly used by ‘mgcv‘, while ‘"ts"‘ adds shrinkage to that family.

Because the estimated derivative can depend on the secondary smoothing choice, reporting or checking sensitivity across these bases can be useful when derivative-based scientific conclusions depend on fine features of the curve.

Direct finite differences

With ‘method = "finite"‘, no smoother is fitted. For three or more points, the derivative at an interior point is obtained from the quadratic interpolant through the previous, current, and next observations. This gives the standard centered three-point derivative for equally spaced data and its corresponding unequal-spacing form for irregular ‘x‘.

Endpoints use the corresponding one-sided three-point formula. With exactly two points, the secant slope is the only estimable derivative and is assigned to both positions.

‘x‘ values must be strictly unique after grouping. The function does not average duplicate positions, discard rows, or replace a zero spacing by ‘eps‘.

Elasticity

Elasticity is dimensionless:

$$E(x) = f'(x)x/f(x).$$

For the GAM method, both $f'(x)$ and $f(x)$ come from the same fitted smooth. For the direct finite method, $f(x)$ is the supplied ‘y‘.

Elasticity can be numerically unstable near zero. EpiExposure therefore returns ‘NA‘ when the relevant curve value is within a scale-aware tolerance of zero. No arbitrary constant is added to the denominator.

Critical points

‘critical = TRUE‘ searches for changes in the sign of the estimated derivative. Near-zero derivative values are treated as a bridge between the nearest non-zero derivative signs. Only genuine positive-to-negative or negative-to-positive transitions are marked.

Therefore:

- positive -> negative: local maximum; - negative -> positive: local minimum.

Flat regions without a sign reversal are not automatically labelled as extrema. Boundary extrema are also not inferred from a one-sided derivative alone.

Relationship to the EpiExposure exact-profile contract

‘epi_sensitivity()‘ does not consume raw fitted exposure profiles. It works on an already evaluated curve, whose number of rows may legitimately differ from the original profile length. Consequently, it does **not** require

$$nrow(data) = max_lag + 1.$$

If current EpiExposure temporal metadata are present on ‘data‘, however, the function validates and propagates them. Specifically,

$$profile_length = max_lag + 1$$

and the stored profile contract must be ‘"all_fitted_exposures_same_exact_max_lag_plus_one"‘.

This validates the provenance of an EpiExposure-derived curve without incorrectly treating curve-evaluation rows as original exposure-profile observations.

Uncertainty

‘epi_sensitivity()‘ does not generate coefficient/posterior draws. If the supplied data contain draw-specific curves, include the draw identifier in ‘scenario_var‘ so each draw is differentiated independently. The function does not silently pool repeated ‘x‘ positions across draws.

Value

A data frame containing all original input columns, ordered within group by increasing ‘x‘, plus:

- ‘sensitivity‘: estimated local derivative ‘dy/dx‘; - ‘sensitivity_curve_value‘: value of the curve whose derivative is being reported. This equals the GAM-fitted value for ‘method = "analytical"‘ and the supplied ‘y‘ for ‘method = "finite"‘; - ‘elasticity‘, when ‘elasticity = TRUE‘; - ‘critical‘, ‘critical_type‘, and ‘critical_x‘, when ‘critical = TRUE‘.

‘critical_type‘ is ‘"local_maximum"‘ or ‘"local_minimum"‘ for detected derivative sign changes and ‘NA‘ otherwise. ‘critical_x‘ is a linear interpolation of the derivative zero crossing between the bracketing evaluation points.

Output attributes document the derivative method, grouping variables, numerical contract, requested/effective GAM basis dimensions, and the auxiliary ‘smooth_basis‘ used by the GAM derivative.

`find_bestfit` *Find the best DLNM model structure using grouped validation*

Description

Evaluates combinations of exposure variables and cross-basis dimensions using grouped validation, then ranks candidate models with family-aware predictive performance metrics. Three grouped validation strategies are available:

Usage

```
find_bestfit(
  data,
  response = "y",
  group = "epi_id",
  time = "time",
  vars,
  max_lag,
  df_var_grid = c(3, 4, 5),
  df_lag_grid = c(3, 4, 5),
  min_vars = 1,
  max_vars = NULL,
  var_sets = NULL,
  fun_var = "ns",
```

```
  fun_lag = "ns",
  model_engine = "glmmTMB",
  family = "beta",
  random_effect = NULL,
  random_effect_prior = NULL,
  spatial_effect = NULL,
  spatial_structure = "matern",
  spatial_group = NULL,
  min_success = 2,
  rank_metric = NULL,
  threshold = 0.5,
  top_n = Inf,
  keep_fits = FALSE,
  verbose = TRUE,
  validation_method = c("LOOCV", "k-fold", "holdout"),
  k = 5,
  test_prop = 0.3,
  seed = NULL,
  ...
)
```

Arguments

data	Long-format data frame containing the response, grouping column, chronological time column, and candidate exposure variables.
response	Character scalar naming the response column in ‘data‘. The response may be repeated over exposure-profile rows but must be constant within each complete validation group because one outcome is predicted per group.
group	Character scalar naming the independent grouped-validation unit. Every row belonging to one group remains together. Internally, ‘data_long$epi_id‘ is a canonical alias of ‘data_long[[group]]‘; it does not redefine the user’s grouping structure.
time	Character scalar naming the chronological time column. Time must be finite and unique within groups. Within every group observations must be complete and equally spaced, and the time step must be the same across groups. Every group must contain exactly ‘max_lag + 1‘ time points. Because all candidate exposure variables are finite columns of the same validated long-format rows, all variables necessarily use the same temporal support. Rows are ordered internally by ‘group‘ and ‘time‘.
vars	Unique character vector of candidate exposure variables. Exposure variables must be distinct from ‘response‘, ‘group‘, and ‘time‘.
max_lag	Non-negative integer maximum lag, expressed in observation steps. A two-element form such as ‘c(0, 85)‘ is accepted and normalized to its maximum. Because all groups must use the same time step, a lag index has the same temporal meaning across groups.
df_var_grid	Positive integer candidate dimensions for the exposure-response basis.
df_lag_grid	Positive integer candidate dimensions for the lag-response basis.

`min_vars` Positive integer minimum number of exposure variables.

`max_vars` Positive integer maximum number of exposure variables, or ‘NULL‘ to use ‘length(vars)‘.

`var_sets` Optional list of exact variable combinations. When supplied, ‘min_vars‘ and ‘max_vars‘ do not determine the combinations, although their basic argument validation is still applied.

`fun_var` Character exposure-basis function passed to ‘define_exposures()‘.

`fun_lag` Character lag-basis function passed to ‘define_exposures()‘.

`model_engine` Character model engine supported by ‘fit_epidlnm()‘: ‘"glm"‘, ‘"glmmTMB"‘, ‘"gam"‘, ‘"gamm"‘, ‘"gls"‘, ‘"spamm"‘, ‘"brms"‘, ‘"inla"‘, or ‘"bdlnm"‘.

`family` Character or family object passed to ‘prepare_response()‘ and ‘fit_epidlnm()‘. Supported EpiExposure v1 canonical families are ‘"beta"‘, ‘"binomial"‘, ‘"poisson"‘, ‘"gamma"‘, ‘"gaussian"‘, and ‘"negative_binomial"‘ (NB2). Ordinal outcomes and NB1 are not supported in EpiExposure v1. Family-object links are checked against the selected engine before validation and passed unchanged to ‘fit_epidlnm()‘. To specify a non-default link, supply a supported family object, such as ‘family = stats::Gamma(link = "inverse")‘; character family names use the EpiExposure default link.

`random_effect` Optional character scalar naming one grouping column used as a conventional random intercept by ‘fit_epidlnm()‘. It must be constant within each validation group. Random slopes, nested or crossed random-effect specifications, and arbitrary engine-specific random-effect expressions are not supported.

For ‘brms‘, the random intercept is fitted as ‘(1 | group)‘. For the INLA-backed ‘inla‘ and ‘bdlnm‘ engines, it is fitted as ‘INLA::f(group, model = "iid")‘. EpiExposure v1 does not expose structured INLA latent effects such as ‘"rw1"‘, ‘"rw2"‘, ‘"ar1"‘, ‘"besag"‘, ‘"bym"‘, ‘"bym2"‘, or SPDE effects.

Random effects may contribute to model fitting, but validation predictions are always population-level and therefore set fitted random effects to zero. ‘glm‘ and ‘gls‘ do not accept ‘random_effect‘; ‘gamm‘ requires it under the current EpiExposure v1 fitting contract.

`random_effect_prior`

Optional named list defining the hyperprior for the precision of the IID random intercept when ‘model_engine = "inla"‘ or ‘"bdlnm"‘. The list is passed unchanged to the ‘hyper‘ argument of the internally generated ‘INLA::f(..., model = "iid")‘ term. For example, ‘list(prec = list(prior = "pc.prec", param = c(1, 0.01)))‘.

‘NULL‘ retains the engine default. A non-‘NULL‘ value requires ‘random_effect‘ and is currently supported only for INLA-backed engines. The argument customizes the hyperprior of the IID structure; it does not select another latent model. The same prior is used in every validation training fit and in retained full-data refits.

`spatial_effect` ‘NULL‘ (default), or two distinct numeric coordinate column names for a Matérn term fitted by ‘model_engine = "spamm"‘ only. Coordinates are constant within each epidemic but may be identical across different epidemics; they never redefine the exposure-profile unit.

`spatial_structure`
: Only ‘"matern"‘ is supported (case-insensitive) when spatial coordinates are supplied; otherwise this setting is inert.

`spatial_group`
: ‘NULL‘ for one shared field or a factor, character, or integer grouping column such as ‘"year"‘ for independent spatial fields. Requires spatial coordinates and must be constant within each epidemic. Spatial effects and ‘random_effect‘ are distinct and may coexist.

`min_success`
: Positive integer of at least 2 giving the minimum number of **evaluation groups with a finite out-of-sample validation prediction** required for a candidate to be eligible for metric calculation and, when requested, full-data refitting. Under LOOCV and k-fold, all groups are evaluation groups. Under holdout, only the fixed test groups count toward ‘n_success‘, ‘n_failed‘, and ‘min_success‘; holdout training groups are never counted as failed predictions. Candidate diagnostics retain ‘n_success_folds‘ and ‘n_failed_folds‘ as complete-validation-iteration counters: for holdout there is one iteration, successful only when every designated test group receives a finite prediction.

`rank_metric`
: Optional character scalar selecting the primary ranking metric. If ‘NULL‘, the default is ‘"CCC"‘ for non-binary outcomes and ‘"ROC_AUC"‘ for binary outcomes. Non-binary metrics are ‘"CCC"‘, ‘"Cb"‘, ‘"rho"‘, ‘"RMSE"‘, and ‘"MAE"‘. Binary metrics are ‘"ROC_AUC"‘, ‘"Brier"‘, ‘"LogLoss"‘, ‘"Accuracy"‘, ‘"Balanced_Accuracy"‘, ‘"Sensitivity"‘, ‘"Specificity"‘, ‘"F1"‘, ‘"MCC"‘, and ‘"Precision"‘. Metric direction is resolved automatically.

`threshold`
: Numeric probability strictly between 0 and 1 used only for binomial classification metrics. The default is ‘0.5‘. Predicted probabilities are retained; thresholding is used only to create the auxiliary 0/1 class required by classification metrics. ROC AUC, Brier score, and Log Loss do not depend on the threshold.

`top_n`
: Positive integer or ‘Inf‘; number of ranked candidates returned.

`keep_fits`
: Logical. Validation metrics are always calculated first and exclusively from out-of-sample predictions produced by the selected validation strategy. If ‘TRUE‘, every candidate that reaches ‘min_success‘ is **then** refitted on the complete data using a full-data basis. These full-data refits do not generate, replace, or alter validation predictions and do not participate in metric calculation or ranking. After ranking and ‘top_n‘ filtering, only retained candidate fits are stored in ‘attr(result, "fits")‘. They are intended for downstream visualization, epidemiological interpretation, exposure-lag-response surfaces, effect summaries, lag decomposition, scenario simulation, and later prediction.

`verbose`
: Logical. Print the validation strategy, candidate grid, execution mode, concise warning summaries, and completion information.

`validation_method`
: Character grouped-validation strategy. ‘"LOOCV"‘ leaves one complete group out per iteration. ‘"k-fold"‘ assigns complete groups to ‘k‘ approximately balanced folds. ‘"holdout"‘ creates one fixed grouped proportional training/test split and uses the test groups as the validation set for every candidate. All strategies operate on complete groups and never split temporal rows from one group across partitions.

`k`
: Positive integer number of folds used only when ‘validation_method = "k-fold"‘. It must satisfy ‘2 <= k < number of groups‘. Fold sizes differ by at most one

group. For binomial outcomes, allocation is additionally stratified by the group-level 0/1 response and each response class must contain at least ‘k‘ groups.

`test_prop` Numeric scalar strictly between 0 and 1 giving the requested approximate proportion of complete groups reserved for the test partition when ‘validation_method = "holdout"‘. The default ‘0.30‘ requests approximately 70 percent of groups for training and 30 percent for testing. It is validated for every call but ignored by LOOCV and k-fold. For binary outcomes, the effective holdout size may be adjusted to the closest feasible value that keeps both outcomes 0 and 1 in both training and test partitions.

`seed` Optional non-negative integer controlling grouped allocation for ‘validation_method = "k-fold"‘ and the grouped train/test split for ‘validation_method = "holdout"‘. The same data, group order, validation method, ‘test_prop‘, and seed reproduce the same allocation. The caller’s existing global ‘.Random.seed‘ is restored after allocation. ‘seed‘ does not alter LOOCV membership and is not passed to ‘fit_epidlnm()‘.

`...` Named additional arguments passed to ‘fit_epidlnm()‘. Core arguments managed by ‘find_bestfit()‘ (‘data‘, ‘model_engine‘, ‘family‘, ‘random_effect‘, ‘random_effect_prior‘, ‘spatial_effect‘, ‘spatial_structure‘, ‘spatial_group‘, ‘epiexposure_spec‘, and ‘basis_objects‘) cannot be supplied again through ‘...‘. For INLA-backed engines, likelihood availability, ‘control.family$control.link$model‘ consistency, and ‘control.compute$config = TRUE‘ compatibility are checked before validation. ‘fit_epidlnm()‘ supplies required INLA controls during each fit.

Details

- ‘"LOOCV"‘: grouped leave-one-out cross-validation; - ‘"k-fold"‘: grouped k-fold cross-validation; - ‘"holdout"‘: one grouped proportional holdout validation split.

The validation unit is always the complete column named by ‘group‘. The function creates the internal canonical alias ‘epi_id‘ from that column, but never splits a group’s exposure-profile rows across training and evaluation. With the default ‘group = "epi_id"‘, complete epidemics are therefore kept intact in every strategy.

The statistical target used for every validation prediction is the same target used by the EpiExposure v1 prediction layer: the population/fixed-component expected response, with fitted random effects set to zero. Model ranking uses deterministic predictions from the harmonized central parameter estimate and does not propagate coefficient, posterior, residual, or future-observation uncertainty.

Grouped validation and cross-validation

Validation membership is constructed once before candidate evaluation and is reused unchanged for every candidate. This guarantees fair comparison on the same out-of-sample groups. LOOCV produces one evaluation iteration per group. Grouped k-fold randomizes complete groups into approximately balanced folds; for binomial outcomes, allocation is stratified so both classes occur in every fold whenever the documented class-count requirement is satisfied.

Grouped proportional holdout

With ‘validation_method = "holdout"‘, the function creates exactly one grouped training/test split before candidate evaluation. ‘test_prop‘ determines the requested approximate fraction of complete groups assigned to testing. For non-binary outcomes, test groups are sampled directly. For binary

outcomes, sampling is stratified at the **group-level response**, not at longitudinal row level, and both outcomes 0 and 1 are required in both partitions. At least two complete groups in each class are therefore required.

Every candidate uses the same fixed holdout partition. ‘define_exposures()‘ is called only on the holdout training groups, so exposure-basis knots, boundary knots, effective dimensions, ‘argvar‘, ‘arglag‘, templates, and related cross-basis attributes are learned exclusively from training data. The stored training template is then transported unchanged to the holdout test profiles by ‘.build_design_from_templates()‘. Test values never redefine the basis.

Holdout metrics are calculated exclusively from predictions for the fixed test groups. Because that same test subset is used to compare and rank candidate structures, it is a **validation** holdout, not an untouched final external test set. A final independent external evaluation requires another data set or partition that was never used for candidate selection. If ‘keep_fits = TRUE‘, eligible candidates are refitted on all available data only after their validation metrics have already been calculated; these refits are downstream models and never replace holdout predictions.

Training-only basis construction

For every validation iteration, ‘define_exposures()‘ receives only the corresponding training groups. The effective ‘argvar‘, ‘arglag‘, lag range, spline knots, boundary knots, and other returned basis attributes are reused to transform the evaluation profiles. For highly skewed predictors or many repeated values, ‘splines::ns()‘ may warn when an interior knot coincides with a boundary knot; the effective training basis returned after that adjustment is the template transported to evaluation data.

The function validates basis transport explicitly. Requested ‘max_lag‘, the lag stored in training cross-bases, and the lag stored in the exposure specification must agree. Reconstructed training and evaluation cross-bases must have matching dimensions and native column names/order. Canonical ‘cb_<variable>_<index>‘ names are assigned deterministically. For ‘model_engine = "bdlnm"‘, an epidemic-level matrix-form cross-basis is built from the same training parameters, numerically checked against the canonical design, and exposure-prefixed so internal INLA names remain globally unique.

Temporal requirements

Every complete group must contain exactly ‘max_lag + 1‘ ordered, equally spaced observations. Profiles are never truncated, padded, or silently realigned. All candidate exposures are finite columns of the same validated long-format rows and therefore share identical temporal support.

Random-effect scope

Candidate models may include one conventional random intercept through ‘random_effect‘. The same random-intercept specification is used in every validation training fit and, when ‘keep_fits = TRUE‘, in the full-data refits retained after validation.

For Bayesian engines, EpiExposure v1 supports ‘(1 | group)‘ through ‘brms‘ and ‘f(group, model = "iid")‘ through ‘inla‘ and ‘bdlnm‘. The candidate grid does not search over alternative latent-effect structures, and ‘find_bestfit()‘ does not support INLA models such as ‘"rw1"‘, ‘"rw2"‘, ‘"ar1"‘, ‘"besag"‘, ‘"bym"‘, ‘"bym2"‘, or SPDE effects.

This restriction keeps the validation target and population-level prediction contract harmonized across candidates and engines. All fitted random-effect contributions are set to zero when out-of-sample validation predictions are generated.

Optional spatial covariance

Only ‘spaMM‘ accepts ‘spatial_effect = c("x_coord", "y_coord")‘ through the EpiExposure v1 interface. This adds a Matérn term during fitting; ‘spatial_group‘ may define independent fields. This

spaMM-specific interface is distinct from INLA/SPDE or areal latent-effect models, which are not currently exposed by ‘find_bestfit()‘.

Conventional random effects remain distinct. All fitted random and spatial effects are set to zero for population-level validation predictions.

Validation prediction target, warnings, and ranking

Every successful evaluation group receives one deterministic out-of-sample expected-response prediction from the harmonized central fixed/population parameter estimate. For LOOCV and k-fold these are out-of-fold predictions; for holdout they are out-of-sample holdout predictions. Coefficient/posterior uncertainty and residual/future-observation noise are excluded from ranking.

Warnings raised during fitting/prediction are captured and classified as ‘"knot"‘, ‘"convergence"‘, ‘"hessian"‘, or ‘"other"‘; warnings do not by themselves make a validation iteration fail. Candidate ranking uses ‘rank_metric‘ first, with the remaining family-appropriate metrics as deterministic tie-breakers in canonical order.

Candidate-level parallelism is preserved. When the caller configures a ‘future‘ plan with more than one worker, one future owns each candidate and that candidate’s validation iterations remain sequential. The function does not change the caller’s global ‘future::plan()‘.

Value

A data frame ranked by ‘rank_metric‘. Non-binary families report ‘CCC‘, ‘Cb‘, ‘rho‘, ‘RMSE‘, and ‘MAE‘; binomial models report ‘ROC_AUC‘, ‘Brier‘, ‘LogLoss‘, ‘Accuracy‘, ‘Balanced_Accuracy‘, ‘Sensitivity‘, ‘Specificity‘, ‘F1‘, ‘MCC‘, and ‘Precision‘.

Candidate diagnostics include ‘n_groups_total‘, ‘n_predictions‘, ‘n_success‘, ‘n_failed‘, ‘n_folds‘, ‘n_success_folds‘, and ‘n_failed_folds‘. ‘n_predictions‘ is the number of groups assigned to out-of-sample evaluation: all groups for LOOCV/k-fold, but only test groups for holdout. The ‘*_folds‘ fields are retained as validation-iteration counters; holdout therefore has one iteration.

Diagnostic attributes include ‘"predictions"‘, ‘"predictions_by_model"‘, ‘"failures"‘, ‘"warnings"‘, ‘"warning_summary"‘, ‘"timing"‘, and ‘"candidate_times"‘. With ‘keep_fits = TRUE‘, ‘"fits"‘ contains retained full-data refits and ‘"retained_fit_scope"‘ is ‘"full_data_refit_after_validation"‘; otherwise ‘"retained_fit_scope"‘ is ‘NA_character_‘.

Grouped-validation metadata include:

- ‘"validation_method"‘: ‘"LOOCV"‘, ‘"k-fold"‘, or ‘"holdout"‘; - ‘"validation_scheme"‘: normalized descriptive scheme; - ‘"k"‘: number of LOOCV iterations for LOOCV, supplied ‘k‘ for k-fold, and ‘NA_integer_‘ for holdout; - ‘"validation_seed"‘: supplied k-fold/holdout seed or ‘NA_integer_‘; - ‘"validation_stratified"‘: whether binomial stratification was used; - ‘"test_prop"‘: requested holdout proportion, otherwise ‘NA_real_‘; - ‘"n_groups_total"‘ and ‘"n_evaluation_groups"‘; - ‘"training_groups"‘, ‘"test_groups"‘, and ‘"evaluation_groups"‘; - ‘"validation_assignments"‘: data frame with ‘group‘, ‘partition‘, ‘fold‘, and ‘observed‘; holdout uses fixed ‘train‘/‘test‘ partitions, whereas LOOCV/k-fold use ‘partition = "evaluation"‘ and the evaluation fold; - ‘"validation_balance"‘: fold-level counts for LOOCV/k-fold or train/test counts for holdout, including 0/1 class counts for binomial outcomes.

Standard downstream metadata are retained in ‘"family"‘, ‘"outcome_type"‘, ‘"rank_metric"‘, ‘"threshold"‘, ‘"prediction_level"‘, ‘"prediction_estimand"‘, ‘"prediction_contract"‘, ‘"random_effect"‘, ‘"random_effect_model"‘, ‘"random_effect_prior"‘, ‘"spatial_effect"‘, ‘"spatial_structure"‘, ‘"spatial_group"‘, ‘"spatial_term"‘, ‘"has_spatial_effect"‘, ‘"basis_training_only"‘, ‘"max_lag"‘, ‘"history_length"‘, ‘"history_contract"‘, and ‘"time_step"‘.

Examples

```
best_models <- find_bestfit(
  data = epi_data,
  response = "y",
  group = "epi_id",
  time = "time",
  vars = "wetness",
  max_lag = 85,
  df_var_grid = 2,
  df_lag_grid = 2,
  var_sets = list("wetness"),
  model_engine = "glmmTMB",
  family = "beta",
  validation_method = "holdout",
  test_prop = 0.20,
  seed = 123,
  min_success = 2,
  top_n = 1,
  keep_fits = FALSE,
  verbose = FALSE
)

head(best_models)
```

---

fit_epidlnm *Fit a harmonized DLNM inferential model*

Description

Fits a distributed lag nonlinear model (DLNM) from an epidemic-level design matrix and stores the standardized EpiExposure metadata required by downstream prediction, effect summarization, scenario simulation, and model comparison functions.

Usage

```
fit_epidlnm(
  data,
  model_engine,
  family,
  random_effect = NULL,
  random_effect_prior = NULL,
  epiexposure_spec = NULL,
  basis_objects = NULL,
  spatial_effect = NULL,
  spatial_structure = "matern",
  spatial_group = NULL,
  ...
)
```

Arguments

`data` A data.frame containing ‘y_model‘ and the fitted DLNM design columns. For all engines except ‘bdlnm‘, cross-basis columns must follow the package convention ‘cb_<variable>_<index>‘. For ‘bdlnm‘, epidemic-level, matrix-form cross-basis objects are supplied through ‘basis_objects‘ instead. Data prepared by ‘prepare_response()‘ may carry the ‘response_family_name‘ attribute; when present, it must agree with ‘family‘.

`model_engine` Character scalar identifying the modeling engine. One of ‘"glm"‘, ‘"glmmTMB"‘, ‘"gam"‘, ‘"gamm"‘, ‘"gls"‘, ‘"spamm"‘, ‘"brms"‘, ‘"inla"‘, or ‘"bdlnm"‘.

`family` Response distribution supplied as a supported canonical name or as a family object from which a supported canonical family and link can be identified. The supported EpiExposure v1 families are ‘"beta"‘, ‘"binomial"‘, ‘"poisson"‘, ‘"gamma"‘, ‘"gaussian"‘, and ‘"negative_binomial"‘. The canonical negative-binomial model is the quadratic-variance NB2 parameterization. NB1 and ordinal models are not supported in this version and produce explicit errors.

`random_effect` Optional character scalar naming one grouping column. In EpiExposure v1, this argument represents a conventional random intercept only. Random slopes, nested or crossed random-effect specifications, and arbitrary engine-specific random-effect expressions are not supported.

For the Bayesian ‘brms‘ engine, the random intercept is fitted as ‘(1 | group)‘. For the INLA-backed ‘inla‘ and ‘bdlnm‘ engines, it is fitted as ‘INLA::f(group, model = "iid")‘. EpiExposure v1 does not expose other INLA latent structures through this argument, including ‘"rw1"‘, ‘"rw2"‘, ‘"ar1"‘, ‘"besag"‘, ‘"bym"‘, ‘"bym2"‘, or SPDE-based effects.

INLA-backed fits internally re-index arbitrary grouping labels to consecutive integers while retaining the original column name in EpiExposure metadata. ‘glm‘ and ‘gls‘ do not support this argument. ‘gamm‘ requires it in the current EpiExposure interface because its generated fixed formula contains no other smooth or random term.

`random_effect_prior`

Optional named list defining the hyperprior for the precision of the random intercept when ‘model_engine = "inla"‘ or ‘"bdlnm"‘. The list is passed unchanged to the ‘hyper‘ argument of the internally generated ‘INLA::f(..., model = "iid")‘ term.

For example, ‘list(prec = list(prior = "pc.prec", param = c(1, 0.01)))‘. ‘NULL‘ retains the engine’s default prior. A non-‘NULL‘ value requires ‘random_effect‘ and is currently supported only for the INLA-backed engines.

This argument customizes the hyperprior of the supported IID random intercept; it does not select or configure other INLA latent models. Structures such as ‘"rw1"‘, ‘"rw2"‘, ‘"ar1"‘, ‘"besag"‘, ‘"bym"‘, ‘"bym2"‘, and SPDE effects are not supported by the EpiExposure v1 random-effect interface.

`epiexposure_spec`

Named list describing the cross-basis construction for every fitted exposure. This metadata is required so downstream functions reconstruct exactly the fitted exposure-lag basis rather than re-inferring ‘df‘, knots, or basis functions. Each element must contain ‘max_lag‘, ‘argvar‘, and ‘arglag‘.

Under the EpiExposure exact-profile contract, each ‘max_lag‘ must be one non-negative integer and all fitted exposures must use the same ‘max_lag‘. Consequently, every original exposure profile used to build the epidemic-level design must contain exactly ‘max_lag + 1‘ observations for every fitted exposure. ‘fit_epidlnm()‘ receives the already collapsed epidemic-level design and therefore validates the common fitted ‘max_lag‘; exact original profile length is validated upstream by the exposure/design construction functions.

`basis_objects` Optional named list of ‘dlnm::crossbasis()‘ objects. For ‘model_engine = "bdlnm"‘, these must be the epidemic-level, matrix-form cross-basis objects containing exactly one row per model row. They are normally recovered from the ‘epiexposure_bdlnm_basis_objects‘ attribute produced by ‘build_design()‘, but may also be supplied explicitly. For other engines, when supplied, their names must match the fitted exposure variables.

`spatial_effect` NULL (default) or two distinct names of numeric, finite coordinate columns in the epidemic-level design. Only ‘model_engine = "spamm"‘ supports spatial autocorrelation. Coordinates may repeat across different epidemics; at least two distinct pairs are needed.

`spatial_structure`

Spatial correlation structure; currently only ‘"matern"‘ (case-insensitive) when ‘spatial_effect‘ is supplied. This argument does not change non-spatial fits.

`spatial_group` NULL for one shared spatial field, or the name of one grouping column (e.g., ‘"year"‘) for independent Matérn realizations with shared correlation parameters. Requires ‘spatial_effect‘; the source column may be factor, character, or integer and is converted in a local fit copy. ‘random_effect‘ remains a separate, conventional random intercept and both effects can be fitted together. Coordinates and spatial groups are not used as epidemic-profile identifiers.

`...` Named additional arguments passed to the selected engine. Core arguments managed by EpiExposure (‘formula‘/‘model‘, ‘data‘, ‘family‘, and the engine-specific random-effect argument) cannot be supplied again in ‘...‘. For INLA-backed fits (‘inla‘ and ‘bdlnm‘), ‘control.compute$config = TRUE‘ is required by the EpiExposure uncertainty contract and is added automatically when absent. If a conflicting value is supplied, the function stops.

Details

Engine harmonization

‘fit_epidlnm()‘ standardizes the *statistical target* across engines without requiring every engine to use the same computational implementation. The model is fitted using the selected engine, while downstream EpiExposure v1 predictions are defined as population-level expected responses. Random effects may therefore affect estimation during model fitting, but downstream predictions exclude group-specific random-effect contributions.

The random-intercept translation used at fitting is:

* ‘glmmTMB‘, ‘brms‘, and ‘spaMM‘: ‘(1 | group)‘; * ‘gam‘: ‘s(group, bs = "re")‘; * ‘gamm‘: ‘random = list(group = ~1)‘; * ‘INLA‘ and ‘bdlnm‘: ‘f(group, model = "iid")‘, optionally extended to ‘f(group, model = "iid", hyper = random_effect_prior)‘ when a custom random-effect hyperprior is supplied.

Random-effect scope in EpiExposure v1

The EpiExposure v1 random-effect interface is intentionally restricted to one conventional random intercept. For ‘brms‘, this corresponds to ‘(1 | group)‘; for ‘inla‘ and ‘bdlnm‘, it corresponds to ‘f(group, model = "iid")‘.

The interface does not currently construct structured Bayesian latent effects such as first- or second-order random walks (‘"rw1"‘ or ‘"rw2"‘), autoregressive effects (‘"ar1"‘), areal spatial effects (‘"besag"‘, ‘"bym"‘, or ‘"bym2"‘), or continuous spatial effects based on SPDE models. These structures may be available in the underlying INLA framework but are outside the harmonized fitting, prediction, validation, and uncertainty contract of EpiExposure v1.

The ‘spatial_effect‘ interface described below is separate from these Bayesian latent structures and currently supports Matérn spatial covariance through ‘spaMM‘ only.

‘glm‘ and ‘gls‘ are fixed-effect engines in this interface and reject a non-‘NULL‘ ‘random_effect‘. The ‘mgcv::gamm()‘ implementation uses the engine’s native ‘random‘ argument rather than lme4-style syntax. Because ‘gamm()‘ uses PQL for non-Gaussian responses and is specifically known to be problematic for binary data, a warning is issued for the binomial family.

Spatial covariance (spaMM only)

Spatial terms are explicitly distinct from ‘random_effect‘: the former model correlated Matérn fields, whereas the latter is an ordinary intercept. ‘spatial_effect = c("x_coord", "y_coord")‘ adds ‘Matern(1 | x_coord + y_coord)‘; adding ‘spatial_group = "year"‘ instead uses ‘Matern(1 | x_coord + y_coord leaves every pre-existing engine and random-intercept fit unchanged. Coordinates and spatial grouping affect estimation, but not EpiExposure’s default fixed-component prediction, which sets all non-fixed contributions to zero rather than integrating them over their distributions.

Families and links

Canonical families are translated to engine-native representations. The canonical ‘negative_binomial‘ family is standardized to an NB2-type model (‘Var(Y) = mu + mu^2 / shape‘) where supported. An explicit NB1 request is rejected instead of being silently converted.

Automatic family availability is engine-specific:

* ‘glm‘: Gaussian, Binomial, Poisson, Gamma; * ‘glmmTMB‘: all six EpiExposure v1 families; * ‘gam‘: all six EpiExposure v1 families; * ‘gamm‘: Gaussian, Binomial, Poisson, Gamma; * ‘gls‘: Gaussian with identity link only; * ‘spaMM‘: all six EpiExposure v1 families; * ‘brms‘: all six EpiExposure v1 families; * ‘INLA‘: all six EpiExposure v1 families, subject to the installed INLA likelihood implementation; * ‘bdlnm‘: all six EpiExposure v1 families through its INLA backend.

When a family object supplies a non-default link, EpiExposure records that exact link in ‘epiexposure_link‘ and reconstructs the corresponding engine-native family. For INLA-based engines the same link is forwarded via ‘control.family$control.link$model‘. Unsupported family-link combinations fail explicitly in EpiExposure or in the selected engine; no Gaussian or identity-link fallback is used.

To specify a non-default link, supply a supported family object, such as family = stats::Gamma(link = "inverse"); character family names use the EpiExposure default link.

For ‘gam‘, Beta and negative-binomial models use ‘mgcv‘ extended families. When needed, ‘method = "REML"‘ is supplied by default to satisfy the supported fitting route; an explicitly supplied incompatible method is rejected. A GAM random intercept also defaults to REML unless the user explicitly selects another method.

For ‘model_engine = "gamm"‘, the returned object contains the native ‘gam‘ and ‘lme‘ components and inherits from ‘"epiexposure_gamm"‘. Calling ‘summary()‘ returns the fixed/population GAM summary by default; use ‘component = "lme"‘ or ‘"both"‘ to inspect the mixed-model component.

Response-scale validation

To avoid engine-specific silent coercion, ‘y_model‘ is checked against the canonical family before fitting: Beta responses must lie strictly inside ‘(0, 1)‘; Binomial responses must be coded ‘0/1‘; Poisson and negative- binomial responses must be non-negative integer counts; Gamma responses must be strictly positive; Gaussian responses need only be finite numeric values.

Common lag and exact-profile contract

EpiExposure models use one common retrospective lag window for every fitted exposure. If the common maximum lag is ‘L‘, all original exposure profiles must contain exactly

$$L + 1$$

equally spaced observations per epidemiological unit and per exposure. Exposures with different fitted ‘max_lag‘ values are rejected here.

Because ‘fit_epidlnm()‘ is called after ‘build_design()‘ has collapsed each complete profile to one epidemic-level model row, the original long-format row count is no longer available at this stage. Exact ‘L + 1‘ temporal coverage must therefore be enforced by ‘define_exposures()‘ and ‘build_design()‘. This function records the common lag/profile contract in the fitted model metadata so downstream functions can enforce it.

Metadata and downstream prediction contract

The fitted exposure specification is part of the model contract, not an optional hint. Downstream functions must use the stored family, link, engine, cross-basis mapping, exposure specification, and data template. They should not infer a link from the family, rebuild a cross-basis from user-supplied degrees of freedom, align coefficients by position, or substitute a Gaussian/identity model when metadata are missing.

EpiExposure v1 prediction functions are intended to return the expected outcome under the fitted model, not a newly simulated observation. They use population-level predictions: random effects are included in model fitting when requested here, but group-specific random effects are not included in downstream predictions. This function itself only fits the model and does not create outcome predictions or uncertainty summaries.

For ‘glm‘ and ‘glmmTMB‘, models are fitted with the fully evaluated EpiExposure design, but their stored calls are compacted after fitting. The original ‘data‘ expression is retained for display instead of the complete evaluated data frame and its EpiExposure attributes. For ‘glm‘, the stored function and family expressions are also compacted to prevent ‘summary()‘ from printing the evaluated function definition and complete family object. These changes affect only the stored calls used for display and re-evaluation; they do not alter fitted coefficients, likelihoods, covariance matrices, predictions, or EpiExposure metadata.

Bayesian engines may generate posterior coefficient draws as part of their native fitting procedure (notably ‘bdlnm‘). Those stored coefficient draws are model output and are available for later draw-by-draw uncertainty propagation; they are not used here as a single deterministic prediction.

Ordinal outcomes are deliberately excluded from EpiExposure v1 so fitting, prediction, uncertainty, performance metrics, and ensemble behavior remain harmonized across the supported response families.

Value

A fitted model object with a strict metadata contract. The following attributes are attached for downstream functions: ‘epiexposure_family_name‘, ‘epiexposure_link‘, ‘epiexposure_engine‘, ‘epiexposure_cb_cols‘, ‘epiexposure_vars‘, ‘epiexposure_data_template‘, ‘epiexposure_id_col‘, ‘epiexposure_spec‘, and ‘epiexposure_basis_objects‘. Additional attributes record the standardized family parameterization, random-intercept structure, the INLA latent random-effect model in ‘epiexposure_random_effect_model‘, the optional ‘epiexposure_random_effect_prior‘, and the spatial specification (‘epiexposure_spatial_effect‘, ‘epiexposure_spatial_structure‘, ‘epiexposure_spatial_group‘, and ‘epiexposure_spatial_term‘), common fitted ‘max_lag‘, expected profile length (‘max_lag + 1‘), the exact-profile contract, the validated temporal step in ‘epiexposure_time_step‘, and the EpiExposure v1 prediction contract.

Examples

```
# Minimal epidemic-level design with one exposure basis column
dat <- data.frame(
  y_model = c(0.12, 0.18, 0.25, 0.31, 0.38, 0.44, 0.51, 0.57),
  cb_rain_1 = c(-1.5, -1.0, -0.5, 0, 0.5, 1.0, 1.5, 2.0)
)

# Cross-basis specification corresponding to the fitted exposure
spec <- list(
  rain = list(
    max_lag = 0L,
    argvar = list(fun = "lin"),
    arglag = list(fun = "integer")
  )
)

fit <- fit_epidlnm(
  data = dat,
  model_engine = "glm",
  family = "gaussian",
  epiexposure_spec = spec
)

stats::coef(fit)

attr(fit, "epiexposure_engine")
attr(fit, "epiexposure_family_name")
attr(fit, "epiexposure_vars")
```

---

gamma_data *Simulated gamma epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and a positive continuous plant disease outcome. The dataset is intended for demonstrating gamma models in EpiExposure.

Usage

```
gamma_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Positive continuous disease outcome repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("gamma_data", package = "EpiExposure")
head(gamma_data)
```

---

gaussian_data *Simulated gaussian epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and a continuous plant disease outcome. The dataset is intended for demonstrating gaussian models in EpiExposure.

Usage

```
gaussian_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Continuous disease outcome repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("gaussian_data", package = "EpiExposure")
head(gaussian_data)
```

---

nb2_data *Simulated negative-binomial epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and an overdispersed count disease outcome. The dataset is intended for demonstrating quadratic-variance negative-binomial models in EpiExposure.

Usage

```
nb2_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Non-negative count disease outcome repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("nb2_data", package = "EpiExposure")
head(nb2_data)
```

`plot_eci` *Plot ECI and lag-specific contribution results*

Description

Creates publication-ready plots from objects returned by ‘compute_eci()‘ and ‘compute_ecilag()‘. Either object may be supplied alone. When both are supplied, the overall ECI perspective and the lag-specific perspective are combined and labelled "(a)" and "(b)", respectively.

Usage

```
plot_eci(
  eci_overall = NULL,
  eci_lag = NULL,
  overall_col = c("ECI_weighted", "ECI_raw", "ECI_raw_centered"),
  lag_col = c("ECI_weighted", "ECI_percent", "ECI_absolute"),
  facet_scales = c("free_y", "fixed", "free", "free_x"),
  overall_nrow = 3L,
  lag_ncol = 1L,
  sample_smooth = TRUE,
```

```
  smooth_method = NULL,
  smooth_formula = NULL,
  linewidth = 0.7,
  alpha = 0.2,
  overall_x_label = "ECI",
  overall_y_label = "Predicted",
  lag_x_label = "Lag",
  lag_y_label = "ECI",
  exposure_y_label = "Exposure difference",
  exposure_colors = c("#4C72B0", "#C44E52", "#55A868"),
  base_size = 10,
  panel_labels = c("(a)", "(b)"),
  label_size = 12,
  rel_widths = NULL
)
```

Arguments

`eci_overall`	Optional data frame returned by ‘compute_eci()‘.
`eci_lag`	Optional object returned by ‘compute_ecilag()‘. A list with ‘by_lag_samples‘ and/or ‘by_lag‘ is handled internally. A lag-level data frame is also accepted.
`overall_col`	Character scalar selecting the ‘compute_eci()‘ column used on the x-axis. One of ‘"ECI_weighted"‘, ‘"ECI_raw"‘, or ‘"ECI_raw_centered"‘. Default is ‘"ECI_weighted"‘.
`lag_col`	Character scalar selecting the lag-specific ECI column from ‘compute_ecilag()‘ used on the y-axis. One of ‘"ECI_weighted"‘, ‘"ECI_percent"‘, or ‘"ECI_absolute"‘. Default is ‘"ECI_weighted"‘.
`facet_scales`	Character scalar controlling facet scales. One of ‘"fixed"‘, ‘"free"‘, ‘"free_x"‘, or ‘"free_y"‘. Default is ‘"free_y"‘.
`overall_nrow`	Positive integer number of facet rows for the overall ECI panel. Default is ‘3‘.
`lag_ncol`	Positive integer number of facet columns for lag panels. Default is ‘1‘.
`sample_smooth`	Logical scalar. If ‘TRUE‘, draw-level curves are shown with ‘geom_smooth(se = FALSE)‘, reproducing the original manual plot. If ‘FALSE‘, draw-level curves are connected with ‘geom_line()‘. Summary outputs are always displayed with a smoothed central curve and, when available, dashed smoothed lower and upper interval curves.
`smooth_method`	Optional smoothing method passed to ‘geom_smooth()‘. The default ‘NULL‘ lets ggplot2 select its standard method.
`smooth_formula`	Optional formula passed to ‘geom_smooth()‘. Default is ‘NULL‘.
`linewidth`	Positive finite line width. Default is ‘0.7‘.
`alpha`	Finite number in ‘[0, 1]‘ retained for backward compatibility with earlier ribbon-based summary plots. Summary uncertainty is now represented by dashed smoothed interval curves. Default is ‘0.20‘.
`overall_x_label`, `overall_y_label`, `lag_x_label`, `lag_y_label`	exposure_y_label Axis labels.

`exposure_y_label`
: Optional character string defining the y-axis label used for the exposure panel. If ‘NULL‘, a default label is generated.

`exposure_colors`
: Character vector of colors used for exposure- difference curves. Named vectors are supported. The default reproduces the original manual palette.

`base_size`
: Positive finite base font size. Default is ‘10‘.

`panel_labels`
: Character vector of length two used when both perspectives are combined. Default is ‘c("(a)", "(b)")‘.

`label_size`
: Positive finite panel-label size. Default is ‘12‘.

`rel_widths`
: Optional positive numeric vector controlling relative panel widths. By default, all displayed panels have equal width. Supply two values when two panels are displayed or three values when the overall ECI, lag contribution, and exposure-difference panels are displayed together.

Details

The function recognizes draw-level output (‘output = "samples"‘) through a ‘sample‘ column and summary output (‘output = "summary"‘) through the corresponding estimate and interval columns. For ‘compute_ecilag()‘ objects, ‘by_lag_samples‘ and ‘by_lag‘ are extracted internally.

Value

A ggplot/cowplot object. When both inputs are supplied, the returned object combines the overall ECI panel and the lag-specific panel. When only ‘eci_lag‘ is supplied, contribution and exposure-difference panels are combined. Component plots are stored in the ‘epiexposure_plot_components‘ attribute.

---

plot_effects *Plot lag-specific or period-specific effects*

Description

Plots lag-specific effect surfaces or period-specific exposure-response curves from an object returned by ‘summarise_effects()‘.

Usage

```
plot_effects(
  data,
  scale = c("lag", "period"),
  vars = NULL,
  metric = c("effect", "delta"),
  delta_multiplier = 100,
  metric_labels = c(effect = "Effect", delta = "Delta"),
  ylab = NULL,
```

```
  metric_ncol = 1,
  vars_ncol = NULL,
  effect_palette = c(low = "#3B4CC0", mid = "white", high = "#B40426"),
  delta_palette = NULL,
  delta_viridis_option = "D",
  clip_quantiles = c(0.02, 0.98),
  effect_breaks = c(-15, -10, -5, 0, 5, 10, 15),
  delta_bins = 10,
  lag_reverse = TRUE,
  lag_xlab = "Lag",
  legend_position = c(effect = "top", delta = "bottom"),
  lag_theme = "bw",
  lag_base_size = 12,
  axis_title_size = 12,
  axis_text_size = 10,
  contour_colour = "black",
  contour_linewidth = 0.25,
  zero_linewidth = 0.8,
  effect_zero_text_size = 3.2,
  delta_zero_text_size = 3,
  interpolate = TRUE,
  period_order = NULL,
  vars_order = NULL,
  period_response = "eta",
  period_exclude_samples = 10,
  period_palette = NULL,
  period_viridis_option = "D",
  period_xlab = "Exposure",
  period_ylab = "Response",
  period_theme = "bw",
  period_base_size = 11,
  period_linewidth = 1,
  period_interval_linetype = 2,
  period_interval_linewidth = 0.5,
  period_show_legend = FALSE,
  panel_labels = c("(a)", "(b)"),
  label_size = 12,
  output = c("plot", "list")
)
```

Arguments

data	Data frame returned by ‘summarise_effects()‘.
scale	Plot scale. Use only ‘"lag"‘ or ‘"period"‘.
vars	Character vector containing variables to plot. ‘NULL‘ uses all available variables.
metric	For lag plots, one or both of ‘c("effect", "delta")‘.

`delta_multiplier`
: Multiplier applied to ‘delta‘ before plotting.

`metric_labels`
: Named labels for lag color legends.

`ylab`
: Lag y-axis labels. Prefer a named vector keyed by variable.

`metric_ncol`
: Number of metric blocks per row. Default ‘1‘ places effect above delta.

`vars_ncol`
: Number of variable panels per metric block. ‘NULL‘ uses all selected variables in one row.

`effect_palette`
: Three colors representing low, middle, and high values.

`delta_palette`
: ‘NULL‘ uses viridis; otherwise a vector of at least two colors.

`delta_viridis_option`
: Viridis option used when ‘delta_palette = NULL‘.

`clip_quantiles`
: Quantiles used to clip lag fill values.

`effect_breaks`
: Contour breaks for ‘effect‘.

`delta_bins`
: Number of contour bins for ‘delta‘.

`lag_reverse`
: Logical; reverse the lag axis.

`lag_xlab`
: X-axis label for lag plots.

`legend_position`
: Named positions for effect and delta legends.

`lag_theme`
: One of ‘"bw"‘, ‘"minimal"‘, ‘"classic"‘, or a ggplot2 theme.

`lag_base_size`
: Base font size for the lag theme.

`axis_title_size`
: Axis-title size for lag plots.

`axis_text_size`
: Axis-text size for lag plots.

`contour_colour`
: Contour color.

`contour_linewidth`
: Contour line width.

`zero_linewidth`
: Delta zero-contour line width.

`effect_zero_text_size`
: Effect zero-contour label size.

`delta_zero_text_size`
: Delta zero-contour label size.

`interpolate`
: Passed to ‘ggplot2::geom_raster()‘.

`period_order`
: Period facet order. ‘NULL‘ reverses names following the ‘PeriodN‘ convention.

`vars_order`
: Period variable order. ‘NULL‘ follows ‘vars‘.

`period_response`
: Response displayed in period plots. Supported values are ‘"eta"‘ or ‘"linear"‘ for the linear-predictor contrast; ‘"effect"‘ for the effect column as stored; ‘"exponentiated"‘ (the alias ‘"exponential"‘ is also accepted) for an effect generated with ‘effect_measure = "exponentiated"‘; ‘"percent"‘ for an effect generated with ‘effect_measure = "percent"‘; and ‘"baseline"‘, ‘"predicted"‘, or ‘"delta"‘ for response-scale quantities. For summary input, corresponding columns named ‘<response>_lower‘ and ‘<response>_upper‘, when present, are drawn as dashed uncertainty curves. Because exponentiated and percent results are both stored in ‘effect‘, their aliases validate the ‘epiexposure_effect_measure‘ attribute created by ‘summarise_effects()‘.

`period_exclude_samples`
: Samples removed from period draw-level plots. Default ‘10‘ preserves the original display; use ‘NULL‘ to retain all samples.

`period_palette`
: ‘NULL‘ uses viridis; otherwise a vector of at least two colors.

`period_viridis_option`
: Viridis option for period curves.

`period_xlab`
: X-axis label for period plots.

`period_ylab`
: Y-axis label for period plots.

`period_theme`
: One of ‘"bw"‘, ‘"minimal"‘, ‘"classic"‘, or a ggplot2 theme.

`period_base_size`
: Base size for the period theme.

`period_linewidth`
: Line width passed to ‘ggplot2::geom_smooth()‘.

`period_interval_linetype`
: Line type used for lower and upper summary uncertainty curves.

`period_interval_linewidth`
: Line width used for lower and upper summary uncertainty curves.

`period_show_legend`
: Logical; show the period color legend.

`panel_labels`
: Character vector used to label lag-specific metric blocks.

`label_size`
: Positive finite size used for ‘panel_labels‘.

`output`
: Either ‘"plot"‘ or ‘"list"‘. For lag plots, ‘"list"‘ returns every component panel; for period plots, it returns ‘list(period = plot)‘.

Details

For period plots, draw-level input is recognized by the presence of a ‘sample‘ column. One smoothed curve is then drawn for each parameter draw. Otherwise, the data are treated as deterministic or summary output. The selected central response is drawn as a solid black curve and, when matching lower and upper columns are available, the uncertainty limits are drawn as dashed black curves. The bounds are used exactly as returned by ‘summarise_effects()‘; ‘plot_effects()‘ does not recalculate uncertainty.

Value

A ggplot/cowplot object, or a list of ggplots when ‘output = "list"‘.

plot_losses *Plot simulated yield and economic losses*

Description

Creates either coordinated heatmaps of yield and economic losses or a regression-style visualization from an object returned by ‘simulate_losses()‘.

Usage

```
plot_losses(
  data,
  x,
  type = c("heatmap", "regression"),
  y = "att_yield",
  yield_loss = "yield_loss",
  economic_loss = "econ_loss",
  reg_y = "ap_yield",
  att_yield = NULL,
  facet = NULL,
  plots = c("yield", "economic"),
  ncol = NULL,
  nrow = NULL,
  facet_ncol = 4L,
  facet_nrow = NULL,
  yield_fill_scale = NULL,
  economic_fill_scale = NULL,
  yield_fill_label = expression(bold(kg ~ ha^{
     -1
 })),
  economic_fill_label = expression(bold(USD ~ ha^{
     -1
 })),
  x_breaks = NULL,
  y_breaks = NULL,
  x_expand = c(0.02, 0.02),
  y_expand = c(0.02, 0.02),
  x_limits = NULL,
  y_limits = NULL,
  x_label = NULL,
  y_label = "Attainable yield (kg/ha)",
  reg_y_label = NULL,
  yield_title = NULL,
  economic_title = NULL,
  show_yield_x_text = FALSE,
  show_yield_x_label = FALSE,
  show_economic_strips = FALSE,
```

```
  raster_interpolate = FALSE,
  raster_args = list(),
  yield_raster_args = list(),
  economic_raster_args = list(),
  theme = NULL,
  base_size = 10,
  text_face = "bold",
  axis_title_size = 12,
  strip_background = ggplot2::element_rect(fill = "white", colour = "black"),
  legend_position = "right",
  tag_levels = "a",
  tag_prefix = "(",
  tag_suffix = ")",
  tag_size = 12,
  tag_face = "bold",
  tag_position = c(0.01, 1.02),
  collect_guides = FALSE,
  yield_add = NULL,
  economic_add = NULL,
  combined_add = NULL,
  ...
)
```

Arguments

data	A non-empty data frame returned by ‘simulate_losses()‘.
x	Character scalar naming the variable displayed on the x-axis.
type	Character scalar selecting ‘"heatmap"‘ or ‘"regression"‘. Default is ‘"heatmap"‘.
y	Character scalar naming the y-axis variable for heatmaps. Default is ‘"att_yield"‘, the shortened output name returned by the current ‘simulate_losses()‘.
yield_loss	Character scalar naming the yield-loss column used by the heatmap. Default is ‘"yield_loss"‘.
economic_loss	Character scalar naming the economic-loss column used by the heatmap. Default is ‘"econ_loss"‘.
reg_y	Character scalar naming the y-axis variable when ‘type = "regression"‘. Default is ‘"ap_yield"‘.
att_yield	Optional finite numeric scalar used only with ‘type = "regression"‘. When supplied, rows are filtered to the selected ‘att_yield‘ value before plotting.
facet	Optional character vector naming scenario variables used as facets. For heatmaps, ‘facet_wrap()‘ is used as in the original function. For regression plots, the first facet variable is placed in rows and any remaining facet variables are placed in columns together with the automatically generated ‘yield_model‘ facet. Thus ‘facet = c("tmean", "wetness")‘ reproduces ‘facet_grid(rows = vars(tmean), cols = vars(wetness, yield_model))‘.
plots	Character vector selecting ‘"yield"‘, ‘"economic"‘, or both. Used only when ‘type = "heatmap"‘ and ignored for regression plots.

`ncol`, `nrow` Optional positive integers controlling the arrangement of the main heatmap plots. When both are 'NULL' and both plots are requested, 'nrow = 2' is used.

`facet_ncol`, `facet_nrow`
Optional positive integers controlling the arrangement of heatmap facets.

`yield_fill_scale`, `economic_fill_scale`
Optional continuous ggplot2 fill scales for heatmaps. When 'NULL', 'ggplot2::scale_fill_viridis_b()' is used.

`yield_fill_label`, `economic_fill_label`
Legend titles used by the default heatmap fill scales.

`x_breaks`, `y_breaks`
Optional numeric axis breaks.

`x_expand`, `y_expand`
Numeric expansion vectors passed to the axes.

`x_limits`, `y_limits`
Optional numeric vectors of length two.

`x_label` Axis label. If 'NULL', a title-case label is generated from 'x'.

`y_label` Y-axis label used for heatmaps.

`reg_y_label` Optional y-axis label used for regression plots. If 'NULL', a label is generated from 'reg_y', with informative defaults for standard 'simulate_losses()' output columns.

`yield_title`, `economic_title`
Optional heatmap panel titles.

`show_yield_x_text`
Logical. Show x-axis text and ticks in the yield heatmap panel. Default is 'FALSE'.

`show_yield_x_label`
Logical. Show the x-axis title in the yield heatmap panel. Default is 'FALSE'.

`show_economic_strips`
Logical. Show facet strips in the economic heatmap panel. Default is 'FALSE'.

`raster_interpolate`
Logical passed to 'ggplot2::geom_raster()'.

`raster_args` Named list of arguments passed to both raster layers.

`yield_raster_args`, `economic_raster_args`
Named lists of panel-specific raster arguments. Panel-specific arguments override 'raster_args' and '...'.

`theme` A ggplot2 theme. The default is 'ggplot2::theme_bw(base_size = base_size)'.

`base_size` Base font size used by the default theme.

`text_face` Font face used for common heatmap text.

`axis_title_size`
Axis-title size.

`strip_background`
Facet-strip background in heatmaps and regression plots.

`legend_position`
Legend position used in heatmaps. Regression plots intentionally suppress the legend, matching the requested display.

`tag_levels` Tag sequence passed to ‘patchwork::plot_annotation()‘.

`tag_prefix`, `tag_suffix`
Prefix and suffix for plot tags.

`tag_size`, `tag_face`, `tag_position`
Appearance and position of plot tags.

`collect_guides` Logical. Collect compatible guides with patchwork.

`yield_add`, `economic_add`
Optional ggplot2 component or list of components added to the corresponding heatmap panel.

`combined_add` Optional patchwork-compatible component added to the combined heatmap.

`...` Additional named arguments passed to both heatmap raster layers.

Details

With ‘type = "heatmap"‘ (default), the function preserves the original raster-plot workflow. Yield loss and economic loss can be displayed alone or together, and scenario variables can be shown as facets.

With ‘type = "regression"‘, ‘plots‘ is not used and a single regression plot is returned. The response plotted on the y-axis is selected with ‘reg_y‘. Optionally, ‘att_yield‘ filters the data to one attainable-yield scenario before plotting. A ‘yield_model‘ facet is generated automatically from ‘intercept_sim‘ and ‘slope_sim‘.

Regression input is recognized as draw-level when a ‘sample‘ column is present. In that case, one ‘geom_smooth(se = FALSE)‘ curve is drawn for each sample using ‘group = sample‘ and ‘color = sample‘. Otherwise, the data are treated as summary output. The central response is drawn as a solid black smooth and, when ‘<reg_y>_lower‘ and ‘<reg_y>_upper‘ are present, the lower and upper curves are drawn as dashed black smooths.

Regression mode

For summary-style data, the function plots the central column selected by ‘reg_y‘. If columns named ‘<reg_y>_lower‘ and ‘<reg_y>_upper‘ are available, they are plotted using dashed black ‘geom_smooth(se = FALSE)‘ curves. These bounds are used exactly as returned by ‘simulate_losses()‘; no interval is recalculated by ‘plot_losses()‘.

For draw-level data containing ‘sample‘, the function plots one smooth curve per sample and maps both ‘group‘ and ‘color‘ to ‘sample‘. No summary interval is calculated.

Value

With ‘type = "regression"‘, a ‘ggplot‘ object. With ‘type = "heatmap"‘, a ‘patchwork‘ object when both heatmaps are requested or a ‘ggplot‘ object when only one is requested. Heatmap panels are retained as ‘"yield_plot"‘ and ‘"economic_plot"‘ attributes.

Examples

```
# Small simulated loss surface
losses <- expand.grid(
  rain = seq(0, 12, by = 3),
  att_yield = seq(4000, 8000, by = 1000)
```

```
)

losses$yield_loss <- with(
  losses,
  0.04 * rain * att_yield
)

losses$econ_loss <- losses$yield_loss * 0.25

# Heatmap of yield loss
loss_heatmap <- plot_losses(
  losses,
  x = "rain",
  type = "heatmap",
  plots = "yield",
  x_breaks = seq(0, 12, by = 3),
  y_breaks = seq(4000, 8000, by = 1000),
  x_label = "Daily precipitation (mm)",
  show_yield_x_text = TRUE,
  show_yield_x_label = TRUE
)

# Summary-style regression data
regression_losses <- data.frame(
  rain = rep(seq(0, 12, length.out = 20), 2),
  att_yield = 6000,
  intercept_sim = rep(c(0.95, 0.90), each = 20),
  slope_sim = rep(c(-0.015, -0.025), each = 20)
)

regression_losses$ap_yield <- with(
  regression_losses,
  att_yield * (intercept_sim + slope_sim * rain)
)

regression_losses$ap_yield_lower <-
  regression_losses$ap_yield - 150

regression_losses$ap_yield_upper <-
  regression_losses$ap_yield + 150

summary_regression_plot <- plot_losses(
  regression_losses,
  x = "rain",
  type = "regression",
  reg_y = "ap_yield",
  att_yield = 6000,
  x_label = "Daily precipitation (mm)"
)

# Draw-level regression data
losses_samples <- expand.grid(
  rain = seq(0, 12, length.out = 20),
```

```
  sample = 1:3
)

losses_samples$att_yield <- 6000
losses_samples$intercept_sim <- 0.95
losses_samples$slope_sim <- -0.02

losses_samples$ap_yield <- with(
  losses_samples,
  att_yield *
    (intercept_sim + slope_sim * rain) +
    sample * 25
)

sample_regression_plot <- plot_losses(
  losses_samples,
  x = "rain",
  type = "regression",
  reg_y = "ap_yield",
  att_yield = 6000,
  x_label = "Daily precipitation (mm)"
)

stopifnot(
  inherits(loss_heatmap, "ggplot"),
  inherits(summary_regression_plot, "ggplot"),
  inherits(sample_regression_plot, "ggplot")
)
```

`plot_performance` *Plot out-of-fold model performance*

Description

Creates performance plots for models evaluated by ‘find_bestfit()‘. For non-binary outcomes, the default is an observed-versus-predicted plot. For binomial outcomes, the default is a receiver operating characteristic (ROC) curve.

Usage

```
plot_performance(
  object,
  model_id = NULL,
  metrics = "CCC",
  type = c("auto", "observed_predicted", "roc", "threshold", "confusion"),
  threshold = NULL,
  x = "observed",
  y = "predicted",
```

```
  scale_factor = 1,
  ncol = NULL,
  metric_digits = 2L,
  x_lab = NULL,
  y_lab = NULL,
  show_fit = TRUE,
  fit_se = FALSE,
  fit_color = "orange",
  show_identity = TRUE,
  identity_color = "red",
  identity_linetype = "dashed",
  label_x = -Inf,
  label_y = Inf,
  label_hjust = -0.1,
  label_vjust = 1.2,
  label_size = 3.5,
  base_size = 11,
  point_args = list(),
  ...
)
```

Arguments

object	A non-empty data frame returned by ‘find_bestfit()‘. It must contain a unique ‘model_id‘ column, the requested metric columns, and retained out-of-fold predictions.
model_id	Optional vector of model identifiers to plot. If ‘NULL‘, every retained model with available predictions is plotted.
metrics	Unique character vector naming validation metrics stored as numeric columns of ‘object‘. Metrics are displayed in the supplied order. Common non-binary metrics include ‘"CCC"‘, ‘"Cb"‘, ‘"rho"‘, ‘"RMSE"‘, and ‘"MAE"‘. Binomial metrics include ‘"ROC_AUC"‘, ‘"Brier"‘, ‘"LogLoss"‘, ‘"Accuracy"‘, ‘"Balanced_Accuracy"‘, ‘"Sensitivity"‘, ‘"Specificity"‘, ‘"Precision"‘, ‘"F1"‘, and ‘"MCC"‘.
type	Character. Plot type selected from ‘"auto"‘, ‘"observed_predicted"‘, ‘"roc"‘, ‘"threshold"‘, or ‘"confusion"‘. ‘"auto"‘ uses an ROC curve for binomial outcomes and an observed-versus-predicted plot otherwise.
threshold	Optional probability threshold strictly between 0 and 1. If ‘NULL‘, ‘attr(object, "threshold")‘ is used when available; otherwise ‘0.5‘ is used. It is used by ‘"threshold"‘ and ‘"confusion"‘ plots and identifies the selected operating point on ROC curves.
x	Character scalar naming the observed-value column in prediction data. Default is ‘"observed"‘.
y	Character scalar naming the predicted-value column in prediction data. For binomial outcomes, this column must contain probabilities in ‘[0, 1]‘. Default is ‘"predicted"‘.
scale_factor	Finite non-zero numeric scalar multiplying both axes only for ‘type = "observed_predicted"‘. Use ‘100‘ to display proportions as percentages.

`ncol`	Optional positive integer passed to ‘ggplot2::facet_wrap()‘.
`metric_digits`	Non-negative integer controlling metric-label precision.
`x_lab`, `y_lab`	Optional axis labels. Plot-type-specific labels are used when either argument is ‘NULL‘.
`show_fit`	Logical. For observed-versus-predicted plots, add a linear fitted line.
`fit_se`	Logical. Display an uncertainty band around the fitted line.
`fit_color`	Color of the fitted regression line.
`show_identity`	Logical. For observed-versus-predicted plots, add the 1:1 line. For ROC curves, add the no-discrimination diagonal.
`identity_color`	Color of the identity or no-discrimination line.
`identity_linetype`	Line type of the identity or no-discrimination line.
`label_x`, `label_y`	Numeric positions for metric labels. Infinite defaults place labels close to a panel corner where supported.
`label_hjust`, `label_vjust`	Horizontal and vertical justification for metric labels.
`label_size`	Numeric text size for metric labels and confusion-matrix counts.
`base_size`	Base font size passed to ‘ggplot2::theme_bw()‘.
`point_args`	Optional named list of additional arguments passed to ‘ggplot2::geom_point()‘ for observed-versus-predicted plots.
`...`	Additional named arguments passed to ‘ggplot2::geom_point()‘ for observed-versus-predicted plots.

Details

The function retrieves out-of-fold predictions from the ‘"predictions_by_model"‘ attribute, or from the ‘"predictions"‘ attribute when the model-specific list is unavailable. Model-level validation metrics are retrieved from columns of the main ‘find_bestfit()‘ result.

‘plot_performance()‘ does not refit models or recalculate their stored validation metrics. Predictions are obtained from:

‘attr(object, "predictions_by_model")‘

or, when that list is unavailable:

‘attr(object, "predictions")‘

For binomial outcomes, ROC curves are reconstructed directly from the observed classes and out-of-fold predicted probabilities. Sensitivity and specificity are calculated over the distinct probability thresholds represented in the selected predictions.

The ROC curve uses false-positive rate (‘1 - specificity‘) on the x-axis and sensitivity on the y-axis. A highlighted point identifies performance at ‘threshold‘.

Threshold plots display sensitivity and specificity across probability thresholds and add a vertical line at the selected threshold.

Confusion matrices classify probabilities using ‘predicted >= threshold‘ and report classification counts.

The ROC and threshold plots require both observed classes. The observed values must be coded as ‘0‘ and ‘1‘, and predicted values must be finite probabilities in ‘[0, 1]‘.

Value

A ‘ggplot‘ object that can be extended with additional ggplot2 layers, scales, labels, or themes.

Examples

```
performance_results <- data.frame(
  model_id = c("model_1", "model_2"),
  CCC = c(0.91, 0.86),
  Cb = c(0.96, 0.93),
  rho = c(0.95, 0.92),
  RMSE = c(0.07, 0.10),
  MAE = c(0.05, 0.08)
)

continuous_predictions <- list(
  model_1 = data.frame(
    observed = c(0.05, 0.12, 0.20, 0.31, 0.43, 0.55),
    predicted = c(0.06, 0.14, 0.18, 0.29, 0.46, 0.53)
  ),
  model_2 = data.frame(
    observed = c(0.05, 0.12, 0.20, 0.31, 0.43, 0.55),
    predicted = c(0.09, 0.16, 0.24, 0.27, 0.39, 0.49)
  )
)

attr(
  performance_results,
  "predictions_by_model"
) <- continuous_predictions

continuous_plot <- plot_performance(
  performance_results,
  metrics = c("CCC", "Cb", "rho"),
  scale_factor = 100,
  x_lab = "Observed (%)",
  y_lab = "Predicted (%)",
  size = 2.5,
  alpha = 0.8
)

binary_results <- data.frame(
  model_id = c("model_1", "model_2"),
  ROC_AUC = c(0.88, 0.82),
  Brier = c(0.14, 0.18)
)

binary_predictions <- list(
  model_1 = data.frame(
    observed = c(0, 0, 0, 1, 1, 1),
```

```
    predicted = c(0.08, 0.21, 0.39, 0.58, 0.77, 0.91)
  ),
  model_2 = data.frame(
    observed = c(0, 0, 0, 1, 1, 1),
    predicted = c(0.18, 0.31, 0.48, 0.52, 0.68, 0.79)
  )
)

attr(binary_results, "outcome_type") <- "binary"
attr(binary_results, "threshold") <- 0.5
attr(binary_results, "predictions_by_model") <- binary_predictions

roc_plot <- plot_performance(
  binary_results,
  metrics = c("ROC_AUC", "Brier")
)

threshold_plot <- plot_performance(
  binary_results,
  model_id = "model_1",
  metrics = "ROC_AUC",
  type = "threshold"
)

confusion_plot <- plot_performance(
  binary_results,
  model_id = "model_1",
  metrics = c("ROC_AUC", "Brier"),
  type = "confusion"
)

stopifnot(
  inherits(continuous_plot, "ggplot"),
  inherits(roc_plot, "ggplot"),
  inherits(threshold_plot, "ggplot"),
  inherits(confusion_plot, "ggplot")
)
```

`plot_scenarios` *Plot predictions across epidemiological scenarios*

Description

Plot predictions generated by ‘simulate_scenarios()‘ while respecting both the scenario design created by ‘simulate_ranges()‘ (‘"grid"‘ or ‘"paired"‘) and the prediction output returned by ‘simulate_scenarios()‘ (‘"samples"‘ or ‘"summary"‘).

Usage

```
plot_scenarios(
  data,
  x,
  scenario_type = c("grid", "paired"),
  output = c("auto", "samples", "summary"),
  mode = NULL,
  y = "prediction",
  facet = NULL,
  sample = "sample",
  lower = NULL,
  upper = NULL,
  scale_factor = 1,
  facet_ncol = NULL,
  facet_nrow = NULL,
  color_scale = NULL,
  color_label = "Sample",
  line_color = "black",
  ribbon_fill = "black",
  ribbon_alpha = 1,
  show_ribbon = TRUE,
  smooth = TRUE,
  smooth_method = NULL,
  smooth_formula = NULL,
  smooth_se = FALSE,
  line_width = NULL,
  line_alpha = 1,
  paired_samples_geom = c("boxplot", "jitter"),
  boxplot_color = "black",
  boxplot_fill = "white",
  boxplot_width = NULL,
  jitter_color = "black",
  jitter_width = 0.12,
  jitter_alpha = 0.65,
  jitter_size = NULL,
  pointrange_color = "black",
  point_size = NULL,
  pointrange_linewidth = NULL,
  paired_x_discrete = TRUE,
  x_breaks = ggplot2::waiver(),
  y_breaks = ggplot2::waiver(),
  x_limits = NULL,
  y_limits = NULL,
  x_expand = NULL,
  y_expand = ggplot2::waiver(),
  x_label = "Exposure",
  y_label = "Predicted",
  title = NULL,
```

```
  subtitle = NULL,
  caption = NULL,
  theme = NULL,
  base_size = 10,
  text_face = "bold",
  axis_title_size = 12,
  strip_background = ggplot2::element_rect(fill = "white", colour = "black"),
  legend_position = "none",
  line_args = list(),
  ribbon_args = list(),
  boxplot_args = list(),
  jitter_args = list(),
  pointrange_args = list(),
  point_args = list(),
  add = NULL,
  ...
)
```

Arguments

data	A non-empty data frame returned by ‘simulate_scenarios()‘.
x	Character scalar naming the scenario variable shown on the x-axis.
scenario_type	Character. Scenario design used to generate the scenarios in ‘simulate_ranges()‘. Must be ‘"grid"‘ or ‘"paired"‘. Default is ‘"grid"‘.
output	Character. Prediction output represented by ‘data‘: ‘"auto"‘, ‘"samples"‘, or ‘"summary"‘. With ‘"auto"‘ (default), the function identifies ‘"samples"‘ when the column named by ‘sample‘ is present; otherwise it uses ‘"summary"‘. This mirrors the two output structures returned by ‘simulate_scenarios()‘.
mode	Deprecated compatibility alias for ‘output‘. It is retained so calls written for earlier versions of ‘plot_scenarios()‘ do not break. New code should use ‘output‘. Do not supply both ‘output‘ and ‘mode‘.
y	Character scalar naming the prediction column. Default is ‘"prediction"‘.
facet	Optional character vector naming variables used by ‘ggplot2::facet_wrap()‘, for example ‘c("wetness", "tmean")‘.
sample	Character scalar naming the sample identifier in ‘output = "samples"‘. Default is ‘"sample"‘.
lower, upper	Optional character scalars naming the lower and upper uncertainty-limit columns in ‘output = "summary"‘. When left as ‘NULL‘, ‘plot_scenarios()‘ looks first for ‘"prediction_lower"‘ and ‘"prediction_upper"‘, respectively, and then for ‘"lower"‘ and ‘"upper"‘. These arguments are not used for ‘output = "samples"‘ because sample-level output contains individual prediction draws rather than pre-computed lower/upper limits.
scale_factor	Finite numeric multiplier applied to predictions and, when present, lower and upper limits. Use ‘100‘ to display proportions as percentages.

`facet_ncol`, `facet_nrow`
: Optional positive integers controlling the facet arrangement.

`color_scale`
: Optional ggplot2 color scale used for sample-specific grid trajectories. When ‘NULL‘, ‘ggplot2::scale_color_viridis_c()‘ is used.

`color_label`
: Legend title used by the default grid color scale.

`line_color`
: Color of the central trajectory for grid-summary plots.

`ribbon_fill`, `ribbon_alpha`
: Legacy argument names retained for backward compatibility. In grid-summary plots, ‘ribbon_fill‘ now controls the color of the dashed lower/upper interval curves and ‘ribbon_alpha‘ controls their transparency. Defaults reproduce the black dashed interval curves used by ‘plot_losses()‘.

`show_ribbon`
: Logical. Legacy argument name retained for backward compatibility. If ‘TRUE‘, a grid-summary plot shows dashed lower/upper uncertainty curves when interval columns are available.

`smooth`
: Logical. For grid plots, use ‘ggplot2::geom_smooth()‘ when ‘TRUE‘ (default) or ‘ggplot2::geom_line()‘ when ‘FALSE‘.

`smooth_method`
: Optional smoothing method passed to ‘ggplot2::geom_smooth()‘. When ‘NULL‘, ggplot2 chooses its default method.

`smooth_formula`
: Optional formula passed to ‘ggplot2::geom_smooth()‘.

`smooth_se`
: Logical. Display the smoothing standard-error band produced by ‘geom_smooth()‘. Default is ‘FALSE‘. This is distinct from the prediction uncertainty represented by ‘lower‘ and ‘upper‘.

`line_width`
: Optional positive line width for grid trajectories. When ‘NULL‘, ggplot2 uses its default.

`line_alpha`
: Transparency of grid trajectories, between 0 and 1.

`paired_samples_geom`
: Character. Geometry used when ‘scenario_type = "paired"‘ and ‘output = "samples"‘. ‘"boxplot"‘ (default) displays the distribution of the sample predictions for each paired scenario; ‘"jitter"‘ displays the individual sample predictions.

`boxplot_color`, `boxplot_fill`
: Color and fill used by the default paired sample boxplot.

`boxplot_width`
: Optional positive box width for paired sample boxplots. When ‘NULL‘, ggplot2 uses its default.

`jitter_color`
: Color used for paired sample jittered points.

`jitter_width`
: Horizontal jitter width for paired sample points.

`jitter_alpha`
: Transparency of paired sample jittered points.

`jitter_size`
: Optional positive size of paired sample jittered points. When ‘NULL‘, ggplot2 uses its default.

`pointrange_color`
: Color used by ‘ggplot2::geom_pointrange()‘ for paired summary plots.

`point_size`
: Optional positive point size for paired-summary plots. When ‘NULL‘, ggplot2 uses its default.

`pointrange_linewidth`
: Optional positive line width for the uncertainty interval in paired-summary plots. When ‘NULL‘, ggplot2 uses its default.

`paired_x_discrete`
: Logical. For paired scenarios, treat the x variable as discrete when ‘TRUE‘ (default), matching a manual mapping such as ‘aes(as.factor(rain), ...)‘. If ‘FALSE‘, ‘x‘ must be numeric and a continuous x-axis is used.

`x_breaks, y_breaks`
: Axis breaks. Defaults to ‘ggplot2::waiver()‘, which lets ggplot2 choose breaks automatically and preserves axis tick labels and the ‘theme_bw()‘ panel grid. Use ‘NULL‘ only to intentionally remove the corresponding breaks and grid lines.

`x_limits, y_limits`
: Optional axis limits.

  For grid plots and paired plots with ‘paired_x_discrete = FALSE‘, ‘x_limits‘ must be a finite numeric vector of length two.

  For paired plots with ‘paired_x_discrete = TRUE‘, ‘x_limits‘ can instead be a vector of observed x levels used to subset and/or order the displayed categories.

`x_expand, y_expand`
: Axis expansion specifications.

  When ‘x_expand = NULL‘, continuous x-axes use ‘ggplot2::expansion(mult = c(0.01, 0.01))‘, matching the original EpiExposure scenario figure, while discrete paired x-axes retain ggplot2’s default discrete expansion.

`x_label, y_label`
: Axis titles. Defaults are ‘"Exposure"‘ and ‘"Predicted"‘.

`title, subtitle, caption`
: Optional plot annotations.

`theme`
: Optional ggplot2 theme added after the default ‘theme_bw()‘ and EpiExposure text settings, allowing users to override them.

`base_size`
: Base text size. Default is 10.

`text_face`
: Common text face. Default is ‘"bold"‘.

`axis_title_size`
: Axis-title size. Default is 12.

`strip_background`
: Facet-strip background element.

`legend_position`
: Legend position. Default is ‘"none"‘.

`line_args`
: Optional named list of additional arguments passed to the active grid trajectory layer (‘geom_smooth()‘ or ‘geom_line()‘).

`ribbon_args`
: Optional named list of additional arguments passed to both lower and upper uncertainty-curve layers in grid-summary mode. The legacy argument name is retained for backward compatibility.

`boxplot_args`
: Optional named list of additional arguments passed to ‘ggplot2::geom_boxplot()‘ in paired-samples mode.

`jitter_args`
: Optional named list of additional arguments passed to ‘ggplot2::geom_jitter()‘ when ‘paired_samples_geom = "jitter"‘.

`pointrange_args`
: Optional named list of additional arguments passed to ‘ggplot2::geom_pointrange()‘ in paired-summary mode.

`point_args`
: Optional named list of additional arguments passed to ‘ggplot2::geom_point()‘ when summary predictions do not contain lower/upper uncertainty limits.

`add`
: Optional ggplot2 component or list of components added last.

`...`
: Additional named arguments passed to the principal geometry used for the selected combination of ‘scenario_type‘ and ‘output‘.

Details

The default plotting behavior is:

* ‘scenario_type = "grid"‘ + ‘output = "samples"‘: sample-specific trajectories using ‘geom_smooth()‘ (or ‘geom_line()‘). * ‘scenario_type = "grid"‘ + ‘output = "summary"‘: one central trajectory, optionally with dashed lower/upper uncertainty curves. * ‘scenario_type = "paired"‘ + ‘output = "samples"‘: the distribution of sample predictions at each paired scenario using ‘geom_boxplot()‘ by default. * ‘scenario_type = "paired"‘ + ‘output = "summary"‘: point estimates and lower/upper uncertainty intervals using ‘geom_pointrange()‘.

‘scenario_type‘ and ‘output‘ describe different aspects of the data and should not be conflated:

* ‘scenario_type‘ describes how exposure scenarios were constructed by ‘simulate_ranges()‘. * ‘output‘ describes how predictions were returned by ‘simulate_scenarios()‘.

Consequently, all four combinations are valid.

For sample-level output, ‘lower‘ and ‘upper‘ are intentionally ignored. The uncertainty is represented by the multiple sample predictions themselves.

For summary output, lower and upper limits are used when available:

* grid + summary: the limits define dashed lower/upper uncertainty curves; * paired + summary: the limits define the vertical point-range.

If summary output does not contain lower/upper limits, the function still plots the central prediction: a grid trajectory for grid scenarios or points for paired scenarios.

For paired sample output, ‘geom_boxplot()‘ is the default because several prediction samples correspond to the same paired scenario value. The user can instead request the individual sample predictions with ‘paired_samples_geom = "jitter"‘.

Value

A ‘ggplot‘ object.

Examples

```
# ------------------------------------------------------------------------
# 1. GRID + SAMPLES
# ------------------------------------------------------------------------
pred_grid_samples <- expand.grid(
  rain = seq(0, 12, by = 3),
  sample = 1:3
)
```

```
pred_grid_samples$prediction <- with(
  pred_grid_samples,
  0.05 + 0.015 * rain + 0.005 * sample
)

grid_samples_plot <- plot_scenarios(
  data = pred_grid_samples,
  x = "rain",
  scenario_type = "grid",
  output = "samples",
  smooth = FALSE,
  scale_factor = 100,
  x_label = "Daily precipitation (mm)",
  y_label = "Predicted (%)"
)

# The default output = "auto" identifies the sample column automatically.
grid_auto_plot <- plot_scenarios(
  data = pred_grid_samples,
  x = "rain",
  scenario_type = "grid",
  smooth = FALSE,
  scale_factor = 100
)

# ------------------------------------------------------------------------
# 2. GRID + SUMMARY
# ------------------------------------------------------------------------
pred_grid_summary <- data.frame(
  rain = seq(0, 12, by = 2)
)

pred_grid_summary$prediction <-
  0.05 + 0.015 * pred_grid_summary$rain

pred_grid_summary$prediction_lower <-
  pred_grid_summary$prediction - 0.02

pred_grid_summary$prediction_upper <-
  pred_grid_summary$prediction + 0.02

grid_summary_plot <- plot_scenarios(
  data = pred_grid_summary,
  x = "rain",
  scenario_type = "grid",
  output = "summary",
  smooth = FALSE,
  scale_factor = 100,
  x_label = "Daily precipitation (mm)",
  y_label = "Predicted (%)"
)
```

```
# ----------------------------------------------------------------------
# 3. PAIRED + SAMPLES
# ----------------------------------------------------------------------
pred_paired_samples <- expand.grid(
  rain = c(0, 4, 8, 12),
  sample = 1:5
)

pred_paired_samples$prediction <- with(
  pred_paired_samples,
  0.05 + 0.015 * rain + 0.004 * sample
)

paired_samples_plot <- plot_scenarios(
  data = pred_paired_samples,
  x = "rain",
  scenario_type = "paired",
  output = "samples",
  scale_factor = 100,
  x_label = "Daily precipitation (mm)",
  y_label = "Predicted (%)"
)

# ----------------------------------------------------------------------
# 4. PAIRED + SUMMARY
# ----------------------------------------------------------------------
pred_paired_summary <- data.frame(
  rain = c(0, 4, 8, 12),
  prediction = c(0.06, 0.12, 0.19, 0.25),
  prediction_lower = c(0.04, 0.09, 0.15, 0.20),
  prediction_upper = c(0.08, 0.15, 0.23, 0.30)
)

paired_summary_plot <- plot_scenarios(
  data = pred_paired_summary,
  x = "rain",
  scenario_type = "paired",
  output = "summary",
  scale_factor = 100,
  x_label = "Daily precipitation (mm)",
  y_label = "Predicted (%)"
)

stopifnot(
  inherits(grid_samples_plot, "ggplot"),
  inherits(grid_auto_plot, "ggplot"),
  inherits(grid_summary_plot, "ggplot"),
  inherits(paired_samples_plot, "ggplot"),
  inherits(paired_summary_plot, "ggplot")
)
```

`poisson_data` *Simulated poisson epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and a count plant disease outcome. The dataset is intended for demonstrating poisson models in EpiExposure.

Usage

```
poisson_data
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Non-negative count disease outcome repeated across the rows of each epidemic profile.

Details

The response is repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("poisson_data", package = "EpiExposure")
head(poisson_data)
```

predict_outcomes *Predict outcomes from fitted EpiExposure DLNM models*

Description

Predicts the population-level expected outcome associated with either user-defined chronological exposure profiles or a new longitudinal exposure data set. The function uses only the standardized model metadata and internal prediction helpers created by 'fit_epidlnm()'; family, link, cross-basis definitions, coefficient mappings, and engine behavior are never inferred again inside this public function.

Usage

```
predict_outcomes(
  fit,
  profiles = NULL,
  newdata = NULL,
  group = "epi_id",
  time = "time",
  type = c("response", "link"),
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000L,
  probs = c(0.025, 0.975),
  seed = NULL,
  extrapolation = c("warn", "error", "allow")
)
```

Arguments

fit	Fitted model returned by the current 'fit_epidlnm()'.
profiles	Optional exposure profiles. Supply either 'profiles' or 'newdata', but not both. For a model containing one exposure, a single profile may be supplied as a numeric vector. For one or more exposures, use a named list whose names match the fitted exposures exactly. Each exposure element may be: • one numeric vector, representing one chronological profile; • a list of numeric vectors, representing multiple profiles; • a numeric matrix or data frame, with one profile per row. Profiles must be ordered from the oldest observation to the most recent observation. The most recent value corresponds to lag 0. Each profile must contain exactly 'max_lag+1' observations. Longer profiles are not truncated and shorter profiles are not padded. When several fitted exposures are supplied, profiles are matched by position rather than combined as a Cartesian product. An exposure with one profile is recycled across exposures that contain multiple profiles.

`newdata` Optional long-format data frame containing new observed exposure profiles. Supply either ‘newdata‘ or ‘profiles‘, but not both. The response variable is not required and, if present, is not used for prediction. All fitted exposure variables must be present. spaMM spatial coordinates and grouping factors are not required for fixed-component prediction; the same applies to ‘profiles‘.

`group` Character scalar naming the column that identifies independent exposure profiles in ‘newdata‘, or ‘NULL‘ to treat all rows as one profile. This argument is used only to split the longitudinal prediction data. It does not request group-specific random-effect or conditional spaMM spatial predictions. Coordinates and spatial-group variables are not required for population-level prediction.

`time` Character scalar naming the chronological numeric time column in ‘newdata‘. Within each prediction group, times must be unique, complete, and equally spaced because one DLNM lag must represent the same time interval throughout the prediction data.

`type` Prediction scale: ‘"response"‘ for the expected outcome on its natural response scale, or ‘"link"‘ for the corresponding population-level linear predictor.

`uncertainty` Logical. If ‘FALSE‘, return the deterministic prediction obtained from the central fixed/population parameter estimate. If ‘TRUE‘, propagate fixed/population parameter uncertainty draw by draw.

`output` Character. When ‘uncertainty = TRUE‘, ‘"summary"‘ returns the median, standard deviation, and empirical interval across expected-response draws, whereas ‘"samples"‘ returns every draw. When ‘uncertainty = FALSE‘, only ‘"summary"‘ is valid and the returned data contain the deterministic ‘prediction‘.

`n_samples` Positive integer number of parameter draws used when ‘uncertainty = TRUE‘. At least two are required.

`probs` Numeric vector of length two defining the lower and upper empirical uncertainty quantiles. The default ‘c(0.025, 0.975)‘ gives a 95 percent interval. Used only when ‘uncertainty = TRUE‘ and ‘output = "summary"‘.

`seed` ‘NULL‘ or one strictly positive finite integer used to make parameter-draw sampling reproducible. For INLA-backed predictions, the value controls both the R random-number generator and the native ‘INLA::inla.posterior.sample()‘ seed. The caller’s global random-number state is restored when the function exits.

`extrapolation` Behavior when prediction exposures extend beyond the exposure range stored with the fitted cross-basis: ‘"warn"‘ (default), ‘"error"‘, or ‘"allow"‘. This controls notification only; the fitted basis is never re-estimated from prediction data.

Details

Prediction estimand

EpiExposure v1 defines prediction as the population-level expected response. Models may be fitted with conventional random intercepts and, for spaMM, spatially autocorrelated Matérn random effects. All non-fixed effects are excluded from predictions. This includes ‘(1 | epi_id)‘, ‘(1 | block_id)‘, ‘Matern(1 | x_coord + y_coord)‘, and independent field realizations such as ‘Matern(1 | x_coord + y_coord zero to the default prediction. In mixed-model notation, the prediction target is based on

$$\eta = X\beta$$

rather than

$$\eta = X\beta + b_i.$$

Here, "population-level" means that conventional and spatial random effects are set to zero or excluded from the prediction component. It does not mean integration over their distributions. For nonlinear links, the resulting expected response need not equal a marginally integrated mean.

With ‘type = "response"‘, the deterministic prediction is

$$\hat{\mu} = g^{-1}(X\hat{\beta}),$$

where $\hat{\beta}$ is the harmonized central fixed/population parameter estimate. Frequentist engines use their fitted fixed-effect estimates; Bayesian engines use posterior means of the fixed/population coefficients. In particular, deterministic Bayesian prediction is not defined as the median of posterior expected-response draws.

Uncertainty

With ‘uncertainty = TRUE‘, parameter uncertainty is propagated using

$$\mu^{(s)} = g^{-1}(X\beta^{(s)})$$

for each draw s. The inverse link is applied separately to every draw before any summary is computed. The same coefficient draw is applied jointly to all prediction rows, preserving covariance among predictions sharing the same fitted parameters.

For frequentist engines, fixed-effect draws are obtained from the estimated joint coefficient covariance matrix. For Bayesian engines, posterior or approximate-posterior fixed-effect draws are used. Group-specific random-effect or spatial-field uncertainty is not propagated because predictions depend exclusively on fixed/population parameters.

The uncertainty distribution therefore describes uncertainty in the expected response. It does not simulate a new observed outcome and does not add residual, observation, process, dispersion, or posterior-predictive noise.

New observed exposure data

‘newdata‘ is transformed using the cross-basis definition stored in ‘fit‘. Knots, boundary knots, spline definitions, and lag-basis parameters are not re-estimated from the new observations. Each prediction group is sorted by ‘time‘, transformed with the fitted basis, and represented by the final cross-basis row.

This is the same train-to-test principle used for out-of-fold prediction in ‘find_bestfit()‘: the fitted/training basis defines the transformation and the new or held-out exposure profile is only projected through that stored basis. spaMM models with Matérn terms do not require coordinates, spatial groups, or levels of conventional random effects in prediction ‘newdata‘ or ‘profiles‘: these variables do not enter the fixed-component prediction. For spatial spaMM models the internal point-prediction helper computes ‘X coefficients, with no simulated spatial fields or new observations.

Multiple explicit profiles

Multiple profiles are paired by profile index. For example, if ‘tmean‘ and ‘rain‘ each contain 100 profiles, profile 1 of ‘tmean‘ is combined with profile 1 of ‘rain‘, and so forth. The function does not create all pairwise combinations. If one exposure contains a single profile and another contains 100, the single profile is recycled 100 times.

Value

A data frame.

For multiple explicit profiles, a ‘profile‘ column identifies matched profile combinations. For ‘newdata‘ with ‘group‘ supplied, the grouping column identifies each predicted profile.

With ‘uncertainty = FALSE‘, the result contains ‘prediction‘.

With ‘uncertainty = TRUE‘ and ‘output = "summary"‘, the result contains ‘prediction‘, ‘prediction_sd‘, ‘prediction_lower‘, and ‘prediction_upper‘.

With ‘uncertainty = TRUE‘ and ‘output = "samples"‘, the result contains ‘sample‘ and ‘prediction‘, with one row for each parameter draw and profile.

---

prepare_response *Prepare and validate the epidemic-level response for modeling*

Description

Validates an epidemic-level response against the distribution family selected by the user and creates the standardized ‘y_model‘ column required by ‘fit_epidlnm()‘.

Usage

```
prepare_response(
  data,
  response = NULL,
  family,
  beta_scale = c("auto", "proportion", "percent"),
  y_var = NULL
)
```

Arguments

data	Non-empty data frame containing the response variable. In the standard workflow this is the output of ‘build_design()‘.
response	Character scalar naming the original response variable in ‘data‘. In the standard workflow this is ‘"y"‘. For compatibility with EpiExposure model-selection code, ‘response‘ may be omitted when ‘y_var‘ is supplied. If both are supplied they must identify the same column.

`family` Distribution family supplied as a supported character name or as a family-like object containing a valid ‘family‘ field.

Supported EpiExposure v1 canonical families are:

- ‘"beta"‘: continuous proportions strictly inside ‘(0, 1)‘; - ‘"binomial"‘: binary outcomes coded exactly ‘0/1‘; - ‘"poisson"‘: non-negative integer counts; - ‘"negative_binomial"‘: NB2 non-negative integer counts; - ‘"gaussian"‘: finite continuous numeric values; - ‘"gamma"‘: strictly positive continuous values.

Aliases such as ‘"bernoulli"‘, ‘"normal"‘, ‘"negbin"‘, ‘"nbinom"‘, and ‘"nbinom2"‘ are normalized to the corresponding canonical family.

‘nbinom1‘ / NB1 are rejected explicitly. EpiExposure v1 standardizes ‘"negative_binomial"‘ to the NB2 variance parameterization. Ordinal outcomes are not supported in EpiExposure v1.

`beta_scale` Character scalar controlling interpretation of a Beta response. One of:

- ‘"auto"‘ (default): values already within ‘[0, 1]‘ are interpreted as proportions; otherwise, if all values lie within ‘[0, 100]‘ and at least one value exceeds 1, they are interpreted as percentages and divided by 100; - ‘"proportion"‘: require the supplied response to be on the ‘[0, 1]‘ scale; - ‘"percent"‘: require values on the ‘[0, 100]‘ scale and divide by 100.

After scale handling, Beta regression still requires the **open** interval ‘(0, 1)‘. Exact boundary values 0 or 1 are not altered silently.

`y_var` Optional compatibility alias for ‘response‘. This is used by existing EpiExposure internal workflows. New user-facing code should prefer ‘response‘.

Details

In the standard EpiExposure workflow:

“‘ templates <- define_exposures(...) design <- build_design(data = data, cb_templates = templates, include_response = TRUE) prepared <- prepare_response(data = design, response = "y", family = "beta") fit <- fit_epidlnm(data = prepared, ...) “‘

‘build_design(include_response = TRUE)‘ and ‘prepare_response()‘ have different responsibilities. ‘build_design()‘ only carries the original epidemic-level outcome into the design matrix. ‘prepare_response()‘ checks whether that outcome is compatible with the requested statistical family and performs only the explicitly documented scale conversion needed for Beta percentages.

‘prepare_response()‘ does **not** choose a probability distribution from the observed data and does not perform goodness-of-fit model selection. The ‘family‘ is selected by the analyst and is then validated here.

Why this function remains necessary after ‘build_design()‘

With ‘include_response = TRUE‘, ‘build_design()‘ ensures that one original epidemic-level ‘y‘ value is carried into each design row. It deliberately does not interpret that outcome statistically.

‘prepare_response()‘ is the family-aware layer between the design matrix and ‘fit_epidlnm()‘. ‘fit_epidlnm()‘ expects a standardized numeric ‘y_model‘ column and checks that the response-family metadata agree with the family used for fitting.

Beta responses

Beta regression requires

$$0 < y < 1.$$

When ‘beta_scale = "auto"‘, the function retains the previous EpiExposure convenience of converting a clear 0–100 percentage scale to proportions. The conversion is recorded in output metadata.

Earlier EpiExposure code silently replaced exact 0 and 1 by ‘1e-5‘ and ‘1 - 1e-5‘. That behavior is intentionally removed. Boundary modification changes observed outcomes and the appropriate treatment depends on the scientific data-generating process. Exact 0/1 values therefore produce an explicit error rather than an undocumented numerical adjustment.

Count responses

Poisson and negative-binomial responses must be non-negative integers. Values are validated but never truncated or rounded to make them valid.

EpiExposure v1 uses NB2 whenever the canonical family is ‘"negative_binomial"‘. Explicit NB1 requests generate an error before model fitting so different engines cannot silently use different mean–variance relationships.

Binomial, Gaussian, and Gamma responses

Binomial responses must be numeric 0/1. Gaussian responses may be any finite numeric values. Gamma responses must be strictly positive.

No distribution inference

This function validates a **chosen** family; it does not infer the best family from histograms, moments, normality tests, overdispersion tests, or other automatic rules. Distribution choice remains a modeling decision based on the outcome definition and study design.

Value

A data frame containing all original columns plus ‘y_model‘.

Existing attributes on ‘data‘ (including the design/template metadata created by ‘build_design()‘) are preserved. The following response metadata are added:

- ‘"response_family_name"‘: canonical EpiExposure family; - ‘"response_name"‘: original response-column name; - ‘"response_original_scale"‘: response scale before preparation; - ‘"response_model_scale"‘: scale represented by ‘y_model‘; - ‘"response_transform"‘: transformation applied, if any; - ‘"y_scale_mult"‘: multiplier that maps the Beta model scale back to the original numerical scale (‘100‘ after percent-to-proportion conversion, otherwise ‘1‘); - ‘"response_nb_parameterization"‘: ‘"NB2"‘ for negative-binomial responses, otherwise ‘NULL‘; - ‘"response_contract"‘: ‘"family_validated_response_v1"‘.

`reduce_effects` *Reduce a fitted DLNM to a one-dimensional association*

Description

Reduces one fitted EpiExposure distributed lag non-linear model (DLNM) exposure-lag association using ‘dlnm::crossreduce()‘.

Usage

```
reduce_effects(
  fit,
  data,
  vars,
  group = "epi_id",
  time = "time",
  type = c("overall", "lag", "var"),
  value = NULL,
  scale = c("percent", "response", "link"),
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  seed = NULL,
  ref = list(method = "median", value = NULL),
  at = NULL,
  interval_probs = c(0.025, 0.975),
  extrapolation = c("error", "warn", "allow")
)
```

Arguments

fit	Fitted model returned by the current ‘fit_epidlnm()‘.
data	Non-empty long-format exposure data containing ‘group‘, ‘time‘, and **every exposure variable fitted in ‘fit‘**. ‘data‘ is used only to: - validate the exact EpiExposure temporal-profile contract; - calculate a method-based reference value when requested; - validate exposure support for ‘value‘, ‘at‘, and the reference. It is **not** used to reconstruct or re-estimate the fitted cross-basis. Every group must contain exactly $$profile_length = max_lag + 1$$ rows, with finite complete exposure profiles, unique chronological times, regular spacing within groups, and the same spacing across groups.
vars	Character scalar naming one exposure variable that was fitted in ‘fit‘.
group	Character scalar naming the grouping column in ‘data‘. Default is ‘"epi_id"‘.
time	Character scalar naming the chronological time column in ‘data‘. Default is ‘"time"‘. The column must be finite numeric.
type	Reduction type: - ‘"overall"‘: overall cumulative exposure-response association across the complete fitted lag window; - ‘"lag"‘: exposure-response association at one specified lag; - ‘"var"‘: lag-response association at one specified exposure value. These meanings follow ‘dlnm::crossreduce()‘.

`value` ‘NULL‘ for ‘type = "overall"‘. For ‘type = "lag"‘, one finite numeric lag coordinate within ‘0:max_lag‘. For ‘type = "var"‘, one finite numeric exposure value at which the lag-response association is reduced.

`scale` Character. One of ‘"percent"‘, ‘"response"‘, or ‘"link"‘.

This argument is retained for backward compatibility with the original EpiExposure API:

- ‘"link"‘ returns the centered DLNM association contrast ‘eta‘ unchanged; - ‘"response"‘ is a legacy label for the directly transformed **association measure**, not an absolute expected response. For a log link it returns ‘exp(eta)‘, a response ratio; for a logit link it returns ‘exp(eta)‘, an odds ratio; for an identity link it returns the additive response-scale difference ‘eta‘; - ‘"percent"‘ returns ‘100 * (exp(eta) - 1)‘ for log and logit links.

For a log link, ‘"percent"‘ is the percent relative change in the expected response. For a logit link, it is the percent change in **odds**, not a percentage-point change in probability.

A centered DLNM contrast alone does not identify an absolute response probability/mean because the model intercept and the complete joint reference profile are not part of the reduced contrast. Therefore ‘"response"‘ is unavailable for probit, complementary-log-log, inverse, or other links for which no direct association-scale transformation is defined here. Use ‘summarise_effects()‘ when absolute response-scale ‘baseline‘, ‘predicted‘, and ‘delta‘ quantities are required.

`uncertainty` Logical scalar. If ‘FALSE‘, use the central fixed/population parameter estimate. If ‘TRUE‘, propagate joint fixed/population parameter uncertainty draw by draw.

`output` Character. ‘"summary"‘ or ‘"samples"‘.

‘"samples"‘ requires ‘uncertainty = TRUE‘ and returns one row per parameter draw and reduction coordinate. ‘"summary"‘ returns deterministic central-parameter values when ‘uncertainty = FALSE‘, or median, standard deviation, and empirical quantiles when ‘uncertainty = TRUE‘.

`n_samples` Positive integer number of parameter draws when ‘uncertainty = TRUE‘. Default is ‘1000‘; at least two draws are required.

`seed` ‘NULL‘ or one non-negative finite integer controlling parameter sampling when ‘uncertainty = TRUE‘. The caller’s previous random-number state is restored when the function exits.

`ref` Reference exposure specification used as ‘cen‘ in ‘dlnm::crossreduce()‘. Default is ‘list(method = "median", value = NULL)‘.

Supported method-based forms are:

- ‘list(method = "median", value = NULL)‘; - ‘list(method = "percentile", value = p)‘, with ‘0 <= p <= 1‘; - ‘list(method = "fixed", value = x)‘, with finite numeric ‘x‘.

Unlike ‘summarise_effects()‘, this function reduces only one focal exposure at a time, so a joint multi-exposure reference profile is not required. The selected scalar reference is passed explicitly to ‘crossreduce(cen = ...)‘; the function never relies on the implicit mid-range centering defaults of ‘dlnm‘.

at	Optional finite numeric vector of exposure values at which the reduced exposure-response association is evaluated for ‘type = "overall"‘ or ‘type = "lag"‘. If ‘NULL‘, the prediction grid is selected by ‘dlnm::crossreduce()‘ from the stored fitted basis range. ‘at‘ is not applicable to ‘type = "var"‘ because that reduction is evaluated over lag coordinates.
interval_probs	Numeric vector of length two defining the empirical uncertainty interval. Default is ‘c(0.025, 0.975)‘.
extrapolation	Character controlling exposure values outside the range stored with the fitted cross-basis: ‘"error"‘ (default), ‘"warn"‘, or ‘"allow"‘. The setting applies to the reference value, ‘value‘ when ‘type = "var"‘, and user-supplied ‘at‘. It never re-estimates the basis.

Details

The function is a **reduction/reparameterization** of the fitted DLNM, not a new model fit. The fitted two-dimensional exposure-by-lag association is re-expressed using the one-dimensional basis for either the exposure dimension or the lag dimension.

Critically, ‘reduce_effects()‘ uses the exact ‘crossbasis‘ object and coefficient mapping stored by the current ‘fit_epidlnm()‘ implementation. It never re-estimates knots, boundary knots, basis dimensions, lag parameterization, or coefficient alignment from the supplied ‘data‘.

What ‘crossreduce()‘ does

A fitted DLNM is parameterized by a two-dimensional cross-basis. The ‘dlnm::crossreduce()‘ operation re-expresses that fit using modified coefficients for a one-dimensional basis. It can produce:

- an overall cumulative exposure-response summary (‘type = "overall"‘); - an exposure-response summary at one lag (‘type = "lag"‘); - a lag-response summary at one exposure value (‘type = "var"‘).

The reduction is algebraic: it does not refit the epidemiological model.

Stored fitted basis is authoritative

The ‘crossbasis‘ used here is the object stored in ‘attr(fit, "epiexposure_basis_objects")‘. Canonical EpiExposure cross-basis coefficient names are mapped to the stored native basis order through the metadata created by ‘define_exposures()‘, ‘build_design()‘, and ‘fit_epidlnm()‘.

‘data‘ is never pooled to create another ‘crossbasis‘. This prevents prediction/reduction data from redefining data-dependent spline knots, boundary knots, ranges, or column order.

Exact common-profile contract

Current EpiExposure fits satisfy:

$$profile_length = max_lag + 1$$

and every fitted exposure uses the same ‘max_lag‘.

‘reduce_effects()‘ validates this metadata and requires every supplied group to contain exactly that many rows. Longer profiles are not truncated and shorter profiles are not padded. Every fitted exposure must be present in ‘data‘ on the same rows, so the function cannot silently compare or derive references from profiles with different temporal support.

For ‘max_lag = 0‘, ‘history_length = 1‘; no temporal step can be inferred from a one-row profile and the stored time step is expected to be ‘NA‘.

Centering and reference

‘dlnm::crossreduce()‘ has its own default centering rules when ‘cen‘ is not supplied. EpiExposure deliberately does not use those implicit defaults. The ‘ref‘ argument resolves one explicit focal-exposure reference value and passes it as ‘cen‘.

Consequently, ‘eta = 0‘ is the neutral centered association and, for log or logit links, ‘exp(eta) = 1‘ is the neutral exponentiated association.

Link scale versus transformed association

‘eta‘ is a contrast on the model’s additive linear-predictor scale. Applying an inverse link directly to a contrast is generally **not** the same as calculating an expected response.

In particular, for a logit model:

$$exp(eta)$$

is an odds ratio, whereas

$$plogis(eta)$$

is not the response probability associated with the fitted model unless the omitted baseline linear predictor were exactly zero. Therefore this function never applies ‘plogis()‘ directly to the reduced contrast.

Uncertainty contract

When ‘uncertainty = FALSE‘, the reduction uses the central parameter estimate:

- fitted fixed coefficients for frequentist engines; - posterior-mean fixed/population coefficients for Bayesian engines.

When ‘uncertainty = TRUE‘, EpiExposure obtains one joint fixed/population-parameter draw matrix from the shared engine helpers. The same row of that matrix defines one coherent draw. The focal cross-basis coefficients are then reduced by ‘crossreduce()‘ draw by draw.

All transformations are performed **before** uncertainty summaries are calculated. Summary output therefore reports the median, SD, and empirical interval of the transformed draw distribution itself.

Fitted random effects are excluded. Residual, observation, process, dispersion, and posterior-predictive noise are not added. Uncertainty therefore refers to the fitted association/expected-response parameter structure, not to a future noisy observation.

Value

A data frame.

The reduction coordinate is stored in ‘x‘:

- for ‘type = "overall"‘ and ‘type = "lag"‘, ‘x‘ is exposure; - for ‘type = "var"‘, ‘x‘ is lag.

Every output contains:

- ‘x‘: reduction coordinate; - ‘eta‘: centered reduced association on the fitted link/linear-predictor scale; - ‘effect‘: requested representation of that association; - ‘type‘, ‘value‘, ‘reference‘, ‘scale‘, and ‘var‘.

With ‘uncertainty = TRUE, output = "summary"‘, the result additionally contains:

- ‘eta_sd‘, ‘low‘, ‘high‘: SD and empirical interval for ‘eta‘; - ‘effect_sd‘, ‘low_eff‘, ‘high_eff‘: SD and empirical interval for the draw-by-draw transformed effect.

With ‘output = "samples"‘, ‘sample‘ identifies the joint parameter draw.

Attributes store the central reduced one-dimensional coefficients and basis returned by ‘dlnm::crossreduce()‘, the reference and transformation contracts, and the inherited EpiExposure temporal metadata.

`simulate_exposures` *Simulate chronological exposure profiles*

Description

Generates one or more complete exposure profiles in chronological order, from the earliest/oldest observation to the most recent observation.

Usage

```
simulate_exposures(
  max_lag,
  n = 1,
  mode = c("profile", "pattern"),
  pattern = c("random_times", "alternating_times", "block_times"),
  fixed_value = NULL,
  n_times = NULL,
  time_range = NULL,
  alternating_values = NULL,
  block_times = NULL,
  background = list(dist = "normal", mean = 0, sd = 1),
  bounds = NULL,
  seed = NULL,
  cumulative = FALSE
)
```

Arguments

`max_lag`	Non-negative integer maximum retrospective lag. Every generated profile has exactly ‘max_lag + 1‘ chronological positions.
`n`	Positive integer number of exposure profiles to simulate. Default is 1.
`mode`	Character. ‘"profile"‘ generates the background process only; ‘"pattern"‘ generates the background and then applies one explicit chronological pattern.

`pattern` Character pattern used when ‘mode = "pattern"‘:

- ‘"random_times"‘: assign ‘fixed_value‘ at ‘n_times‘ distinct chronological positions sampled without replacement from ‘time_range‘; - ‘"alternating_times"‘: recycle ‘alternating_values‘ through ‘time_range‘ in chronological order; - ‘"block_times"‘: assign ‘fixed_value‘ to the complete inclusive interval defined by ‘block_times‘.

`fixed_value` Finite numeric scalar used by ‘"random_times"‘ and ‘"block_times"‘.

`n_times` Positive integer number of selected chronological positions for ‘"random_times"‘.

`time_range` Optional unique integer vector of chronological times used by ‘"random_times"‘ or ‘"alternating_times"‘. Values must be between 0 and ‘max_lag‘, inclusive. If ‘NULL‘, every chronological position is eligible.

‘time_range‘ is not used by ‘"block_times"‘; supplying it with that pattern is an error rather than being silently ignored.

`alternating_values` Numeric vector containing at least two finite values. Values are recycled across the selected ‘time_range‘ in chronological order when ‘pattern = "alternating_times"‘.

`block_times` Unique integer vector of length two defining the first and final chronological positions of the inclusive block when ‘pattern = "block_times"‘.

`background` Named list describing the background process. Supported distributions are:

‘"normal"‘ ‘list(dist = "normal", mean = ..., sd = ...)‘

‘"empirical"‘ ‘list(dist = "empirical", values = ...)‘

‘"ar1"‘ ‘list(dist = "ar1", mean = ..., sd = ..., phi = ...)‘

‘"fixed"‘ ‘list(dist = "fixed", value = ...)‘

For the stationary Gaussian AR(1) generator, ‘sd‘ is the stationary marginal standard deviation and ‘phi‘ must lie strictly inside ‘(-1, 1)‘.

Unknown background fields are rejected explicitly to catch misspelled parameter names.

`bounds` Optional finite numeric vector ‘c(lower, upper)‘. Background values are clipped to these bounds before the pattern and values are clipped again after the pattern.

If ‘cumulative = TRUE‘, ‘bounds‘ constrain the exposure **increments** before ‘cumsum()‘; the final cumulative trajectory is not clipped again.

`seed` Optional finite integer seed controlling the complete set of ‘n‘ exposure simulations. The caller’s random-number state is restored after the function returns.

`cumulative` Logical. If ‘TRUE‘, applies ‘cumsum()‘ in chronological order after background generation, pattern assignment, and bounds.

Details

The first profile value corresponds internally to the maximum retrospective lag and the final profile value corresponds internally to lag 0:

“‘ time_0 = earliest/oldest observation = lag max_lag time_max_lag = most recent observation = lag 0 “‘

‘simulate_exposures()‘ simulates exposure profiles only. It does not draw model coefficients and therefore has no ‘"summary"‘/‘"samples"‘ output switch. The number of simulated exposure profiles is controlled exclusively by ‘n‘.

One simulation contract

‘n‘ is the only argument controlling how many exposure trajectories are generated:

“‘ x1 <- simulate_exposures(..., n = 1) length(x1$profiles) # 1

x100 <- simulate_exposures(..., n = 100) length(x100$profiles) # 100 “‘

There is no statistical summarization inside this function. If ‘n = 100‘, all 100 simulated profiles are retained.

This is deliberately different from ‘predict_outcomes()‘ and ‘compare_predictions()‘, where ‘"samples"‘ refers to uncertainty draws of fitted model parameters and ‘"summary"‘ summarizes those parameter-draw predictions.

Direct use in ‘compare_predictions()‘

The whole returned object can be used directly as one exposure component:

“‘ rain_A <- simulate_exposures(..., n = 100)

scenario_A <- list(tmean = 25, rain = rain_A, wetness = 10) “‘

‘compare_predictions()‘ recognizes ‘$profiles‘ as the canonical collection. If other exposure variables in the same scenario also contain 100 profiles, they are paired by simulation index:

“‘ tmean profile 1 + rain profile 1 + wetness profile 1 tmean profile 2 + rain profile 2 + wetness profile 2 ... “‘

A variable with exactly one profile may be recycled across the common scenario profile count. No Cartesian product is constructed.

Chronological versus retrospective indexing

All user-facing pattern arguments use chronological time:

“‘ time = 0 -> earliest/oldest observation time = max_lag -> most recent observation “‘

The corresponding retrospective lag is:

$$lag = max_lag - time.$$

No profile reversal is performed after simulation.

Background generators

For ‘background$dist = "normal"‘, positions are independent Gaussian draws.

For ‘"empirical"‘, positions are sampled with replacement from the supplied finite empirical values.

For ‘"ar1"‘, the process is initialized from its stationary marginal distribution and uses innovation SD

$$sd_{innovation} = sd\sqrt{1 - \phi^2}.$$

Thus ‘background$sd‘ is the stationary marginal SD rather than the innovation SD.

For ‘"fixed"‘, every background position equals ‘background$value‘.

Seeds across exposure variables

‘seed‘ controls one call to ‘simulate_exposures()‘. Reusing exactly the same seed in separate stochastic calls for different exposure variables can reuse the same underlying pseudo-random sequence and unintentionally induce dependence among simulated exposures. Use different seeds for independently simulated variables unless synchronized random structure is intentional.

Pattern-specific arguments

Pattern arguments that do not belong to the selected mode/pattern are rejected rather than silently ignored.

Value

An object of class ‘"epiexposure_simulated_exposures"‘ with:

‘profiles‘ Canonical list containing exactly ‘n‘ simulated numeric chronological profiles, named ‘"simulation_1"‘, ..., ‘"simulation_n"‘.

‘profile‘ Backward-compatible numeric alias when ‘n = 1‘; otherwise ‘NULL‘.

‘simulation_data‘ Long data frame containing ‘simulation‘, ‘position‘, ‘time‘, ‘lag‘, and ‘profile‘. This is a tabular view of the same simulated exposure profiles, not model-parameter samples.

‘meta‘ Common simulation metadata, including ‘max_lag‘, chronological order, ‘n‘, background specification, and RNG contract.

‘simulation_meta‘ List with simulation-specific pattern metadata, such as selected chronological positions and corresponding retrospective lags.

---

simulate_losses *Simulate yield and economic losses*

Description

Converts a non-negative response variable into yield and economic losses using an externally supplied linear yield-response relationship.

Usage

```
simulate_losses(
  data,
  y,
  lower = NULL,
  upper = NULL,
  slope = 100,
  intercept = 11142.94,
  parameter_mode = c("uniform", "values", "draws"),
  attainable_yield = seq(4000, 12000, by = 50),
  price = seq(100, 300, by = 10),
  n = 1,
  random_sd = 0,
  y_multiplier = 1,
  constraint_action = c("warn", "error"),
  seed = NULL
)
```

Arguments

data	Non-empty data frame containing the response variable and any additional scenario/prediction identifiers that should be retained.
y	Character scalar naming the numeric response column used in the yield-loss relationship.
lower	Optional character scalar naming a lower response bound. If supplied, ‘upper‘ must also be supplied.
upper	Optional character scalar naming an upper response bound. If supplied, ‘lower‘ must also be supplied. Bounds are propagated deterministically through the loss equations for each loss-model simulation. They are therefore endpoint-propagated bounds, not automatically a joint confidence/credible interval after combining response uncertainty with yield-model parameter uncertainty.
slope	Finite non-negative numeric slope specification. Interpretation depends on ‘parameter_mode‘: - ‘"values"‘: every supplied value is evaluated exactly; - ‘"uniform"‘: length 1 is fixed and length 2 defines Uniform bounds; - ‘"draws"‘: values are user-supplied draws paired by simulation index. Increasing response is assumed not to improve yield, so slope values must be non-negative.
intercept	Finite positive reference-yield intercept specification. Interpretation follows the same ‘parameter_mode‘ contract as ‘slope‘. In ‘"values"‘ mode, all exact slope-intercept combinations are evaluated. In ‘"uniform"‘ mode, two-value ranges are sampled independently. In ‘"draws"‘ mode, non-scalar slope and intercept vectors must have the same length and are paired by position; scalar parameters are recycled.
parameter_mode	Character controlling interpretation of ‘slope‘ and ‘intercept‘. One of ‘"uniform"‘ (default, preserving the original interface), ‘"values"‘, or ‘"draws"‘. ‘"values"‘ evaluates exact supplied parameter values. For example, ‘slope = c(49.3, 80)‘ evaluates both slopes exactly rather than sampling between them. If both ‘slope‘ and ‘intercept‘ contain multiple exact values, all combinations are evaluated and ‘n‘ is ignored. ‘"uniform"‘ uses ‘n‘ simulations. A scalar parameter is fixed, whereas a two-value vector defines the lower and upper limits of an independent Uniform distribution. ‘"draws"‘ treats supplied vectors as paired external parameter draws. Scalars are recycled. If ‘n‘ is omitted, it is inferred from the common non-scalar draw length; if ‘n‘ is supplied explicitly, it must match that length.
attainable_yield	Unique non-negative numeric vector of attainable yields in kg/ha, used to scale the disease-attributable proportional loss into absolute yield loss in kg/ha. ‘attainable_yield‘ is conceptually distinct from the intercept of the external calibration equation. The intercept defines the reference yield used to estimate proportional loss, whereas ‘attainable_yield‘ defines the scenario-specific yield to which that proportional loss is applied.

`price` Unique non-negative numeric vector of commodity prices in USD per metric ton (USD/t).

`n` Positive integer number of loss-model parameter simulations. Default is 1. It controls the number of simulations in ‘parameter_mode = "uniform"‘. In ‘parameter_mode = "draws"‘, it may be supplied explicitly to validate the paired draw length; when omitted, the length is inferred from non-scalar supplied draws. In ‘parameter_mode = "values"‘, the exact parameter grid determines the number of simulations and ‘n‘ is ignored.

‘n‘ does not represent EpiExposure model-coefficient draws unless the user explicitly supplies such external parameter draws through ‘slope‘ and ‘intercept‘.

`random_sd` Non-negative numeric scalar controlling an optional simulation-level Gaussian deviation in the external yield-model intercept. Default is 0.

For simulation ‘s‘,

$$Y_{0,s} = intercept_s + b_s, \qquad b_s \sim N(0, random_sd^2).$$

The same ‘b_s‘ is shared by every row, attainable-yield value, and price within loss simulation ‘s‘. This preserves matched scenario comparisons under the same external yield-model realization.

This quantity is **not** a random effect from the EpiExposure epidemiological fit. If residual/future-yield noise is not scientifically intended, leave ‘random_sd = 0‘.

`y_multiplier` Positive numeric scalar applied to ‘y‘, ‘lower‘, and ‘upper‘ before the yield-loss equations. Default is 1.

For example, use ‘y_multiplier = 100‘ when the response is stored as a proportion from 0 to 1 but the external yield-loss coefficients were calibrated against response percentages from 0 to 100.

`constraint_action`

Character. Action when the raw proportional disease-attributable loss falls outside the physically interpretable range ‘[0, 1]‘:

- ‘"warn"‘ (default): constrain to ‘[0, 1]‘ and issue one warning; - ‘"error"‘: stop before returning results.

Negative raw loss is normally prevented by the non-negative response and slope validation; values above 1 can occur when the supplied linear relationship extrapolates beyond complete yield loss.

`seed` Optional finite integer random seed. It controls only stochastic generation requested by this function (Uniform parameter ranges and/or the Gaussian intercept deviation). The caller’s RNG state is restored.

Details

Each row of ‘data‘ is retained and expanded across all combinations of ‘attainable_yield‘, ‘price‘, and resolved loss-model parameter simulations.

‘simulate_losses()‘ does not fit an epidemiological model and does not draw coefficients from an EpiExposure fit. The response supplied in ‘y‘ is treated as already available. Parameter simulations refer only to exact external values, user-requested Uniform draws, supplied parameter draws, and/or the optional external intercept deviation.

Loss equations

The internally scaled response is

$$R = y \times y_multiplier.$$

For loss simulation s, the zero-response reference yield is

$$Y_{0,s} = \alpha_s + b_s,$$

where α_s is 'intercept_sim' and b_s is the optional simulation-level 'rand_eff'.

The raw disease-attributable proportional loss is

$$L_{raw} = \frac{\beta_s R}{Y_{0,s}},$$

where β_s is 'slope_sim'.

The physically interpretable loss proportion is

$$L = min\{1, max(0, L_{raw})\}.$$

Consequently,

$$relative_yield = 1 - L$$

and the calibration-scale predicted yield is

$$predicted_yield = Y_{0,s}(1 - L).$$

This construction deliberately keeps 'pred_yield', 'rl_prop', and 'yl_prop' internally consistent. Earlier code subtracted the simulated intercept deviation from 'pred_yield' but omitted it from the proportional-loss denominator, allowing those quantities to disagree.

Attainable-yield scaling

The proportional loss estimated from the external calibration relationship is applied to each scenario-specific attainable yield:

$$absolute_loss = attainable_yield \times L,$$

$$attainable_predicted_yield = attainable_yield \times (1 - L).$$

These two quantities sum exactly to 'attainable_yield' up to floating-point tolerance.

Economic conversion

The unit contract of 'simulate_losses()' is fixed:

- 'att_yield' and 'yield_loss' are in kg/ha; - 'price' is in USD per metric ton (USD/t); - 'econ_loss' is returned in USD/ha.

Because one metric ton equals 1000 kg,

$$economic_loss = \left(\frac{yield_loss}{1000}\right) \times price.$$

The division by 1000 is therefore part of the function's unit contract, not a tunable modeling parameter.

Parameter simulations

‘parameter_mode‘ removes the original ambiguity of two-value parameter vectors.

With ‘"values"‘, supplied parameter values are exact. All slope-intercept combinations are evaluated, so ‘slope = c(49.3, 80)‘ represents two exact slope scenarios when ‘intercept‘ is scalar.

With ‘"uniform"‘, scalar parameters remain fixed and two-value vectors define Uniform bounds. ‘n‘ controls the number of independently sampled parameter realizations. If both slope and intercept are ranges, they are sampled independently and therefore do not preserve covariance from an external fitted yield model.

With ‘"draws"‘, supplied non-scalar vectors are treated as externally generated paired draws and are matched by simulation index. This mode should be used when joint slope-intercept uncertainty has already been estimated.

‘random_sd‘, when positive, remains an optional simulation-level Gaussian deviation in the external yield-model intercept under every parameter mode.

Propagating response bounds

When ‘lower‘ and ‘upper‘ are supplied, every loss simulation evaluates both endpoints using the **same** slope, intercept, random-effect realization, attainable yield, and price. Metric-specific minima and maxima are then reported as ‘_lower‘ and ‘_upper‘.

These are propagated endpoint bounds. If ‘lower‘ and ‘upper‘ are marginal prediction intervals and multiple external parameter simulations simultaneously represent uncertainty in a yield model, the resulting endpoint columns are not automatically a calibrated joint probability interval. For fully joint Monte Carlo propagation, supply draw-level response values as rows of ‘data‘ and preserve their draw identifiers.

Relationship to EpiExposure predictions

‘simulate_losses()‘ is a downstream transformation. It does not alter the EpiExposure prediction contract. If ‘y‘ comes from ‘predict_outcomes()‘ or ‘compare_predictions()‘, the user should normally provide a response-scale expected outcome that is on the same scale used to calibrate ‘slope‘ and ‘intercept‘.

‘random_sd‘ pertains only to the external yield-loss relationship. It does not reactivate group-specific random effects from the epidemiological model.

Value

A data frame containing all original columns plus:

‘.loss_row_id‘ Original row index in ‘data‘.

‘.sim‘ Loss-model simulation index.

‘slope_sim‘, ‘intercept_sim‘ External yield-model parameters used in that simulation.

‘rand_eff‘ Simulation-level external yield-intercept deviation.

‘ref_yield‘ Zero-response yield for that loss simulation: ‘intercept_sim + rand_eff‘.

‘y_used‘ Scaled response ‘y * y_multiplier‘.

‘yl_prop_raw‘ Unconstrained proportional loss ‘slope_sim * y_used / ref_yield‘.

‘yl_capped‘ Whether the raw proportional loss required constraining to ‘[0, 1]‘.

‘yl_prop‘, ‘yl_pct‘ Constrained disease-attributable loss proportion and percent.

‘rl_prop‘, ‘rl_pct‘ Remaining yield relative to the simulation-specific reference yield. By construction, ‘rl_prop = 1 - yl_prop‘.

‘pred_yield‘ Yield on the external calibration-model scale: ‘ref_yield * rl_prop‘.

‘att_yield‘ User-specified attainable yield in kg/ha.

‘ap_yield‘ Attainable yield remaining after the same proportional loss.

‘yield_loss‘ Absolute attributable yield loss in kg/ha: ‘att_yield * yl_prop‘.

‘price‘ Commodity price in USD per metric ton (USD/t).

‘econ_loss‘ Economic loss in USD/ha: ‘(yield_loss / 1000) * price‘.

If ‘lower‘ and ‘upper‘ are supplied, corresponding ‘_lower‘ and ‘_upper‘ columns are added for every response-dependent metric.

Attributes document the loss-model specification, unit conversion, constraint behavior, parameter-simulation contract, and propagated-bound interpretation.

`simulate_ranges` *Generate exposure-value scenarios across DLNM lag periods*

Description

Creates structured scenario definitions for ‘simulate_scenarios()‘ by generating combinations of user-specified exposure values and copying each resulting combination unchanged to every supplied lag period.

Usage

```
simulate_ranges(
  periods,
  vary = list(),
  fixed = list(),
  scenario_type = c("grid", "paired"),
  scenario_names = NULL,
  scenario_var = NULL
)
```

Arguments

periods	A non-empty data frame, typically returned by ‘define_periods()‘, containing ‘period‘, ‘lag_start‘, and ‘lag_end‘. Periods are expressed on the retrospective lag scale. For example, ‘lag_start = 0‘ and ‘lag_end = 20‘ represent lags 0 through 20, the portion of the exposure profile closest to outcome assessment. Periods must not overlap. Gaps are allowed: any exposure-lag positions not covered later by a scenario remain available for background filling in ‘simulate_scenarios()‘.
vary	Named list of exposure variables and finite numeric values to evaluate. At least one varying variable is required. With ‘scenario_type = "grid"‘, the Cartesian product across the variables in ‘vary‘ is generated **once**. Each resulting combination is then copied to every supplied lag period. No additional Cartesian product is generated across periods. With ‘scenario_type = "paired"‘, values are matched by position across variables. All vectors in ‘vary‘ must therefore have the same length. Values within each varying variable must be unique. Repeated values would create duplicated scenario combinations and are rejected explicitly.
fixed	Named list of exposure variables held fixed in every generated scenario and every supplied period. Each fixed variable must contain exactly one finite numeric value. Use ‘list()‘ when none are required.
scenario_type	Character. ‘"grid"‘ or ‘"paired"‘.
scenario_names	Optional character vector containing exactly one unique, non-empty name for each generated scenario.
scenario_var	Optional name of one variable in ‘vary‘. It is used only to construct the automatic scenario-name prefix and is retained as metadata. It does **not** give that exposure a special statistical role. If ‘NULL‘, the first variable in ‘vary‘ is used.

Details

This function generates **scenario definitions only**. It does not fit a model, construct a crossbasis, or predict an outcome. Conversion from retrospective lag-period definitions to complete chronological exposure profiles is performed later by ‘simulate_scenarios()‘, which then calls ‘predict_outcomes()‘.

How values are repeated across periods

‘simulate_ranges()‘ deliberately does **not** generate separate exposure values for each period. Suppose four periods cover lags 0–85 and one grid combination is:

“‘ tmean = 20 rain = 8 wetness = 2 “‘

The generated scenario is equivalent to:

“‘ P1 = list(tmean = 20, rain = 8, wetness = 2) P2 = list(tmean = 20, rain = 8, wetness = 2) P3 = list(tmean = 20, rain = 8, wetness = 2) P4 = list(tmean = 20, rain = 8, wetness = 2) “‘

‘simulate_scenarios()‘ then expands each scalar across all lags belonging to its period. If P1–P4 jointly cover the entire fitted lag window and the variables in ‘vary‘ plus ‘fixed‘ include every exposure in the fitted model, the resulting exposure profile is complete and no background/reference value is needed for those positions.

If periods leave gaps, do not extend to the fitted maximum lag, or omit one or more fitted exposures, ‘simulate_scenarios()‘ fills only the unassigned positions from ‘ref_vals‘ or from exposure medians derived from ‘data‘.

Grid versus paired scenarios

With:

“‘ vary = list(tmean = c(20, 34), rain = 0:15, wetness = c(2, 10)) “‘

‘scenario_type = "grid"‘ generates

$$2 \times 16 \times 2 = 64$$

scenarios. If four periods are supplied, there are still 64 scenarios, not 64^4: each of the 64 combinations is simply copied to all four periods.

‘scenario_type = "paired"‘ instead combines the first value of each varying exposure, then the second value of each, and so forth.

Value

A structured list containing:

‘scenarios‘ Named scenario definitions. Each generated exposure combination is copied to every supplied period.

‘periods‘ The validated lag-period definitions.

‘info‘ One metadata row per generated scenario. It contains the scenario name and the values of both varying and fixed exposures.

Additional attributes record ‘scenario_type‘, ‘scenario_var‘, and simple period-coverage diagnostics.

`simulate_scenarios` *Simulate epidemiological exposure-profile scenarios*

Description

Builds complete chronological exposure profiles from user-defined lag-period scenarios and predicts their population-level expected outcomes with ‘predict_outcomes()‘.

Usage

```
simulate_scenarios(
  fit,
  scenarios,
  data = NULL,
  periods = NULL,
  ref_vals = NULL,
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  probs = c(0.025, 0.975),
  seed = NULL,
  extrapolation = c("warn", "error", "allow")
)
```

Arguments

`fit`	Fitted model returned by the current ‘fit_epidlnm()‘.
`scenarios`	Scenario definitions. Two input forms are supported: 1. A named list in which each top-level element is one scenario. Within a scenario, elements are named lag periods, and each period is a named list of fitted exposure variables. 2. A structured object containing a ‘$scenarios‘ component and, optionally, ‘$periods‘ and ‘$info‘ components, such as the object produced by EpiExposure scenario-generation utilities. For a period/exposure entry, a scalar applies one value uniformly to every lag in that period. A numeric vector of length greater than one represents multiple matched **scenario points**, not a lag-varying within-period trajectory. All varying vectors within the same scenario must have the same length; scalar entries are recycled across those points. If a genuinely lag-varying chronological exposure profile is required, supply that profile directly to ‘predict_outcomes(profiles = ...)‘ instead of encoding it as a period vector here.
`data`	Optional long-format exposure data used only when a scenario leaves one or more fitted exposure-lag positions unspecified and no corresponding value was supplied in ‘ref_vals‘. In that case, the median of the required fitted exposure in ‘data‘ is used as the background value. If every scenario completely specifies every fitted exposure over its full lag window, ‘data‘ is not required. ‘data‘ is never used to reconstruct, refit, or redefine a cross-basis.
`periods`	Optional data frame containing ‘period‘, ‘lag_start‘, and ‘lag_end‘. If ‘scenarios‘ is a structured object with an embedded ‘$periods‘, ‘periods‘ may be omitted. If both are supplied, their required columns must agree exactly; conflicting definitions are rejected.
`ref_vals`	Optional named list of finite numeric background values. Values are used **only** for fitted exposure-lag positions left unspecified by a scenario. The list may

contain all fitted exposures or only the exposures for which background filling is needed.

If an unassigned position requires a background value and that exposure is absent from ‘ref_vals‘, its median is derived from ‘data‘. If neither source is available, the function stops with an explicit error naming the exposure that still requires a background value.

Complete scenarios therefore do not need ‘ref_vals‘ or ‘data‘. For example, when ‘simulate_ranges()‘ copies every fitted exposure combination across non-overlapping periods that jointly cover the full fitted lag window, every exposure-lag position is already specified.

EpiExposure does not use ‘argvar$cen‘ or another spline-centering attribute as an implicit scenario background because DLNM centering and a complete epidemiological background profile are distinct concepts.

`uncertainty` Logical. If ‘FALSE‘, predict from the harmonized central fixed/population parameter estimate. If ‘TRUE‘, propagate joint fixed/population parameter uncertainty draw by draw through ‘predict_outcomes()‘.

`output` Character. ‘"summary"‘ returns deterministic predictions when ‘uncertainty = FALSE‘, or median, SD, and empirical uncertainty intervals when ‘uncertainty = TRUE‘. ‘"samples"‘ returns one row per parameter draw and scenario point and requires ‘uncertainty = TRUE‘.

`n_samples` Positive integer number of parameter draws used when ‘uncertainty = TRUE‘. At least two are required.

`probs` Numeric vector of length two defining the empirical uncertainty interval when ‘uncertainty = TRUE‘ and ‘output = "summary"‘. The default ‘c(0.025, 0.975)‘ gives a 95 percent interval.

`seed` Optional finite integer for reproducible parameter sampling. Random-number handling is delegated to ‘predict_outcomes()‘, which restores the caller’s global random-number state when it exits.

`extrapolation` Character. Behavior when a complete scenario profile contains exposure values outside the fitted cross-basis exposure range: ‘"warn"‘ (default), ‘"error"‘, or ‘"allow"‘. The fitted basis is never re-estimated from scenario values.

Details

‘simulate_scenarios()‘ is intentionally a thin scenario-construction wrapper. It does not contain engine-specific prediction code, does not refit or redefine DLNM bases, and does not summarize parameter draws independently of ‘predict_outcomes()‘. All model-specific prediction, link handling, and uncertainty propagation are delegated to the harmonized prediction layer.

Scenario interpretation

Each scenario is converted into one complete chronological exposure profile for every fitted exposure. Period-specific assignments are written first. Only positions that remain unassigned are filled from ‘ref_vals‘ or, when needed, exposure medians derived from ‘data‘.

This means a complete scenario is predicted exactly as supplied. Background values do not modify or recenter positions that the scenario already defines.

For a fitted maximum lag L, a profile has $L + 1$ elements ordered chronologically:

$$(x_L, x_{L-1}, \ldots, x_1, x_0),$$

so the final profile element corresponds to lag 0.

A scenario prediction therefore describes the expected outcome under the **entire assembled exposure profile**, not the effect of a single lag in isolation.

Population-level expected response

EpiExposure v1 reports scenario predictions from the fixed/population component of the fitted model:

$$\eta = X\beta,$$

with fitted random effects set to zero. Random effects may have contributed to model estimation, but no fitted group-specific random effect is inherited by a scenario. With response-scale output, the target is

$$E(Y \mid X) = g^{-1}(X\beta).$$

This is not integration over the random-effect distribution and is not a simulation of a future observed outcome.

Uncertainty and scenario comparisons

When ‘uncertainty = TRUE‘, all scenario profiles are passed to ‘predict_outcomes()‘ in a **single prediction call**. Consequently, parameter draw s is applied jointly to every scenario:

$$\mu_j^{(s)} = g^{-1}(X_j\beta^{(s)}),$$

where j indexes scenario points. This preserves covariance among scenario predictions and allows sample-wise differences between scenarios to be calculated correctly downstream.

The uncertainty distribution describes uncertainty in expected responses. Residual, observation, process, dispersion, posterior-predictive, and group-specific random-effect noise are not added.

Overlapping periods

‘periods‘ may contain overlapping intervals, but within a single scenario the same exposure cannot be assigned by two overlapping period blocks. Such a conflict is rejected rather than allowing one block to silently overwrite the other.

Value

A data frame containing scenario identifiers, optional scenario metadata, and population-level expected-response predictions.

‘scenario‘ identifies the named scenario and ‘scenario_point‘ identifies matched points when a scenario contains varying vectors.

With ‘uncertainty = FALSE‘, the result contains ‘prediction‘.

With ‘uncertainty = TRUE‘ and ‘output = "summary"‘, it contains ‘prediction‘, ‘prediction_sd‘, ‘prediction_lower‘, and ‘prediction_upper‘.

With ‘uncertainty = TRUE‘ and ‘output = "samples"‘, it contains ‘sample‘ and ‘prediction‘, with the same ‘sample‘ index referring to the same joint fixed/population parameter draw across all scenario points.

`spatial_replicate` *Simulated epidemiological data with independent spatial replicates*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles for fixed field locations observed across independent epidemiological replicates. The dataset is intended for demonstrating distributed exposure-lag models with separate spatial fields across epidemic replicates.

Usage

```
spatial_replicate
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique identifier combining field location and epidemiological replicate.

location_id Identifier for the field location.

epidemic_replicate Identifier for an independent epidemiological replicate.

block Experimental block identifier within a replicate.

block_id Identifier combining epidemiological replicate and experimental block.

time Chronological time index within each epidemic profile.

x_coord First numeric spatial coordinate of the field location.

y_coord Second numeric spatial coordinate of the field location.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Non-negative count disease outcome repeated across the rows of each epidemic profile.

Details

The response and location-level descriptors are repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("spatial_replicate", package = "EpiExposure")
head(spatial_replicate)
```

`spatial_year` *Simulated epidemiological data with independent spatial fields by year*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles for fixed field locations observed across multiple years. The dataset is intended for demonstrating distributed exposure-lag models with separate spatial fields by year.

Usage

```
spatial_year
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique identifier combining field location and year.

location_id Identifier for the field location.

year Year identifying the independent spatial field.

block Experimental block identifier within a year.

block_id Identifier combining year and experimental block.

time Chronological time index within each epidemic profile.

x_coord First numeric spatial coordinate of the field location.

y_coord Second numeric spatial coordinate of the field location.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Non-negative count disease outcome repeated across the rows of each epidemic profile.

Details

The response and location-level descriptors are repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("spatial_year", package = "EpiExposure")
head(spatial_year)
```

`st_poisson` *Simulated spatial poisson epidemiological dataset*

Description

A simulated longitudinal dataset containing complete environmental exposure profiles and spatial coordinates for demonstrating distributed exposure-lag models with Matérn spatial dependence and a poisson disease outcome.

Usage

```
st_poisson
```

Format

A data frame with rows representing temporal observations within epidemic exposure profiles and the following variables:

epi_id Unique numeric identifier for each epidemic.

block Experimental block identifier.

time Chronological time index within each epidemic profile.

x_coord First numeric spatial coordinate of the epidemic location.

y_coord Second numeric spatial coordinate of the epidemic location.

tmean Daily mean temperature in degrees Celsius.

wetness Daily leaf wetness duration in hours.

rain Daily rainfall in millimeters.

y Non-negative count disease outcome repeated across the rows of each epidemic profile.

Details

The response, block, and spatial coordinates are repeated across the temporal rows belonging to the same epidemic because each complete exposure profile is associated with one final epidemic-level disease outcome.

Source

Simulated for the EpiExposure package.

Examples

```
data("st_poisson", package = "EpiExposure")
head(st_poisson)
```

summarise_effects *Summarise DLNM exposure-lag effects on link and response scales*

Description

Summarises lag-specific or period-specific distributed lag nonlinear model (DLNM) associations for models fitted with ‘fit_epidlnm()‘. The function preserves the centering logic of ‘dlnm::crosspred()‘: each exposure-lag association is expressed relative to a reference exposure value (‘cen‘) for the focal exposure. EpiExposure then extends that DLNM contrast by anchoring it to a joint reference exposure profile, allowing the same association to be reported as ‘baseline‘, ‘predicted‘, and ‘delta‘ on the response scale.

Usage

```
summarise_effects(
  fit,
  data,
  vars = NULL,
  scale = c("lag", "period"),
  lag_periods = NULL,
  probs = seq(0.05, 0.95, by = 0.01),
  at = NULL,
  ref = list(method = "median", value = NULL),
  effect_measure = c("linear", "exponentiated", "percent"),
  incremental = FALSE,
  uncertainty = FALSE,
  output = c("summary", "samples"),
  n_samples = 1000,
  interval_probs = c(0.025, 0.975),
  diagnostics = FALSE,
  seed = NULL,
  extrapolation = c("error", "warn", "allow")
)
```

Arguments

fit	Fitted model returned by the current ‘fit_epidlnm()‘.
data	Long-format exposure data containing ‘epi_id‘, ‘time‘, and every exposure variable used in the fitted model. ‘data‘ is used to define default exposure grids and reference values and to validate temporal coverage. Under the EpiExposure exact-profile contract, every ‘epi_id‘ must contain exactly the common fitted ‘max_lag + 1‘ time points, and all fitted exposures therefore share the same temporal-profile length. Longer profiles are not truncated and shorter profiles are not padded. ‘data‘ is **not** used to re-estimate spline knots, boundary knots, lag bases, or any other fitted cross-basis component.
vars	Exposure-variable name or names to summarise, or ‘NULL‘ to summarise all fitted exposures.

`scale` Character. ‘"lag"‘ returns lag-specific associations. ‘"period"‘ aggregates lag-specific linear-predictor contrasts over intervals.

`lag_periods` Optional data frame containing ‘period‘, ‘lag_start‘, and ‘lag_end‘. Required when ‘scale = "period"‘ and ‘incremental = FALSE‘. Both ‘lag_start‘ and ‘lag_end‘ are interpreted on the retrospective lag scale, where lag 0 is the most recent exposure and increasing lag values correspond to older exposure observations. Period bounds are inclusive. Periods may be non-overlapping or overlapping; each period is interpreted independently.

`probs` Unique probabilities used only to construct the default exposure evaluation grid when ‘at‘ is not supplied for a variable. Defaults to ‘seq(0.05, 0.95, by = 0.01)‘.

`at` Optional exposure values at which associations are evaluated. ‘NULL‘ uses the quantiles specified by ‘probs‘. A numeric vector can be used when exactly one exposure is requested. For multiple exposures, supply a named list, for example ‘list(tmean = seq(20, 35, by = 0.25), rain = seq(0, 40, by = 1))‘. A named list may be partial; omitted requested variables use ‘probs‘.

`ref` Reference exposure specification.

Two forms are supported:

- Method-based: ‘list(method = "median", value = NULL)‘, ‘list(method = "percentile", value = p)‘, or ‘list(method = "fixed", value = x)‘.
- Exposure-specific: a named list with exactly one finite reference value for every fitted exposure, for example ‘list(tmean = 25, rain = 5, wetness = 10)‘.

Method-based references are applied consistently to **every** fitted exposure. Thus ‘method = "median"‘ uses each exposure’s own median; ‘method = "percentile"‘ uses the same percentile for each exposure; and ‘method = "fixed"‘ applies the same numeric value to every exposure. Because exposures often have different units, the exposure-specific form is recommended for multivariable models when fixed reference values are desired.

The focal exposure’s reference value is passed to the DLNM centering operation (‘cen‘). The complete set of reference values simultaneously defines the joint reference profile used for ‘baseline‘.

`effect_measure` Character. ‘"linear"‘ returns the DLNM contrast on the linear-predictor scale. ‘"exponentiated"‘ returns ‘exp(eta)‘ and ‘"percent"‘ returns ‘100 * (exp(eta) - 1)‘.

Exponentiation is allowed only for fitted ‘"log"‘ and ‘"logit"‘ links, matching the convention used by ‘dlnm::crosspred()‘. With a log link, ‘exp(eta)‘ is a response-scale ratio and ‘"percent"‘ is the corresponding percent relative change in the expected response. With a logit link, ‘exp(eta)‘ is an odds ratio and ‘"percent"‘ is the percent change in odds; neither is a percentage-point change in probability. For links such as ‘"identity"‘, ‘"probit"‘, ‘"cloglog"‘, or ‘"inverse"‘, use ‘effect_measure = "linear"‘.

`incremental` Logical. Used only with ‘scale = "period"‘. If ‘TRUE‘, return cumulative effects from lag 0 through each successive lag. If ‘FALSE‘, aggregate over ‘lag_periods‘.

uncertainty	Logical. If 'FALSE', use the harmonized central fixed/population parameter estimate. If 'TRUE', propagate joint fixed/population parameter uncertainty draw by draw.
output	Character. '"summary"' returns deterministic values when 'uncertainty = FALSE', or median, SD, and empirical intervals when 'uncertainty = TRUE'. '"samples"' returns one row per parameter draw and exposure-lag/period combination and therefore requires 'uncertainty = TRUE'.
n_samples	Positive integer number of parameter draws used when 'uncertainty = TRUE'. At least two are required.
interval_probs	Numeric vector of length two defining the empirical uncertainty interval. The default 'c(0.025, 0.975)' gives a 95 percent interval.
diagnostics	Logical. If 'TRUE', additionally return lag-ranking and lag-contribution diagnostics. Diagnostics are available only for 'scale = "lag"' and are always based on the additive linear-predictor contrast 'eta', regardless of 'effect_measure'. This avoids treating the neutral value 1 of an exponentiated effect as if it were an additive contribution.
seed	Optional finite integer used for parameter sampling. The caller's global random-number state is restored when the function exits.
extrapolation	Character controlling values outside the exposure range stored with the fitted cross-basis: '"error"' (default), '"warn"', or '"allow"'. This does not re-estimate the basis; it only controls whether extrapolation is rejected, warned about, or allowed.

Details

Exposure profiles are supplied chronologically: the oldest exposure is first and the most recent exposure is last. Internally, EpiExposure converts these profiles to the retrospective lag scale used by DLNMs, where lag 0 represents the most recent exposure observation and increasing lag values represent progressively older exposure observations. All lag-specific and period-specific summaries reported by 'summarise_effects()' are therefore indexed on this retrospective lag scale.

DLNM contrast and the EpiExposure response-scale extension

'eta' is constructed using the centering mechanism of 'dlnm::crosspred()'. The fitted cross-basis is never redefined from 'data': EpiExposure uses the stored 'argvar', 'arglag', maximum lag, and, when available, the original stored 'crossbasis' object. Consequently, 'eta' retains the standard DLNM interpretation as an association relative to 'cen'.

EpiExposure additionally defines a joint response-scale reference prediction

$$B = g^{-1}(X_{ref}\beta),$$

where every fitted exposure profile is held at its selected reference value. A lag- or period-specific DLNM contrast $\Delta\eta_{x,l}$ is then translated to the response scale as

$$P_{x,l} = g^{-1}\{g(B) + \Delta\eta_{x,l}\},$$

and

$$D_{x,l} = P_{x,l} - B.$$

Thus ‘baseline‘, ‘predicted‘, and ‘delta‘ do not replace the DLNM contrast; they provide an additional outcome-scale interpretation anchored to the same reference condition.

Uncertainty

With ‘uncertainty = TRUE‘, one **joint** fixed/population coefficient draw is used consistently across the complete requested effect surface. For draw s,

$$B^{(s)} = g^{-1}(X_{ref}\beta^{(s)}),$$

$$P_{x,l}^{(s)} = g^{-1}\{X_{ref}\beta^{(s)} + \Delta\eta_{x,l}^{(s)}\},$$

and

$$D_{x,l}^{(s)} = P_{x,l}^{(s)} - B^{(s)}.$$

The same draw therefore determines ‘baseline‘, ‘eta‘, ‘effect‘, ‘predicted‘, and ‘delta‘, preserving their covariance. ‘baseline‘ changes across parameter draws because the fitted parameters are uncertain, but for a given draw it remains the same across all exposure-lag/period rows that use the same joint reference profile. In summary output, the same baseline uncertainty summary is therefore repeated across those rows.

Parameter uncertainty includes the joint uncertainty of the fitted fixed/population coefficients. Fitted group-specific random effects are excluded because EpiExposure v1 reports population-level effects. Residual, observation, process, dispersion, and posterior-predictive noise are not added. The uncertainty interval therefore describes uncertainty in the expected response and in the exposure-lag association, not the dispersion of a future individual observation.

For uncertainty summaries, all response-scale transformations are performed draw by draw before medians, SDs, and empirical quantiles are calculated.

Exact common lag/profile contract

‘summarise_effects()‘ requires the fitted model to use one common ‘max_lag‘ across all fitted exposure variables. If that common maximum lag is ‘L‘, every epidemic profile supplied in ‘data‘ must contain exactly

$$L + 1$$

equally spaced observations. Profiles with fewer or more observations are rejected explicitly. Because all exposure variables are columns of the same validated long-format profile and missing/non-finite exposure values are not permitted, all fitted exposures necessarily use the same time support.

Period effects

Period definitions always use the retrospective lag scale employed by EpiExposure and DLNMs. Thus, lag 0 corresponds to the most recent exposure observation and increasing lag values correspond to progressively older exposure observations. For example, if ‘max_lag = 85‘, a period defined as ‘0–14‘ represents the 15 most recent exposure observations, whereas a period defined as ‘71–85‘ represents the oldest portion of the fitted exposure profile.

DLNM contributions are additive on the linear-predictor scale. Period summaries therefore sum lag-specific ‘eta‘ contrasts first and only then transform the resulting contrast, if requested. With ‘incremental = TRUE‘, periods are ‘0-0‘, ‘0-1‘, ..., ‘0-L‘. Otherwise each row of ‘lag_periods‘ defines an independent lag interval.

Value

A data frame, or when ‘diagnostics = TRUE‘, an object of class ‘"epiexposure_effects"‘ containing ‘effects‘ and ‘diagnostics‘.

The principal columns have the following meanings:

‘eta‘ DLNM association contrast on the fitted link/linear-predictor scale, centered at the focal exposure reference. For ‘scale = "lag"‘, this is the contribution of the specified exposure value at that lag relative to the same lag at the reference exposure. For period summaries, lag-specific ‘eta‘ contrasts are summed on the additive linear-predictor scale.

‘effect‘ ‘eta‘ itself for ‘effect_measure = "linear"‘, or its permitted log/logit transformation for ‘"exponentiated"‘ or ‘"percent"‘.

‘baseline‘ Population-level expected outcome under the **joint reference exposure profile**: every fitted exposure is held at its own reference value at every fitted lag and fitted random effects are set to zero. This is an EpiExposure response-scale reference prediction; it is not the DLNM contrast itself and should not be confused with the neutral DLNM values ‘eta = 0‘ or ‘exp(eta) = 1‘.

‘predicted‘ Population-level expected outcome after applying the focal DLNM contrast to that joint baseline. For a lag-specific row, only the focal exposure at that lag is changed from its reference to ‘value‘; all other lags and all other fitted exposures remain at their reference values. For a period row, the focal exposure is changed to ‘value‘ throughout the specified lag interval while all remaining exposure-lag positions remain at reference.

‘delta‘ Absolute response-scale difference ‘predicted - baseline‘. For Gaussian/Gamma-type mean responses this is a difference in expected means; for Poisson/negative-binomial outcomes it is a difference in expected counts; and for Beta/Binomial outcomes it is a difference in expected proportions/probabilities. Multiplying ‘delta‘ by 100 for Beta or Binomial models gives percentage-point differences.

Index

∗ **datasets**
  binomial_data, 3
  epi_data, 43
  gamma_data, 61
  gaussian_data, 62
  nb2_data, 63
  poisson_data, 87
  spatial_replicate, 112
  spatial_year, 113
  st_poisson, 114

binomial_data, 3
build_design, 4

check_identifiability, 6
compare_exposures, 10
compare_periods, 14
compare_predictions, 18
compute_eci, 22
compute_ecilag, 29

define_exposures, 33
define_periods, 36

ensemble_bestfit, 38
epi_data, 43
epi_sensitivity, 44

find_bestfit, 49
fit_epidlnm, 56

gamma_data, 61
gaussian_data, 62

nb2_data, 63

plot_eci, 64
plot_effects, 66
plot_losses, 70
plot_performance, 75
plot_scenarios, 79
poisson_data, 87
predict_outcomes, 88
prepare_response, 91

reduce_effects, 93

simulate_exposures, 98
simulate_losses, 101
simulate_ranges, 106
simulate_scenarios, 108
spatial_replicate, 112
spatial_year, 113
st_poisson, 114
summarise_effects, 115