Package 'ErrorTracer'

July 19, 2026

Type Package

Title Bayesian Error Propagation and Forecast Uncertainty Decomposition

Version 1.2.1

Date 2026-07-18

Description Provides a full pipeline from regularized or standard regression models (elastic net, linear models, generalized linear models, random forests) to informed Bayesian priors, structured forecast uncertainty decomposition (parameter / environmental / residual, plus a temporal component when the model carries an autocorrelation term), and forecast shelf life analysis (the quantification of when a forecast becomes uninformative). Designed for ecological and genomic forecasting with climate or environmental covariates. Methods build on Bürkner (2017) <doi:10.18637/jss.v080.i01> for Bayesian regression via 'Stan', Friedman, Hastie, and Tibshirani (2010) <doi:10.18637/jss.v033.i01> for elastic net regularization, Wright and Ziegler (2017) <doi:10.18637/jss.v077.i01> for random forests, and Vehtari, Gelman, and Gabry (2017) <doi:10.1007/s11222-016-9696-4> for leave-one-out cross-validation.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports brms (>= 2.20.0), ggplot2 (>= 3.4.0), rlang (>= 1.1.0), stats, utils

Suggests loo (>= 2.6.0), bayesplot (>= 1.10.0), scales (>= 1.3.0), tidyr (>= 1.3.0), glmnet (>= 4.1.0), ranger (>= 0.15.0), dplyr (>= 1.1.0), testthat (>= 3.0.0), knitr, rmarkdown, covr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Luis Javier Madrigal-Roca [aut, cre],
John Kelly [aut]

Maintainer Luis Javier Madrigal-Roca <madrigalrocalj@yahoo.com>

Repository CRAN

Date/Publication 2026-07-19 07:20:14 UTC

Contents

`decompose_uncertainty` *Extract or recompute uncertainty decomposition*

Description

Returns a `data.frame` with the uncertainty decomposition stored inside an `et_prediction` object:

param_var Variance of the posterior linear predictor — captures uncertainty in fitted regression coefficients.

env_var Variance arising from measurement or prediction uncertainty in the predictor values (estimated via perturbation in `et_predict`). A genuine sub-share of `total_var` (see below), not an add-on. Zero when `env_noise = NULL`.

residual_var Posterior mean of σ^2 (or its family-specific analogue) — biological process noise, unmeasured drivers, and drift. For autocorrelation models this is the *innovation* variance, not the stationary marginal variance; the autocorrelated accumulation is reported separately in `temporal_var`.

temporal_var (Only present when the model formula contains an autocorrelation term such as `ar()`, `ma()`, `arma()`, `cosy()`, `unstr()`, `sar()`, or `car()`.) Variance attributable to residual temporal or spatial dependence beyond the iid `param` + `residual` sum, computed as the excess of the sampled posterior-predictive variance over `param_var` + `residual_var`, i.e. `pmax(0, var(posterior_predict) - (param_var + residual_var))`. This captures the AR/MA/ARMA innovation accumulation that the analytic single-innovation `residual_var` does not.

total_var Total predictive variance, *defined as the sum of its components* so the budget reconciles exactly (law of total variance): `param_var` + `env_var` + `residual_var` for iid models, plus `temporal_var` for autocorrelation models. `env_var` thus enters `total_var` through the perturbed predictors, rather than being added on top of an env-free total.

Usage

```
decompose_uncertainty(predictions, ...)
```

Arguments

`predictions`	An `et_prediction` object from `et_predict`, or an `et_prediction_list` (grouped).
`...`	Unused.

Details

All variance components are guaranteed non-negative, and because `total_var` is the sum of the other components the reported percentage shares sum to 100% exactly (`param_var` + `env_var` + `residual_var` (+ `temporal_var`) == `total_var`).

Value

A `data.frame` with columns `obs_id`, `param_var`, `env_var`, `residual_var`, `total_var` (plus `temporal_var` for autocorrelation models, and a leading `group` column for grouped predictions).

Examples

```
set.seed(1)
df  <- data.frame(y = rnorm(20), x1 = rnorm(20))
fit <- et_fit(y ~ x1, data = df,
              chains = 1, iter = 500, warmup = 250,
              cores = 1, refresh = 0)
new_df <- data.frame(x1 = rnorm(5))
pred   <- et_predict(fit, newdata = new_df,
                     env_noise = list(x1 = 0.2),
                     n_draws = 200, n_perturb = 50)
decomp <- decompose_uncertainty(pred)
head(decomp)
```

`et_calibrate` *Assess calibration of posterior predictive intervals*

Description

Computes observed coverage probability at multiple nominal CI levels. A well-calibrated Bayesian model should produce 90% CIs that contain the true value 90% of the time, etc.

Usage

```
et_calibrate(predictions, observed, response_col = NULL, ci_levels = NULL, ...)
```

Arguments

`predictions`	An `et_prediction` or `et_prediction_list`.
`observed`	A `data.frame` with true response values. Must have the same number of rows as `predictions$newdata` (rows are matched positionally) and a column with the true response values.
`response_col`	Character. Name of the response column in `observed`. Defaults to the left-hand side of the model formula if it can be inferred, otherwise must be specified.
`ci_levels`	Numeric vector. CI levels to assess. Defaults to all levels present in the `et_prediction` object.
`...`	Unused.

Value

A `data.frame` with columns:

ci_level Nominal CI level.

nominal Same as ci_level.

observed_coverage Fraction of true values falling inside the CI.

n_obs Number of observations used.

calibration_error Signed difference: observed - nominal. Positive = over-coverage (CIs too wide / conservative). Negative = under-coverage (CIs too narrow / overconfident).

sharpness Mean CI width across observations. Sharpness and calibration are complementary: a model can be calibrated but useless if sharpness is poor (very wide CIs).

For grouped predictions, a `group` column is prepended.

Examples

```
set.seed(1)
df  <- data.frame(y = rnorm(20), x1 = rnorm(20))
fit <- et_fit(y ~ x1, data = df,
              chains = 1, iter = 500, warmup = 250,
              cores = 1, refresh = 0)
valid_df <- data.frame(y = rnorm(5), x1 = rnorm(5))
pred <- et_predict(fit, newdata = valid_df,
                   n_draws = 200, n_perturb = 50)
cal <- et_calibrate(pred, observed = valid_df, response_col = "y")
print(cal)
```

`et_diagnose` *Diagnose a fitted ErrorTracer model*

Description

Computes Rhat, effective sample size ratios, divergent transitions, and leave-one-out cross-validation (LOO-CV) for a fitted `et_model`.

Usage

```
et_diagnose(model, loo = TRUE, ...)
```

Arguments

`model`	An `et_model` or `et_model_list`.
`loo`	Logical. Whether to run LOO-CV (can be slow; default `TRUE`).
`...`	Unused.

Value

A list with elements:

convergence List: `rhat_max`, `rhat_all_ok`, `neff_min`, `neff_all_ok`, `n_divergences`.

loo List or `NULL`: `elpd_loo`, `p_loo`, `looic`, `n_bad_pareto_k`, `loo_object`.

summary Printed summary from `brms::summary()`.

For `et_model_list`, a named list of per-group diagnostic lists plus an aggregated `summary` data.frame.

et_fit *Fit a Bayesian regression model with informed priors*

Description

Wraps `brms::brm()` and attaches the prior specification, training data reference, and configuration for downstream uncertainty decomposition. Pass `priors` from `extract_priors` to use regularized-model coefficients to set the prior scale (and, with `shrinkage = "estimate"`, the prior mean); omit it for default (weakly informative) priors.

Usage

```
et_fit(
  formula,
  data,
  priors = NULL,
  chains = 4L,
  iter = 2000L,
  warmup = floor(iter/2),
  cores = min(chains, parallel::detectCores()),
  seed = 42L,
  adapt_delta = 0.95,
  max_treedepth = 12L,
  grouping = NULL,
  eiv = NULL,
  silent = 2L,
  ...
)
```

Arguments

`formula`	An R formula, e.g.\ `response ~ .` or `y ~ x1 + x2`.
`data`	A `data.frame` with all predictors and the response.
`priors`	An `et_prior_spec` object from `extract_priors`, or a `brmsprior` object, or `NULL` for brms defaults.
`chains`	Integer. Number of MCMC chains (default 4).
`iter`	Integer. Total iterations per chain, including warmup (default 2000).
`warmup`	Integer. Warmup iterations per chain (default `floor(iter / 2)`).
`cores`	Integer. Parallel cores (default `min(chains, parallel::detectCores())`).
`seed`	Integer. Random seed for reproducibility (default 42).
`adapt_delta`	Numeric. Target acceptance probability for HMC (default 0.95).
`max_treedepth`	Integer. Maximum tree depth (default 12).
`grouping`	Character. Name of a column in `data` to use for grouping. If non-`NULL`, one model is fitted per unique group value and an `et_model_list` is returned.

eiv	Optional errors-in-variables specification. A named list / vector mapping predictor names to either a scalar SD or a vector of per-row SDs (length nrow(data)). For each entry, the formula term for that predictor is rewritten as brms::me(pred, se_pred) (an auxiliary se_<pred> column is appended to data), so the posterior reflects measurement error in the predictor as well as coefficient uncertainty. The beta posteriors widen accordingly, which partially absorbs what ErrorTracer's downstream env_var component would otherwise report. When eiv is supplied together with an et_prior_spec from extract_priors, the informed priors are *dropped* because they target class = "b" terms and me() terms live under class = "bsp"; brms defaults are used instead (and a warning is logged).
silent	Integer passed to brms::brm() (default 2, no Stan output).
...	Additional arguments passed to brms::brm().

Value

An et_model object (or an et_model_list if grouping is specified).

Scope and limitations

et_fit() is deliberately a **thin wrapper around** brms: the formula, family, and any autocorrelation term (ar()/ma()/ arma()/...) are passed straight through, so ErrorTracer inherits brms's modelling scope and its Stan back-end. The added value is the prior pipeline, the structured uncertainty decomposition, forecast skill / shelf life, and calibration diagnostics — not new model classes.

Two boundaries to keep in mind:

- **Hierarchical / random-effects models** fit through the formula (y ~ x + (x | group)), but the current decomposition does *not* yet split out a group-level variance component, and it does not distinguish in-sample prediction (using group-specific parameters) from out-of-sample prediction (integrating over the group level). Treat the decomposition of hierarchical fits as provisional until a dedicated group-variance term is added. The grouping argument here is a *separate* device: it fits one independent model per group (no pooling), not a single multilevel model.
- **Convergence is your responsibility.** et_fit() emits a warning on high Rhat or divergent transitions, but you should inspect et_diagnose before predicting; the defaults (iter = 2000, 4 chains) are a starting point, not a guarantee.

Examples

```
set.seed(1)
df  <- data.frame(y = rnorm(20), x1 = rnorm(20), x2 = rnorm(20))
ps  <- extract_priors(lm(y ~ x1 + x2, data = df))
fit <- et_fit(y ~ x1 + x2, data = df, priors = ps,
              chains = 1, iter = 500, warmup = 250,
              cores = 1, refresh = 0)
print(fit)
```

et_pit *Probability integral transform (PIT) for calibration diagnostics*

Description

Computes the probability integral transform of held-out observations under the (reported) posterior predictive distribution — the same distribution the credible intervals and et_calibrate coverage are built from, including environmental uncertainty when it was folded in. For a well-calibrated model the PIT values are $\mathrm{Uniform}(0, 1)$. Systematic departures diagnose *how* the model is miscalibrated, which interval coverage alone cannot: a **U-shape** means the predictive is too narrow (overconfident), a central **hump** means it is too wide (underconfident), and a **slope / shift** means it is biased. Visualise with et_plot_pit (a PIT / rank histogram).

Usage

```
et_pit(predictions, observed, response_col = NULL, randomize = TRUE, ...)
```

Arguments

predictions	An et_prediction or et_prediction_list.
observed	A data.frame of true responses, positionally matched to predictions$newdata (and containing the grouping column for a grouped prediction).
response_col	Character. Response column name; inferred from the model formula when NULL.
randomize	Logical. Use the *randomized* PIT (default TRUE), which spreads the probability mass at ties so the values are uniform under calibration for **discrete** predictions (counts, binary). It is a no-op for continuous predictions, where exact ties have probability ~0. Set FALSE for the non-randomized $P(Y \leq y)$.
...	Unused.

Value

A data.frame with columns obs_id and pit (plus a leading group column for grouped predictions).

See Also

et_plot_pit, et_calibrate

et_plot_calibration *Plot calibration: observed vs nominal coverage*

Description

A well-calibrated model produces points along the 1:1 diagonal. Points above the diagonal indicate over-coverage (conservative); below indicates under-coverage (anti-conservative).

Usage

```
et_plot_calibration(cal, group_col = NULL)
```

Arguments

cal	A data.frame from et_calibrate.
group_col	Optional character. Name of a column in cal that identifies sub-groups (e.g.\ "species", "cluster_id"). When NULL (default), et_plot_calibration uses the group column if present; otherwise it auto-detects any single non-canonical column with more than one unique value and treats it as the grouping. Set to NA to force a single un-grouped series.

Value

A ggplot2 object.

et_plot_coefficients *Forest plot of regression coefficients*

Description

Compares Bayesian posterior estimates (95% CI) with the regularized coefficient values used as prior means (shown as crosses).

Usage

```
et_plot_coefficients(model)
```

Arguments

model	An et_model or et_model_list.

Value

A ggplot2 object.

et_plot_decomposition *Plot uncertainty decomposition*

Description

Produces a stacked bar chart showing the relative contributions of parameter, environmental, and residual variance for each observation, plus a fourth temporal-autocorrelation component when present in `decomp` (see `decompose_uncertainty`).

Usage

```
et_plot_decomposition(decomp, proportional = TRUE, group_col = NULL)
```

Arguments

`decomp`	A `data.frame` from `decompose_uncertainty` or directly the `$decomposition` slot of an `et_prediction`.
`proportional`	Logical. If `TRUE` (default), bars are scaled to sum to 1 (proportional contribution). If `FALSE`, raw variances are shown.
`group_col`	Character. Optional name of a grouping column in `decomp` (present for grouped predictions).

Value

A `ggplot2` object.

et_plot_forecast *Plot posterior predictive fan chart*

Description

Overlays nested credible interval ribbons on the median forecast, with optional observed values for calibration assessment.

Usage

```
et_plot_forecast(
  predictions,
  observed = NULL,
  response_col = NULL,
  time_col = NULL
)
```

Arguments

predictions	An et_prediction object.
observed	Optional data.frame with true response values. If provided, points are overlaid.
response_col	Character. Name of the response column in observed.
time_col	Character. Column in predictions$newdata used as the x-axis. Defaults to observation index.

Value

A ggplot2 object.

---

et_plot_pit *Plot a PIT / rank histogram for calibration*

Description

Histogram of the probability-integral-transform values from et_pit. Bars level with the dashed uniform reference line indicate calibration; a **U-shape** means overconfident (too-narrow) intervals, a central **hump** means underconfident (too-wide), and a **tilt** means bias. The shaded band is an approximate 95% consistency region for the bar heights under uniformity.

Usage

```
et_plot_pit(pit, bins = 10L)
```

Arguments

pit	A data.frame from et_pit (a pit column, optionally group), or a bare numeric vector of PIT values.
bins	Integer. Number of histogram bins (default 10).

Value

A ggplot object.

See Also

et_pit, et_plot_calibration

et_plot_prior_posterior *Plot prior vs posterior distributions for model coefficients*

Description

Overlays prior and posterior density for each predictor coefficient, visualising how much the data update the priors.

Usage

```
et_plot_prior_posterior(model, max_preds = 8L, n_prior_draws = 4000L)
```

Arguments

`model`	An `et_model` object.
`max_preds`	Integer. Maximum number of predictors to show (default 8). Predictors are shown in the order they appear in the prior specification.
`n_prior_draws`	Integer. Number of random draws for the prior density (default 4000).

Value

A `ggplot2` object.

et_plot_sensitivity *Plot a sensitivity profile*

Description

Visualises the output of `et_sensitivity_profile`: for each noise grid point, shows how the **environmental share** of total variance and the **forecast horizon** respond. Observed horizons are drawn as solid points; projected horizons are hollow points; lower-bound rows are drawn as upward arrows at the last informative time.

Usage

```
et_plot_sensitivity(sens, show = c("horizon", "env_share", "ratio"))
```

Arguments

`sens`	An `et_sensitivity` object from `et_sensitivity_profile`.
`show`	`"horizon"` (default) shows the shelf-life horizon; `"env_share"` shows `env_var / total_var`; `"ratio"` shows the mean CI width / plausible range.

Details

The x-axis is the noise fraction (when the grid was built from `fraction_grid`) or the grid step label otherwise. When both a numeric fraction and a descriptive label exist, the fraction is preferred for continuous x-positioning.

Value

A `ggplot2` object.

See Also

`et_sensitivity_profile`

---

et_plot_shelf_life *Plot forecast shelf life*

Description

Shows how credible interval width grows over the forecast horizon and marks the threshold beyond which the forecast is uninformative.

Usage

```
et_plot_shelf_life(sl, show_ratio = TRUE)
```

Arguments

`sl`	An `et_shelf_life` object from `shelf_life`.
`show_ratio`	Logical. If `TRUE` (default), plots the ratio (CI width / plausible range) rather than raw CI width.

Value

A `ggplot2` object.

`et_predict` *Posterior prediction with uncertainty decomposition*

Description

Generates posterior predictive draws for new observations, propagates environmental measurement uncertainty through the model, and computes credible intervals. The resulting `et_prediction` object is the input to `decompose_uncertainty`, `shelf_life`, and the plotting functions.

Usage

```
et_predict(
  model,
  newdata,
  env_noise = NULL,
  env_cov = NULL,
  env_dist = NULL,
  env_ensemble = NULL,
  n_draws = 2000L,
  ci_levels = c(0.5, 0.8, 0.9, 0.95),
  n_perturb = NULL,
  n_env_draws = 1L,
  interval_type = c("predictive", "linpred"),
  include_env_in_ci = NULL,
  ...
)
```

Arguments

`model`	An `et_model` or `et_model_list` object from `et_fit`.
`newdata`	A `data.frame` containing the predictor columns named in the model formula. For grouped models, must also contain the grouping column.
`env_noise`	Environmental measurement / prediction uncertainty. Can be: • `NULL` (default): no environmental noise. • A single numeric: applied as a fraction of each predictor's empirical SD in `newdata` (e.g.\ `0.1` means 10% noise, constant across all observations). • A named `list` or named numeric vector with one scalar per predictor: constant absolute noise SD per predictor, e.g.\ `list(Tmean = 0.5, PPT = 10)`. • A named `list` where each entry is a **numeric vector of length** `nrow(newdata)`: *time-varying* (per-row) noise SDs. Use this when predictor uncertainty grows with forecast horizon, e.g.\ from a GCM ensemble spread that increases over time: `list(Tmean = 0.30 + 0.01 * (years - base_year))`. Entries not supplied default to zero (no noise for that predictor).
`env_cov`	Correlation structure of the environmental noise. The *magnitudes* of the noise come from `env_noise`; `env_cov` supplies the *correlation* between predictors, so

that a perturbation on row i is drawn from $\mathcal{N}(0,\ D_i R D_i)$ with $D_i = \mathrm{diag}(\sigma_{i1}, \ldots, \sigma_{ip})$ and R = the correlation matrix. One of:

- `NULL` (default): independent noise, $R = I$ — equivalent to ErrorTracer behaviour prior to this feature and the right choice when predictor measurement errors are genuinely independent (e.g.\ separate instruments on unrelated variables).
- `"empirical"`: compute the correlation of the predictor columns in the **training data** (`model$data`). Use this when predictor *errors* are expected to inherit the correlation structure of the predictors themselves — e.g.\ temperature and humidity that co-vary in the underlying climate system.
- `"newdata"`: compute the correlation of the predictor columns in `newdata`. Useful when the forecast window has a different covariance structure than training (e.g.\ scenario runs).
- A numeric $p \times p$ matrix with `dimnames` matching the model's predictors. Entries with an off-diagonal exceeding 1 are rescaled to a correlation matrix. Use this to supply an independent estimate of the *error* correlation structure (e.g.\ from a reanalysis product or a sensor covariance report).

A correlation derived from training data is a working assumption: the structure of the *errors* is assumed to mirror the structure of the *values*. When this is implausible, pass a matrix directly.

`env_dist` Distributional form of the per-predictor noise. The `env_noise` SDs set the *magnitude* of the perturbation; `env_dist` sets its *shape*. For every distribution other than `"gaussian"`, the noise is calibrated so that (approximately) $E[\tilde{x}] = x$ and $\mathrm{Var}[\tilde{x}] = \sigma^2$, using a Gaussian copula to honour `env_cov`. One of:

- `NULL` (default): `"gaussian"` for every predictor — additive Gaussian noise, legacy behaviour.
- A single string (`"gaussian"`, `"lognormal"`, `"gamma"`, `"beta"`): applied to all predictors.
- A named list / character vector with one entry per predictor to override the default, e.g.\ `list(PPT = "gamma", tmax = "gaussian")`.

Distributions:

`"gaussian"` Additive normal noise ($\tilde{x} = x + \varepsilon, \varepsilon \sim N(0, \sigma^2)$). Appropriate for symmetric measurement error on a continuous, potentially negative scale (temperature, anomalies).

`"lognormal"` Multiplicative noise: $\log \tilde{x} \sim N(\log x - s^2/2,\ s^2)$ with $s^2 = \log(1 + (\sigma/x)^2)$. Preserves positivity; right-tail skewed. Natural for strictly positive continuous variables whose error scales with magnitude (e.g.\ enzyme activity, biomass). Rows with $x \leq 0$ are left unperturbed.

`"gamma"` $\tilde{x} \sim \mathrm{Gamma}(\mathrm{shape} = (x/\sigma)^2,\ \mathrm{rate} = x/\sigma^2)$. Positive support, right-skewed, analytic mean/variance match. Natural for precipitation, rates, and other non-negative continuous variables. Rows with $x \leq 0$ are left unperturbed.

`"beta"` $\tilde{x} \sim \mathrm{Beta}(\alpha, \beta)$ with $\alpha + \beta = x(1-x)/\sigma^2 - 1$. Support in $(0, 1)$; appropriate for proportions and probabilities (allele frequencies, presence rates). Rows with $x \notin (0, 1)$ or $\sigma^2 \geq x(1-x)$ are left unperturbed.

Correlation (`env_cov`) is applied to the latent standard-normal draws before the marginal quantile transform, so rank correlations are preserved across distributions.

`env_ensemble` Optional pathwise driver ensemble for forecasting under uncertain future covariates. A **named list**, one entry per uncertain driver predictor, where each entry is an $M \times H$ matrix of M scenario trajectories over the $H = \mathrm{nrow}(\mathrm{newdata})$ forecast steps (rows are scenarios, columns are horizon steps). Unlike `env_noise` — which draws *independent* per-observation jitter that averages out and leaves the driver variance flat with lead time — each posterior draw here is paired with a whole *trajectory*, so the driver uncertainty is temporally coherent and **accumulates with lead time** as a real driver ensemble (e.g.\ CMIP members, or resampled climatology) fans out. Predictors absent from `env_ensemble` are held at their `newdata` values. When supplied, `env_ensemble` supersedes `env_noise` for the perturbation and is folded into the credible interval by default. This is the recommended way to build a genuine *reforecast* in which only information available at the issue date is used.

`n_draws` Integer. Number of posterior draws to use (default 2000; capped at the number of draws available in the fit).

`ci_levels` Numeric vector. Credible interval levels to compute (default `c(0.5, 0.8, 0.9, 0.95)`).

`n_perturb` Integer. Number of posterior draws used for the environmental perturbation step (default `min(500, n_draws)`). Reducing this speeds up computation.

`n_env_draws` Integer. Number of independent environmental perturbations averaged *per posterior draw* when estimating `env_var` (default `1`). Increasing this reduces Monte Carlo noise on the environmental-variance estimate at the cost of proportional computation. The decomposition always reports a Monte Carlo SE (`v_env_mcse`) alongside `env_var`; it decreases roughly like $1/\sqrt{n\_env\_draws \cdot n\_perturb}$.

`interval_type` Character. Which draws to use when computing credible intervals:

- `"predictive"` (default): draws from `posterior_predict`, which include sigma (residual noise). Use this when forecasting **individual observations** — e.g. a single population's allele frequency, one site's ozone reading on a specific day.
- `"linpred"`: draws from `posterior_linpred`, which capture only parameter uncertainty (no sigma). Use this when forecasting the **mean response** — e.g. the expected ozone across many similar days, or mean delta f across replicate populations. These intervals are always narrower; they will undercover individual observations unless sigma is negligible.

The decomposition components and `posterior_predict` / `posterior_linpred` matrices are always computed regardless of this setting.

`include_env_in_ci`

Logical or `NULL`. Controls whether environmental (predictor) uncertainty is folded into the credible interval when `interval_type = "predictive"`. When `TRUE`, intervals are built from **environmentally inflated** draws in which each posterior-predictive residual is re-centred on the env-perturbed response-scale mean, $\tilde{y} = \tilde{\mu} + (y^{pp} - \mu)$ (family-general; for Gaussian identity this equals $\tilde{\mathrm{lp}} + \varepsilon$, $\varepsilon \sim N(0, \sigma^2)$). The interval variance is then param + env + residual, coherent with

the env-inclusive `total_var`. The default `NULL` means **auto**: env is folded in whenever the caller supplied non-zero `env_noise`, and omitted otherwise. Pass `FALSE` to force a parameter + residual interval even when `env_noise` is given.

`...` Passed to methods.

Value

An `et_prediction` object (list) containing:

`posterior_predict` Matrix `[n_draws x n_obs]`: full posterior predictive draws (parameter + residual uncertainty).

`posterior_linpred` Matrix `[n_draws x n_obs]`: linear predictor draws on the **link scale** (parameter uncertainty only).

`lp_perturbed` Matrix `[n_perturb x n_obs]`: linear predictor on the link scale computed on environmentally perturbed inputs.

`sigma_draws` Numeric vector: posterior draws of sigma (`NA` for families without a sigma parameter, e.g.\ Binomial).

`credible_intervals` data.frame with columns `row_id`, `ci_level`, `lower`, `median`, `upper`, `width`.

`decomposition` data.frame with columns `obs_id`, `total_var`, `param_var`, `env_var`, `v_env_mcse`, `residual_var`. All components are on the **response scale**. `v_env_mcse` is the Monte Carlo SE of `env_var`. `residual_var` is per-observation for non-Gaussian families (e.g.\ Binomial: $E_s[\mu^{(s)}(1-\mu^{(s)})]$) and constant for Gaussian.

`newdata` The input `newdata`.

`model` Reference to the `et_model` used.

`env_cov` The $p \times p$ correlation matrix actually used for perturbation (identity for `env_cov = NULL`).

`env_dist` Named character vector mapping each predictor to the distribution actually used for its perturbation.

See Also

`decompose_uncertainty`, `shelf_life`, `et_calibrate`

---

et_sensitivity_profile

Sensitivity profile of the environmental uncertainty component

---

Description

Sweeps a grid of noise magnitudes, re-runs `et_predict` for each, and summarises how the environmental variance component and the forecast shelf life respond. This turns the subjective choice of `env_noise` into an auditable *curve*: instead of reporting a single shelf life at one (often hand-picked) measurement-error level, the user sees how the horizon degrades as assumed noise grows.

Usage

```
et_sensitivity_profile(
  model,
  newdata,
  response_scale,
  fraction_grid = NULL,
  absolute_grid = NULL,
  env_cov = NULL,
  env_dist = NULL,
  include_env_in_ci = TRUE,
  ci_level = 0.9,
  ci_levels = c(0.5, 0.8, 0.9, 0.95),
  threshold = 1,
  time_col = NULL,
  n_draws = 2000L,
  n_perturb = NULL,
  max_extrapolation_factor = 10,
  verbose = TRUE,
  plausible_range = NULL
)
```

Arguments

model	An et_model from et_fit.
newdata	data.frame passed to et_predict.
response_scale	Two-element numeric vector c(min, max), as in shelf_life. The effective range is diff(response_scale).
fraction_grid	Optional numeric vector of scalar noise fractions.
absolute_grid	Optional list of env_noise arguments (each a named list / vector). If supplied together with fraction_grid, fraction_grid is ignored.
env_cov, env_dist	Passed through to et_predict.
include_env_in_ci	Logical. If TRUE (default), credible intervals — and hence the shelf-life horizon — are computed from environmentally inflated draws (et_predict(..., include_env_in_ci = TRUE)). Set FALSE to see how env_var evolves while holding the CI fixed at parameter + residual.
ci_level	Numeric. Used for shelf life and CI width tracking (default 0.90). Must be in ci_levels.
ci_levels	Numeric vector of CI levels computed per iteration (default c(0.5, 0.8, 0.9, 0.95)).
threshold	Numeric. Shelf-life threshold (default 1.0).
time_col	Character. Optional time column for shelf life.
n_draws, n_perturb	Passed to et_predict.

`max_extrapolation_factor`
: Passed to `shelf_life`.

`verbose`
: Logical. If `TRUE` (default), logs each iteration.

`plausible_range`
: Deprecated. Use `response_scale` instead.

Details

Three argument styles are supported for the grid:

- `fraction_grid`: scalar fractions of each predictor's SD in `newdata`. These are passed straight to the scalar form of `et_predict`'s `env_noise`. Use this for an apples- to-apples sweep that scales all predictors together.
- `absolute_grid`: a `list` of named numeric lists / vectors; each list element becomes a single `env_noise` argument (so you can sweep absolute SDs for one predictor while holding others fixed).
- Neither supplied (`NULL`): a default log-spaced fraction grid `c(0, 0.01, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8)` is used.

Each iteration calls `et_predict(..., env_noise = g)` then `shelf_life` and records:

- Mean parameter / environmental / residual / total variance across forecast observations.
- The horizon description (`"observed"`, `"projected"`, or `"lower_bound"`) and numeric horizon if any.
- Mean / max CI width / plausible-range ratio.

Value

An `et_sensitivity` object: a `data.frame` with one row per grid point and columns

`grid_id` 1-indexed step.

`label` Short descriptor of the env_noise used (e.g.\ `"fraction = 0.05"`).

`fraction` Scalar fraction when applicable; `NA` for absolute-grid rows.

`param_var, env_var, residual_var, total_var` Mean across forecast observations.

`env_share` `env_var / total_var` — the fraction of total predictive variance attributable to predictor noise. Because `total_var` now contains `env_var` as a sub-share (law of total variance), this is bounded in $[0, 1]$.

`ci_width_mean, ci_width_max` At `ci_level`.

`ratio_mean, ratio_max` CI width / plausible range.

`horizon_type` One of `"observed"`, `"projected"`, `"lower_bound"`.

`horizon` Numeric horizon (observed or projected) or `NA` for lower-bound rows.

`horizon_description` The long description returned by `shelf_life`.

The returned object carries the original call and `response_scale` as attributes for use by `et_plot_sensitivity`.

See Also

`et_predict`, `shelf_life`, `et_plot_sensitivity`

`et_sim` *Simulated allele-frequency-change dataset for ErrorTracer tutorials*

Description

A named list containing training data, forecast predictors, validation responses, and ground-truth parameters for two synthetic SNP clusters (A and B) at a hypothetical mountain plant site. The dataset is designed to exercise every step of the ErrorTracer pipeline and to support parameter-recovery validation (true coefficients are known).

Usage

```
et_sim
```

Format

A named `list` with seven elements:

`train` `data.frame` with 100 rows and 6 columns (2 clusters $\times$ 50 training years, 1970–2019):

year Integer. Calendar year.

cluster_id Character. SNP cluster identifier: `"A"` or `"B"`.

Tmean Numeric. Standardised mean growing-season temperature (original unit: °C; standardised using training-period mean and SD).

PPT Numeric. Standardised total growing-season precipitation (original unit: mm).

SWE Numeric. Standardised peak snow water equivalent (original unit: mm).

z_diff Numeric. Simulated allele-frequency change on the arcsin-sqrt scale ($\arcsin(\sqrt{f})$ transformation). This transformation is unbounded and has no fixed $[-1, 1]$ constraint; the plausible range should be derived from the training data or from the theoretical bounds ($[0, \pi/2]$\ on the arcsin scale).

`forecast` `data.frame` with 30 rows and 5 columns (2 clusters $\times$ 15 forecast years, 2020–2034). Same columns as `train` except `z_diff` is absent — these are the prediction targets. Predictors are standardised using the training-period statistics stored in `standardization`.

`validation` `data.frame` with 30 rows and 3 columns (`year`, `cluster_id`, `z_diff`). True response values for the forecast period; used with `et_calibrate` to assess posterior predictive coverage.

`true_params` Named `list` with one element per cluster. Each element is a named numeric vector of the true data-generating parameters: `intercept`, `Tmean`, `PPT`, `SWE`, and `sigma` (residual SD).

Cluster A `intercept` = `0.00`, `Tmean` = `0.50`, `PPT` = `-0.30`, `SWE` = `0.20`, `sigma` = `0.30`

Cluster B `intercept` = `0.10`, `Tmean` = `0.30`, `PPT` = `-0.20`, `SWE` = `-0.25`, `sigma` = `0.35`

`env_noise` Named `list`. Suggested per-predictor environmental noise SDs (on the standardised scale) for use with the `env_noise` argument of `et_predict`: `Tmean` = `0.30`, `PPT` = `0.20`, `SWE` = `0.15`.

`standardization` Named `list` with one element per predictor. Each element is a named numeric vector `c(mean = ..., sd = ...)` giving the training-period statistics used to standardise the data. Needed if you wish to back-transform predictions to the original (unstandardised) scale with `unstandardize`.

`description` Character string. Human-readable description of the dataset and how it was generated.

Details

The climate time series are simulated as AR(1) processes with a warming trend in `Tmean` (+0.015 °C yr^{-1}, cumulating to ~0.30 SD above the training mean over the 15-year forecast window). `SWE` is generated with a weak dependence on `Tmean` (negative coupling) and `PPT` (positive coupling) plus a dominant independent noise component, so that the three predictors have only mild pairwise correlation ($|R| < 0.2$ on the training subset). This makes all regression coefficients reliably identifiable in a single replicate. All predictors are standardised using training-period statistics only.

In addition to the three real climate predictors, the dataset includes ten independent standardised nuisance predictors `d1`, ..., `d10` with zero true effect on the response. They give the regularized-prior extraction step a real selection job: cv.glmnet (alpha = 0.5) is expected to identify the three real predictors and shrink the ten dummies toward zero. Without them, regularization on three well-identified predictors merely biases the true coefficients without selecting anything.

Responses are generated from a linear model with Gaussian noise:

$$z\_diff_t = \alpha + \beta_1 Tmean_t + \beta_2 PPT_t + \beta_3 SWE_t + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

The true parameters are stored in `et_sim$true_params` for parameter-recovery validation.

The dataset was generated with `set.seed(111)`. The full generation script is at `data-raw/generate_et_sim.R`.

Source

Generated by `data-raw/generate_et_sim.R` with `set.seed(111)`.

Examples

```
data(et_sim)

# Inspect structure
str(et_sim, max.level = 2)

# Training data
head(et_sim$train)

# True parameters for cluster A
et_sim$true_params$A

# Suggested noise SDs for et_predict()
et_sim$env_noise
```

`et_skill_score` *Null-model-relative forecast skill (CRPS)*

Description

Scores a forecast against a null model with the continuous ranked probability score (CRPS) and returns the per-lead-time skill, $1 - \mathrm{CRPS}_{\mathrm{model}}/\mathrm{CRPS}_{\mathrm{null}}$ (positive when the model beats the null). This is the accuracy-relative counterpart to `shelf_life`'s precision criterion; report both. The **forecast limit** — the first lead time at which skill drops to zero — is attached as an attribute.

Usage

```
et_skill_score(
  predictions,
  observed,
  response_col = NULL,
  time_col = NULL,
  null = c("climatology", "random_walk"),
  climatology = NULL,
  rw_sd = NULL,
  rw_start = NULL,
  ...
)
```

Arguments

`predictions`	An `et_prediction` or `et_prediction_list`.
`observed`	A `data.frame` of true responses positionally matched to `predictions$newdata` (with the grouping column for grouped input).
`response_col`	Character. Response column name; inferred from the model formula when `NULL`.
`time_col`	Character. Optional time column in `newdata` used for the `time` axis (lead time otherwise defaults to the row index).
`null`	Character. The null model: `"climatology"` (default; the unconditional response distribution) or `"random_walk"` (persistence: the last observed value carried forward with variance growing linearly with lead time).
`climatology`	Optional numeric vector giving the climatological reference distribution. Defaults to the training response values.
`rw_sd`, `rw_start`	Optional random-walk per-step innovation SD and start value. Default to `sd(diff(train_response))` and the last training response, respectively.
`...`	Unused.

Value

A `data.frame` with columns `obs_id`, `time`, `crps_model`, `crps_null`, `skill`, and `skillful` (`skill > 0`); a leading `group` column for grouped input. The attribute `"forecast_limit"` holds a list with the null-relative horizon (first non-skillful lead time), or is per-group for grouped input.

See Also

shelf_life, et_calibrate

`et_sobol` *Global (Sobol) variance-based sensitivity of the predictive mean*

Description

An optional, order-independent alternative to the additive budget from `decompose_uncertainty`. Decomposes the variance of the response-scale predictive mean $\mu = g^{-1}(\eta)$ into first-order **parameter** and **environmental** (driver) Sobol indices plus their **interaction**, by independently resampling the parameter draws and the predictor perturbations (Saltelli 2010 first-order; Jansen total-order). Unlike the sequential additive split, this is insensitive to the order in which factors are added and makes any parameter×driver interaction explicit — the general case a non-linear link or correlated drivers can produce.

Usage

```
et_sobol(predictions, n_sobol = 1024L, seed = NULL, ...)
```

Arguments

`predictions`	An `et_prediction` that was produced with non-zero `env_noise` (the environmental factor needs a perturbation scale).
`n_sobol`	Integer. Base sample size for the Sobol estimator (default 1024). Total model evaluations per observation are `4 * n_sobol`.
`seed`	Optional integer for reproducible sampling.
`...`	Unused.

Value

A `data.frame` with one row per forecast observation:

obs_id Observation index.

S_param, S_env First-order Sobol indices (fraction of the predictive-mean variance from parameters / drivers alone).

ST_param, ST_env Total-order indices (including interactions).

interaction `1 - S_param - S_env`: the share of predictive-mean variance attributable to the parameter×driver interaction.

var_mu Variance of the predictive mean (the quantity being decomposed).

residual_var Mean within-draw family variance (the irreducible part; `var_mu + residual_var` approximates the total predictive variance).

See Also

`decompose_uncertainty` (the default additive budget).

et_theme — *Minimal ggplot2 theme for ErrorTracer plots*

Description

Minimal ggplot2 theme for ErrorTracer plots

Usage

```
et_theme(base_size = 12)
```

Arguments

base_size	Base font size (default 12).

Value

A ggplot2 theme object.

extract_priors — *Extract brms prior specification from a regularized or standard regression model*

Description

Converts a fitted model object into a brms prior specification suitable for et_fit. The regularized or standard fit sets the *scale* of each coefficient's prior (its SD); the shrinkage argument controls the prior *mean*. With the default shrinkage = "zero" the priors are centred at zero — a regularizing (ridge-style) prior whose width is informed by the fit but whose location carries no reused point estimate. With shrinkage = "estimate" the coefficient estimates are used as informative prior means, so the Bayesian model starts close to the regularized solution.

Usage

```
extract_priors(
  model,
  multiplier = 2,
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  ...
)

## S3 method for class 'lm'
extract_priors(
```

```
  model,
  multiplier = 2,
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  ...
)

## S3 method for class 'glm'
extract_priors(
  model,
  multiplier = 2,
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  ...
)

## S3 method for class 'cv.glmnet'
extract_priors(
  model,
  multiplier = 2,
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  lambda = "lambda.min",
  ...
)

## S3 method for class 'glmnet'
extract_priors(
  model,
  multiplier = 2,
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  s = 1L,
  ...
)

## S3 method for class 'ranger'
extract_priors(
  model,
  multiplier = 2,
```

```
  min_sd = 0.1,
  shrinkage = c("zero", "estimate"),
  intercept_prior_sd = NULL,
  sigma_prior_scale = 1,
  ...
)
```

Arguments

model	A fitted model. Supported classes: • cv.glmnet / glmnet — elastic net / lasso • lm — ordinary least squares • glm — generalized linear model • ranger — random forest (importance-scaled flat priors)
multiplier	Numeric scalar. Prior SD is set to multiplier * |coef| (for signed-coefficient methods) or multiplier * importance_normalised (for ranger). Default 2.0.
min_sd	Numeric scalar. Minimum prior SD to avoid degenerate (spike) priors on near-zero coefficients. Default 0.1.
shrinkage	Character. Sets the prior *mean*: "zero" (default) centres every coefficient prior at 0 (a regularizing prior with a data-informed scale but no reused location); "estimate" uses the fitted coefficient as the prior mean (informative, but reuses the data — see the double-use note above). For ranger the mean is always 0 (no signed coefficients), so shrinkage has no effect.
intercept_prior_sd	Optional prior SD for the intercept term. The default NULL emits *no* explicit intercept prior and lets brms pick its data-aware default (student_t(3, median(y), mad(y))). If you supply a numeric value, the prior becomes normal(0, intercept_prior_sd) — this is only appropriate when the response has been centred (or nearly so) before fitting, since brms's class = "Intercept" refers to the intercept at the centre of the predictors, whose posterior mean equals E[y] there. Supplying a small intercept_prior_sd for an uncentred response will pull the intercept toward zero and cripple the fit.
sigma_prior_scale	Scale parameter for the half-Cauchy prior on the residual SD sigma. Default 1.0.
...	Additional arguments passed to methods.
lambda	Which lambda to extract coefficients from when the model is a cv.glmnet object. Either "lambda.min" (default) or "lambda.1se".
s	Index (integer) or value of lambda to extract coefficients at when the model is a plain glmnet object. Defaults to the first lambda (smallest regularisation).

Details

The ranger **path is experimental.** Random-forest permutation importance has no sign and no natural scale in the units of a GLM coefficient, so the mapping used here — priors centred at zero with SD set to multiplier * importance_normalised (importance normalised to [min_sd, 1];

only variables with positive importance retained) — is a heuristic, not a calibrated correspondence. It is offered for exploratory feature screening only; for a principled prior prefer `glmnet/ lm/glm`, whose coefficients are on the model's scale. A signed importance (e.g. from SHAP values) would be needed to inform the prior mean.

Value

An `et_prior_spec` list containing:

`prior` A `brmsprior` object for use in `brms::brm()`.

`pred_names` Character vector of included predictor names.

`coefs` Named numeric vector of regularized coefficients (NULL for ranger, which uses importance instead).

`method` Character: the dispatch method used.

`multiplier, min_sd` Settings echo.

Prior/likelihood double use

Centring the priors on the coefficient *estimates* (`shrinkage = "estimate"`) and then refitting the Bayesian model on the *same* data uses those data twice — once to locate the prior mean and once through the likelihood — which can make posteriors overconfident, especially at small n. The default `shrinkage = "zero"` avoids this by not reusing the estimated location: only the prior scale is data-informed, which is a standard weakly-informative device. If you use `shrinkage = "estimate"`, check calibration with `et_calibrate`, or build the priors on data held out from the likelihood fit.

Examples

```
fit_lm <- lm(mpg ~ wt + hp + cyl, data = mtcars)
ps <- extract_priors(fit_lm, multiplier = 2, min_sd = 0.1)
print(ps)
```

---

shelf_life *Compute the forecast shelf life*

Description

Quantifies *when* a forecast becomes uninformative by comparing the width of credible intervals to a response scale. A forecast is uninformative when its CI width exceeds `threshold * response_scale`.

Usage

```
shelf_life(
  predictions,
  response_scale = NULL,
  ci_level = 0.9,
  threshold = 1,
```

```
  time_col = NULL,
  skill = NULL,
  min_slope_for_projection = 1e-04,
  max_extrapolation_factor = 10,
  ...,
  plausible_range = NULL
)
```

Arguments

predictions	An et_prediction or et_prediction_list.
response_scale	Numeric vector of length 2 (c(min, max)) giving the response scale used as the denominator in the CI-width / range ratio. Supply a biologically motivated interval when one exists. When NULL (default), the observed range of the **training response** is used as a documented, reproducible default (with an informational message) instead of an arbitrary number. The effective range is diff(response_scale).
ci_level	Numeric. The credible interval level to use (default 0.90). Must be present in the et_prediction object.
threshold	Numeric. CI width / response scale above which the forecast is uninformative (default 1.0).
time_col	Character. Optional column in predictions$newdata to use as the time axis. If NULL, observation index is used.
skill	Optional skill gate. A data.frame returned by et_skill_score (positionally matched to the forecast rows; its skillful column is used) or a bare logical vector. When supplied, a forecast period is informative only if it is both precise *and* skillful, so a precise-but-biased forecast does not pass. When NULL (default) the precision criterion is used alone.
min_slope_for_projection	Numeric. Minimum linear slope (of ratio vs.\ time) required to attempt extrapolation when all periods are informative. Below this value the shelf life is reported as a lower bound only. Default 1e-4.
max_extrapolation_factor	Numeric. Cap on how far the linear projection may reach beyond the observed window. If the projected crossing time exceeds max(time) + max_extrapolation_factor * (max(time) - min(time)), the result is reported as a lower bound instead of a projection. Set to Inf to disable the cap. Default 10.
...	Unused.
plausible_range	Deprecated. Use response_scale instead.

Details

Shelf life is a **precision** criterion on the response scale — a deliberate *complement* to accuracy-relative forecast limits (Petchey et al. 2015; Wesselkamp et al. 2025), not a replacement. It answers "when do the error bars get too wide to act on?", whereas a forecast limit answers "when does the model stop beating a null?". Because a forecast can be precise yet biased, pass a skill table from

et_skill_score to **gate** the shelf life on skill: a period then counts as informative only if it is both precise *and* beats the null model. This couples the two views and prevents a precise-but-biased forecast from silently passing.

The function operates in three modes depending on the available data and whether the uninformative threshold is crossed within the forecast window:

Observed The threshold is crossed within the forecast/validation window. The shelf life is the first time point at which `ratio >= threshold`.

Projected All forecast periods remain informative but the CI/range ratio is trending upward. A linear trend is fitted to the ratios and extrapolated to estimate when the threshold would be reached. The projected crossing time $t^* = (\tau - a)/b$ (where τ is the threshold, a the fitted intercept, b the fitted slope) is reported together with a Monte Carlo standard error `se_t_star` derived via the delta method.

Lower bound All forecast periods are informative with no upward trend in the ratio. The shelf life is reported as a lower bound: `> last observed time`.

The intended workflow is:

1. Fit the model (et_fit).
2. Predict on held-out data or a future time window (et_predict).
3. Call `shelf_life()` on those predictions.
4. If the threshold is not crossed in the held-out window, the *projected* mode extrapolates the horizon automatically.

Value

An `et_shelf_life` object (a `data.frame`) with columns:

obs_id Observation index.

time Time axis value.

ci_width Width of the credible interval at `ci_level`.

plausible_range Effective response scale (scalar diff).

ratio CI width / response scale.

informative Logical; `TRUE` when ratio < threshold.

For grouped predictions a `group` column is prepended. The object carries three attributes that drive `print()`:

`horizon` Named list with elements `value`, `type` (`"observed"`, `"projected"`, or `"lower_bound"`), `last_informative`, `description`, and (for projected) `se_t_star`.

`horizon_by_group` For grouped objects: named list of per-group horizon lists.

`threshold` The threshold value used.

Examples

```
set.seed(1)
df  <- data.frame(y = rnorm(20), year = 2001:2020, x1 = rnorm(20))
fit <- et_fit(y ~ x1, data = df,
              chains = 1, iter = 500, warmup = 250,
              cores = 1, refresh = 0)
new_df <- data.frame(x1 = rnorm(10), year = 2021:2030)
pred   <- et_predict(fit, newdata = new_df,
                     n_draws = 200, n_perturb = 50)
sl <- shelf_life(pred,
                 response_scale            = range(df$y),
                 ci_level                  = 0.90,
                 threshold                 = 1.0,
                 time_col                  = "year",
                 max_extrapolation_factor  = 10)
print(sl)
```

---

standardize *Standardize a numeric vector (mean-centre, unit-variance)*

Description

Standardize a numeric vector (mean-centre, unit-variance)

Usage

```
standardize(x)
```

Arguments

x	Numeric vector.

Value

Standardized numeric vector. Returns zeros if variance is zero.

---

unstandardize *Reverse standardization*

Description

Reverse standardization

Usage

```
unstandardize(z, mu, s)
```

Arguments

z	Standardized values.
mu	Original mean.
s	Original standard deviation.

Value

Values on the original scale.

Index