Package: BayesianDEB (via r-universe)

October 27, 2026

Title Bayesian Dynamic Energy Budget Modelling

Version 0.2.1

Date 2026-06-17

Description Provides a Bayesian framework for Dynamic Energy Budget (DEB) modelling via 'Stan'. Implements the standard DEB model of Kooijman (2010, <doi:10.1017/CBO9780511805400>) as a state-space model with Hamiltonian Monte Carlo inference (Carpenter et al., 2017, <doi:10.18637/jss.v076.i01>). Includes individual-level growth models, growth-reproduction models, hierarchical multi-individual models with partial pooling, and toxicokinetic-toxicodynamic (TKTD) models for ecotoxicology following the DEBtox framework (Jager et al., 2006, <doi:10.1007/s10646-006-0060-x>). Supports prior specification from biological knowledge, convergence diagnostics (Vehtari et al., 2021, <doi:10.1214/20-BA1221>), posterior predictive checks, derived quantity estimation, and visualisation via 'ggplot2'.

License MIT + file LICENSE

URL https://github.com/sciom/BayesianDEB

BugReports https://github.com/sciom/BayesianDEB/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements CmdStan (https://mc-stan.org/users/interfaces/cmdstan)

Depends R (>= 4.1.0)

Imports posterior (>= 1.5.0), rlang (>= 1.1.0), cli (>= 3.6.0), ggplot2 (>= 3.4.0), bayesplot (>= 1.10.0), deSolve (>= 1.40), stats, utils

Suggests cmdstanr (>= 0.8.0), testthat (>= 3.0.0), loo (>= 2.6.0), digest, dplyr (>= 1.1.0), tidyr (>= 1.3.0), gridExtra, knitr, rmarkdown, withr

Additional_repositories https://stan-dev.r-universe.dev

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-06-29 18:51:08 UTC

RemoteUrl https://github.com/sciom/BayesianDEB

RemoteRef HEAD

RemoteSha 06cffd795cc6e019510c3b5fe8b65df4ec41cd77

Contents

arrhenius . . . 3
bdeb_data . . . 4
bdeb_derived . . . 5
bdeb_diagnose . . . 7
bdeb_ec50 . . . 9
bdeb_fit . . . 10
bdeb_loo . . . 12
bdeb_model . . . 14
bdeb_plots . . . 15
bdeb_ppc . . . 16
bdeb_prior-methods . . . 17
bdeb_prior_predictive . . . 18
bdeb_session_info . . . 19
bdeb_tox . . . 20
coef.bdeb_fit . . . 21
confint.bdeb_fit . . . 22
deb_fluxes . . . 23
deb_simulate . . . 24
debtox_growth . . . 25
debtox_simulate . . . 26
eisenia_cd . . . 27
eisenia_growth . . . 28
eisenia_neuhauser . . . 29
fitted.bdeb_fit . . . 30
folsomia_repro . . . 31
logLik.bdeb_fit . . . 32
nobs.bdeb_fit . . . 33
observation_models . . . 33
plot.bdeb_data . . . 34
plot.bdeb_diagnostics . . . 35
plot.bdeb_fit . . . 36
plot.bdeb_model . . . 37

plot.bdeb_ppc . . . 38
plot.bdeb_prediction . . . 39
plot_dose_response . . . 39
predict.bdeb_fit . . . 41
print.bdeb_data . . . 42
print.bdeb_diagnostics . . . 42
print.bdeb_fit . . . 43
print.bdeb_model . . . 44
print.bdeb_ppc . . . 44
print.bdeb_prediction . . . 45
prior_default . . . 45
prior_species . . . 46
priors . . . 47
repro_to_intervals . . . 49
residuals.bdeb_fit . . . 50
summary.bdeb_data . . . 51
summary.bdeb_diagnostics . . . 51
summary.bdeb_fit . . . 52
summary.bdeb_model . . . 53
summary.bdeb_prediction . . . 54
vcov.bdeb_fit . . . 55

Index **56**

`arrhenius` *Arrhenius Temperature Correction*

Description

Computes the temperature correction factor for DEB rate parameters based on the Arrhenius relationship (Kooijman, 2010, Eq. 1.2). In DEB theory all rate parameters (e.g., $\{p_{Am}\}$, $[p_M]$, v) scale with temperature by the same factor:

Usage

```
arrhenius(temp, T_ref = 293.15, T_A = 8000)
```

Arguments

`temp`	Body (or ambient) temperature in Kelvin. Renamed from `T` in version 0.1.4 to avoid shadowing R's built-in `T` symbol (= `TRUE`); pass positionally or use `temp = ...` explicitly.
`T_ref`	Reference temperature in Kelvin (default 293.15 K = 20 °C).
`T_A`	Arrhenius temperature in Kelvin (default 8000 K).

Details

$$c_T = \exp\left(\frac{T_A}{T_{\text{ref}}} - \frac{T_A}{T}\right)$$

where T and T_{ref} are in Kelvin and T_A is the Arrhenius temperature (a species-specific constant, typically 6000–12000 K for ectotherms; Kooijman, 2010, Table 8.1). At $T = T_{\text{ref}}$, the factor is exactly 1.

Value

Numeric correction factor (dimensionless, > 0).

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press, Eq. 1.2. doi:10.1017/CBO9780511805400

Examples

```
# Correction at 25 C relative to 20 C reference
arrhenius(298.15, T_ref = 293.15, T_A = 8000)  # ~ 1.74

# No correction at reference temperature
arrhenius(293.15)  # exactly 1
```

---

bdeb_data *Prepare Data for BDEB Models*

Description

Converts long-format data frames into the structured list required by the BayesianDEB Stan programs. Growth observations are matched to the DEB state variable $L = V^{1/3}$ (structural length); reproduction records are interval counts of offspring over $[t_{\text{start}}, t_{\text{end}})$. The function validates column names, rejects negative times/lengths, sorts by individual and time, and (for hierarchical models) pads ragged observation vectors into matrices with `NaN` fill, as required by Stan's fixed-size array declarations.

Usage

```
bdeb_data(growth = NULL, reproduction = NULL, concentration = NULL, f_food = 1)
```

Arguments

`growth`	A data frame with columns: `id`, `time` (days), `length` (**structural** length in cm, i.e., $L = V^{1/3}$). If your data are **physical (observed) lengths**, multiply by the shape coefficient: $L = \delta_M \times L_w$. See the package vignette for guidance on δ_M.
`reproduction`	A data frame with columns: `id`, `t_start`, `t_end`, `count` (number of offspring in the interval). For cumulative counts use `repro_to_intervals()` first.
`concentration`	Optional named numeric vector or data frame mapping individual/group id to external toxicant concentration C_w (for DEBtox models; see `bdeb_tox()`).
`f_food`	Scaled functional response $f \in [0, 1]$, the ratio of actual to maximum ingestion rate (Kooijman, 2010, Eq. 2.3). Default 1 (ad libitum feeding).

Value

A `bdeb_data` object (S3 list) ready for `bdeb_model()`.

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press. doi:10.1017/CBO9780511805400

Examples

```
# Simple growth data
df <- data.frame(
  id = rep(1, 10),
  time = seq(0, 45, by = 5),
  length = c(0.1, 0.15, 0.22, 0.30, 0.38, 0.44, 0.49, 0.52, 0.54, 0.55)
)
dat <- bdeb_data(growth = df)
```

`bdeb_derived` *Compute Derived Biological Quantities from the Posterior*

Description

Transforms the raw DEB parameter draws into biologically interpretable quantities, automatically propagating parameter uncertainty. All formulas follow Kooijman (2010, Ch. 3, Table 3.1) strictly.

Usage

```
bdeb_derived(object, ...)

## Default S3 method:
bdeb_derived(object, ...)

## S3 method for class 'bdeb_fit'
bdeb_derived(object, quantities = c("L_inf", "k_M", "growth_rate"), f = 1, ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`...`	Additional arguments passed to methods.
`quantities`	Character vector of quantities to compute. One or more of `"L_m"`, `"L_inf"`, `"k_M"`, `"growth_rate"`, `"g"`.
`f`	Scaled functional response $f \in (0, 1]$ for computing food-dependent quantities (`"L_inf"`, `"growth_rate"`). Default 1 (ad libitum).

Value

A posterior::draws_df with one column per requested quantity and one row per posterior draw.

DEB length terminology

DEB theory distinguishes several length measures. This function returns **structural length** $L = V^{1/3}$, which is the cube root of structural volume. Structural length is *not* the same as **physical (observed) length** L_w, which relates to L via the shape coefficient: $L_w = L/\delta_M$. The shape coefficient δ_M is species-specific (typically 0.1–0.5) and is not estimated by this package. If your data are physical lengths, you should either (a) convert to structural length before fitting, or (b) divide `L_inf` by your species' δ_M to obtain the physical asymptotic length.

Available quantities

`"L_m"` Maximum structural length at $f = 1$ (Kooijman, 2010, Table 3.1):

$$L_m = \kappa \{p_{Am}\}/[p_M]$$

This is a **species-level constant** independent of food. Units: cm.

`"L_inf"` Ultimate structural length at food level f (Kooijman, 2010, Eq. 3.4):

$$L_i = f \cdot L_m = f\,\kappa \{p_{Am}\}/[p_M]$$

The asymptotic length when $dV/dt = 0$ at constant food. Depends on f. Units: cm. Dimensional check: $(-)(-)(\text{J d}^{-1}\text{cm}^{-2})/(\text{J d}^{-1}\text{cm}^{-3}) = \text{cm}$.

`"k_M"` Somatic maintenance rate constant:

$$k_M = [p_M]/[E_G]$$

Units: d^{-1}.

`"growth_rate"` Von Bertalanffy growth rate (Kooijman, 2010, Eq. 3.23):

$$\dot{r}_B = \frac{k_M\, g}{3\,(f + g)}$$

where $g = [E_G]\, v/(\kappa \{p_{Am}\})$ is the energy investment ratio. Depends on f. Units: d^{-1}.

`"g"` Energy investment ratio (Kooijman, 2010, Table 3.1):

$$g = [E_G]\, v/(\kappa \{p_{Am}\})$$

Dimensionless. Large g means growth is expensive relative to reserve turnover.

Reference example

For *Eisenia fetida* with AmP parameters $\{p_{Am}\} = 5.0$, $[p_M] = 0.5$, $\kappa = 0.75$: $L_m = 0.75 \times 5.0/0.5 = 7.5$ cm (structural). With $\delta_M \approx 0.24$, the physical maximum length would be $L_m/\delta_M \approx 31$ mm, consistent with observations.

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press. doi:10.1017/CBO9780511805400

Marques, G.M., Augustine, S., Lika, K., Pecquerie, L., Domingos, T. and Kooijman, S.A.L.M. (2018). The AmP project: comparing species on the basis of dynamic energy budget parameters. *PLOS Computational Biology*, 14(5), e1006100. doi:10.1371/journal.pcbi.1006100

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  bdeb_derived(fit, quantities = c("L_inf", "k_M"))
}
```

---

bdeb_diagnose *MCMC Convergence Diagnostics*

Description

Reports a comprehensive set of NUTS/HMC diagnostics following the recommendations of Vehtari et al. (2021):

Divergent transitions Indicate that the numerical leapfrog integrator encountered regions of high curvature. Even a single divergence can bias the posterior. Remedy: increase `adapt_delta`.

Treedepth saturation The NUTS trajectory hit the maximum allowed tree depth, meaning it could not find a U-turn. Remedy: increase `max_treedepth`.

E-BFMI Energy Bayesian Fraction of Missing Information. Values below 0.3 indicate that the momentum resampling is inefficient (Betancourt, 2016).

$\hat{R}$ Split-$\hat{R}$ convergence diagnostic. Values > 1.01 indicate incomplete mixing across chains.

Bulk and tail ESS Effective sample size for the bulk and tails of the posterior. Values below 400 suggest that posterior summaries may be unreliable.

Usage

```
bdeb_diagnose(fit, pars = NULL)
```

Arguments

fit	A bdeb_fit() object.
pars	Character vector of parameter names to report. Default: all model parameters (excluding generated quantities such as log_lik, L_rep, and lp__).

Details

Returns a bdeb_diagnostics S3 object. The object has dedicated print(), summary() and plot() methods. When called interactively the print method is invoked automatically; assign the result (d <- bdeb_diagnose(fit)) to suppress output.

Value

An object of class bdeb_diagnostics with components n_divergent, n_max_treedepth, ebfmi, summary (a posterior::summarise_draws() tibble), pars, and model_type.

References

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B. and Bürkner, P.-C. (2021). Rank-normalization, folding, and localization: an improved $\hat{R}$ for assessing convergence of MCMC. *Bayesian Analysis*, 16(2), 667–718. doi:10.1214/20BA1221

Betancourt, M. (2016). Diagnosing biased inference with divergences. Stan case study. https://mc-stan.org/users/documentation/case-studies/divergences_and_bias.html

See Also

print.bdeb_diagnostics(), summary.bdeb_diagnostics(), plot.bdeb_diagnostics()

Examples

```
# Requires the CmdStan toolchain (Suggests: cmdstanr); gated on its
# availability and wrapped in \donttest{} so example("bdeb_diagnose")
# runs it when a toolchain is present, mirroring bdeb_fit().

if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(),
                    error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  d <- bdeb_diagnose(fit)
  print(d)
  summary(d)
  plot(d, type = "rhat")
}
```

`bdeb_ec50` *Extract EC50 and NEC from a DEBtox Fit*

Description

Extracts the full posterior distribution of the EC_{50} and the NEC (no-effect concentration, z_w) from a fitted DEBtox model. Both quantities are computed analytically in the Stan `generated quantities` block, avoiding the need for post-hoc root finding. At toxicokinetic steady state the stress factor equals $s = b_w(C_w - z_w)$ for $C_w > z_w$, so setting $s = 0.5$ yields

Usage

```
bdeb_ec50(fit, prob = 0.9, verbose = TRUE)
```

Arguments

`fit`	A `bdeb_fit()` object from a DEBtox model.
`prob`	Credible interval probability. Default 0.90.
`verbose`	Logical; if `TRUE` (default) the summary table is printed via `cli::cli_verbatim()` / `message()` and can be silenced with `suppressMessages()`. Set to `FALSE` for a silent run; the invisible return value is identical.

Details

$$EC_{50} = z_w + \frac{0.5}{b_w}.$$

The NEC is the threshold concentration below which no effect occurs; it corresponds directly to the parameter z_w in the damage model of Kooijman & Bedaux (1996).

Value

A named list with:

- `draws`: posterior draws of EC50
- `summary`: mean, median, sd, lower, upper
- `NEC`: posterior summary of the no-effect concentration

References

Kooijman, S.A.L.M. and Bedaux, J.J.M. (1996). *The Analysis of Aquatic Toxicity Data*. VU University Press, Amsterdam.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(debtox_growth)
  dat <- bdeb_data(growth = debtox_growth,
                   concentration = c("1" = 0, "11" = 20,
                                     "21" = 80, "31" = 200))
  fit <- bdeb_fit(bdeb_tox(dat, stress = "assimilation"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  init = 0.5, refresh = 0)
  bdeb_ec50(fit, prob = 0.95)
}
```

---

bdeb_fit *Fit a BDEB Model via Hamiltonian Monte Carlo*

Description

Compiles the bundled Stan program via **cmdstanr** (which handles caching internally based on the Stan source hash) and runs the No-U-Turn Sampler (NUTS; Hoffman & Gelman, 2014). The Stan ODE system is solved at each leapfrog step using the BDF stiff solver (`ode_bdf`) with absolute and relative tolerances of 10^{-6}.

Usage

```
bdeb_fit(
  model,
  chains = 4,
  iter_warmup = 1000,
  iter_sampling = 1000,
  adapt_delta = 0.8,
  max_treedepth = 10,
  seed = NULL,
  parallel_chains = NULL,
  threads_per_chain = NULL,
  refresh = 200,
  algorithm = c("sampling", "variational"),
  ...
)
```

Arguments

`model`	A `bdeb_model()` object.
`chains`	Number of independent MCMC chains. Default 4 (the minimum recommended by Vehtari et al., 2021, for reliable $\hat{R}$).

`iter_warmup` Number of warmup (adaptation) iterations per chain. Default 1000. Stan uses dual-averaging to tune the step size and diagonal mass matrix during warmup.

`iter_sampling` Number of post-warmup sampling iterations per chain. Default 1000 (yielding 4000 total draws with 4 chains).

`adapt_delta` Target Metropolis acceptance probability for NUTS. Default 0.8. Increase toward 1.0 to reduce divergences at the cost of smaller step sizes and longer runtime.

`max_treedepth` Maximum binary-tree depth for NUTS. Default 10 (i.e., up to $2^{10} = 1024$ leapfrog steps per transition).

`seed` Integer random seed for full reproducibility.

`parallel_chains` Number of chains to run in parallel. Default `min(chains, detectCores() - 1)`.

`threads_per_chain` Number of threads per chain for within-chain parallelism via Stan's `reduce_sum`. Default `NULL` (no threading; the model runs sequentially within each chain). Set to e.g. 4 for the `"hierarchical"` or `"debtox"` models to distribute per-individual / per-group ODE solves across threads. When threading is active, adjust `parallel_chains` so that `chains * threads_per_chain` does not exceed available cores. Has no effect on `"individual"` or `"growth_repro"` models (single ODE solve, nothing to distribute).

`refresh` How often to print sampling progress (iterations). Default 200. Set to 0 for silent operation.

`algorithm` One of `"sampling"` (default; full HMC via NUTS) or `"variational"` (mean-field automatic differentiation variational inference, ADVI). ADVI is **substantially faster** but yields an *approximation* of the posterior, not an exact draw. Use it for illustrative or exploratory work; **always use the default `"sampling"` for publication-grade inference**. When `algorithm = "variational"`, the arguments `chains`, `iter_warmup`, `iter_sampling`, `adapt_delta`, `max_treedepth`, and `parallel_chains` are ignored; ADVI-specific defaults are used instead (1000 output samples). Diagnostics that depend on chain mixing (R-hat, divergences, treedepth) are not defined for ADVI fits, and `bdeb_diagnose()` will refuse to run on them.

`...` Additional arguments forwarded to `CmdStanModel$sample()` (or `CmdStanModel$variational()` when `algorithm = "variational"`).

Details

Tuning guidance. If `bdeb_diagnose()` reports divergent transitions, increase `adapt_delta` toward 1.0 (e.g., 0.95 or 0.99). This reduces the step size, trading speed for geometric fidelity of the sampler. If the maximum treedepth is frequently saturated, increase `max_treedepth` (e.g., 12 or 15). For hierarchical models, starting values can matter; the non-centred parameterisation used in `bdeb_hierarchical_growth.stan` should suffice in most cases.

Value

A `bdeb_fit` object containing the `CmdStanMCMC` (or `CmdStanVB`) result, the model specification, and sampling metadata. The list element `algorithm` records which inference engine was used.

Notes on Stan informational messages

During warmup it is normal for Stan to emit informational messages such as "Informational Message: The current Metropolis proposal is about to be rejected because of the following issue(s)" or occasional ODE solver warnings. These are emitted whenever the leapfrog integrator probes a region of parameter space where the DEB ODE is stiff or numerically extreme; the proposal is then rejected and the chain continues. As of version 0.2.0 the bundled Stan models call `ode_bdf_tol` with `max_num_steps = 1e5` (raised from `1e4` in 0.1.x) which substantially reduces these messages but does not eliminate them in pathological priors or with very few warmup iterations. These messages are **benign** as long as `bdeb_diagnose()` reports no divergent transitions, no max treedepth saturation, and adequate $\hat{R}$/ESS for all parameters.

References

Hoffman, M.D. and Gelman, A. (2014). The No-U-Turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47), 1593–1623.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B. and Bürkner, P.-C. (2021). Rank-normalization, folding, and localization: an improved $\hat{R}$ for assessing convergence of MCMC. *Bayesian Analysis*, 16(2), 667–718. doi:10.1214/20BA1221

Examples

```
# bdeb_fit() requires the external CmdStan toolchain (Suggests:
# cmdstanr) and a fresh fit takes > 30 seconds (Stan compilation
# plus MCMC).  The example is wrapped in \donttest{} and gated on
# cmdstanr availability so that R CMD check skips it on CRAN's
# toolchain-free workers but power users can run it after
# cmdstanr::install_cmdstan().

if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(),
                    error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  mod <- bdeb_model(dat, type = "individual")
  fit <- bdeb_fit(mod, chains = 1, iter_warmup = 200,
                  iter_sampling = 200, refresh = 0)
  print(fit)
}
```

---

bdeb_loo *LOO Cross-Validation for Model Comparison*

Description

Computes approximate leave-one-out cross-validation (LOO-CV) using Pareto-smoothed importance sampling (PSIS; Vehtari et al., 2017). Requires that the Stan model includes a `log_lik` array in the `generated quantities` block.

Usage

```
bdeb_loo(fit, endpoint = c("all", "growth", "reproduction"), ...)
```

Arguments

fit	A bdeb_fit() object.
endpoint	Which log-likelihood to use for "growth_repro" models: "all" (default, concatenates growth + reproduction), "growth", or "reproduction". Ignored for "individual" models.
...	Additional arguments passed to loo::loo().

Details

Currently supported for "individual" and "growth_repro" models only. The "hierarchical" and "debtox" models do not store per-observation log_lik in generated quantities: "hierarchical" computes the likelihood inside reduce_sum (no access to individual contributions outside the function); "debtox" uses the same reduce_sum approach and its generated quantities block only computes EC50/NEC. Adding per-group log_lik to DEBtox is planned for a future version. An informative error is raised for unsupported types.

Value

A loo object (see loo::loo()).

Conditional independence assumption

For "growth_repro" models with endpoint = "all", growth and reproduction observations are concatenated and each treated as an independent data point. This is valid because the two endpoints are **conditionally independent given the latent DEB process** (growth observations depend only on $V(t)$, reproduction counts depend only on ΔE_R; given the ODE solution, they share no additional error). Use endpoint = "growth" or endpoint = "reproduction" to compute LOO for a single endpoint.

References

Vehtari, A., Gelman, A. and Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5), 1413–1432. doi:10.1007/s1122201696964

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  bdeb_loo(fit)
```

```
}
```

`bdeb_model` *Specify a BDEB Model*

Description

Creates a model specification that binds together the prepared data, the DEB process model (encoded as a pre-written Stan program), the prior distributions, and the observation model. Internally it calls the appropriate `build_stan_data_*()` function to assemble the list of named values that Stan expects. The Stan program is *not* compiled at this stage — compilation and sampling happen in `bdeb_fit()`.

Usage

```
bdeb_model(
  data,
  type = c("individual", "growth_repro", "hierarchical", "debtox"),
  priors = list(),
  observation = list(),
  temperature = NULL
)
```

Arguments

`data`	A `bdeb_data()` object.
`type`	Model type: `"individual"`, `"growth_repro"`, `"hierarchical"`, or `"debtox"`.
`priors`	Named list of `bdeb_prior` objects (see `prior_lognormal()`, `prior_beta()`, etc.). Entries not supplied are filled from `prior_default()`.
`observation`	Named list of observation model specs for each endpoint. Default: `list(growth = obs_normal(), reproduction = obs_negbinom())`.
`temperature`	Optional list with components `T_obs` (observation temperature in Kelvin), `T_ref` (reference temperature in K), and `T_A` (Arrhenius temperature in K). If provided, the correction factor $c_T = \exp(T_A/T_{\text{ref}} - T_A/T_{\text{obs}})$ (see `arrhenius()`) is applied **inside the Stan ODE** to all rate parameters: $\{p_{Am}\}$, $[p_M]$, v, k_J (growth_repro), k_d (debtox). Parameters that are **not** rates (κ, $[E_G]$, z_w, b_w) are not corrected. The posterior gives estimates at T_{ref}, making results comparable across experiments at different temperatures. The correction is global (single temperature for the entire experiment). If `NULL` (default), no correction is applied ($c_T = 1$). The field was renamed from `T` to `T_obs` in version 0.1.4 to avoid shadowing R's built-in `T` symbol.

Details

The four model types correspond to increasingly complex DEB formulations:

`"individual"` Standard 2-state DEB (reserve E, structure V), Kooijman (2010, Ch. 2). The ODE is solved with Stan's `ode_bdf` (stiff BDF solver) at tolerances 10^{-6}.

`"growth_repro"` 3-state model adding the reproduction buffer E_R. Offspring counts are modelled as $R_i \sim \text{NegBin}(k_R \Delta E_R, \phi)$, where $\Delta E_R = E_R(t_{\text{end}}) - E_R(t_{\text{start}})$.

`"hierarchical"` 2-state model with a lognormal random effect on $\{p_{Am}\}$: $\log p_{Am,j} = \mu + \sigma z_j$, $z_j \sim N(0,1)$ (non-centred). Shared parameters $[p_M], \kappa, v, [E_G]$ are estimated from all individuals jointly (partial pooling; Gelman & Hill, 2006). Initial structural length $L_{0,j}$ varies per individual; initial reserve density E_0 is shared (assumes organisms start with the same reserve state, which is typical for lab-reared cohorts).

`"debtox"` 4-state TKTD model following Jager et al. (2006). Adds scaled damage D_w with $dD_w/dt = k_d(\max(C_w - z_w, 0) - D_w)$. The EC_{50} is computed analytically as $z_w + 0.5/b_w$.

Value

A `bdeb_model` object (S3 list).

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press. doi:10.1017/CBO9780511805400

Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Jager, T., Heugens, E.H.W. and Kooijman, S.A.L.M. (2006). Making sense of ecotoxicological test results: towards application of process-based models. *Ecotoxicology*, 15(3), 305–314. doi:10.1007/s106460060060x

Examples

```
df <- data.frame(id = 1, time = 0:5, length = seq(0.1, 0.6, by = 0.1))
dat <- bdeb_data(growth = df)
mod <- bdeb_model(dat, type = "individual")
print(mod)
summary(mod)
```

`bdeb_plots` *Plot Methods for BDEB Objects*

Description

Visualisation methods for BDEB fits, posterior predictive checks, and derived quantities using ggplot2. All plots return ggplot2 objects that can be further customised.

bdeb_ppc *Posterior Predictive Checks for BDEB Models*

Description

Extracts replicated data y^{rep} generated in the Stan `generated quantities` block and pairs them with the observed data y. Following Gelman et al. (2013, Ch. 6), posterior predictive checks (PPCs) answer: "could the observed data plausibly have been generated by the fitted model?" If the replicated data envelopes consistently miss systematic features of y, the model is mis-specified.

Usage

```
bdeb_ppc(fit, type = c("growth", "reproduction", "all"))
```

Arguments

`fit`	A `bdeb_fit()` object.
`type`	Which endpoint to check: `"growth"`, `"reproduction"`, or `"all"`. Default `"growth"`.

Details

For growth endpoints, the replicated lengths L_i^{rep} are drawn as $L_i^{\text{rep}} \sim N(\hat{L}_i, \sigma_L)$ in the Stan program. For reproduction, $R_i^{\text{rep}} \sim \text{NegBin}(k_R \Delta E_R, \phi)$.

Value

A `bdeb_ppc` object containing observed data (`L_obs` or `R_obs`), a matrix of replicated data (`L_rep` or `R_rep`), and metadata for `plot.bdeb_ppc()`.

Supported models

`"individual"` Full PPC with `L_rep` and `L_obs`.

`"growth_repro"` Growth PPC via `L_rep`; reproduction PPC via `R_rep`.

`"hierarchical"` Not available. The Stan model uses `reduce_sum` and does not generate `L_rep`. Use `bdeb_diagnose()` and trace plots for model checking.

`"debtox"` Not available for the same reason. Use `plot_dose_response()` for visual model checking.

References

Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A. and Rubin, D.B. (2013). *Bayesian Data Analysis*. 3rd edition. Chapman & Hall/CRC. (Chapter 6: Model checking.)

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  ppc <- bdeb_ppc(fit, type = "growth")
  plot(ppc)
}
```

---

bdeb_prior-methods *Methods for* bdeb_prior *Objects*

Description

Display, summarise, and plot a bdeb_prior object returned by prior_lognormal(), prior_normal(), prior_beta(), prior_halfnormal(), prior_halfcauchy(), or prior_exponential().

Usage

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

## S3 method for class 'bdeb_prior'
summary(object, ...)

## S3 method for class 'summary.bdeb_prior'
print(x, ...)

## S3 method for class 'bdeb_prior'
plot(x, n = 200, ...)
```

Arguments

x, object	A bdeb_prior object.
...	Ignored.
n	Number of grid points for the density curve. Default 200.

Value

print() returns the object invisibly. summary() returns a list with theoretical mean, sd, and the 5th, 50th, and 95th quantiles where defined for the family. plot() returns a ggplot2::ggplot object showing the prior density between the 0.001 and 0.999 quantiles.

Examples

```
p <- prior_lognormal(mu = 1.5, sigma = 0.5)
print(p)
summary(p)
plot(p)
```

`bdeb_prior_predictive` *Prior Predictive Check*

Description

Simulates growth trajectories by sampling parameters from the prior distributions and forward-integrating the DEB ODE in R (Euler method). This is the prior analogue of `bdeb_ppc()` and answers: "what range of data does the model predict *before seeing any data*?" Following Gabry et al. (2019), if the prior predictive distribution covers biologically implausible values, the priors should be tightened.

Usage

```
bdeb_prior_predictive(model, n_draws = 500, dt = 0.5, seed = NULL)
```

Arguments

`model`	A `bdeb_model()` object.
`n_draws`	Number of prior draws. Default 500.
`dt`	Euler step size (days). Default 0.5.
`seed`	Integer seed for reproducibility. Default `NULL`.

Details

Does **not** require CmdStan — all sampling and simulation is done in R using Euler integration (step size `dt`). This is an **approximate visualisation tool**, not exact Stan inference. The ODE trajectories may differ slightly from the BDF solver used during fitting.

Value

A list with class `"bdeb_prior_predictive"` containing:

`t` Time vector.

`L` Matrix of simulated trajectories (draws x time points).

`L_inf` Vector of prior-predictive ultimate lengths.

`priors` Named list of prior objects used.

References

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M. and Gelman, A. (2019). Visualization in Bayesian workflow. *Journal of the Royal Statistical Society: Series A*, 182(2), 389–402. doi:10.1111/rssa.12378

Examples

```
dat <- bdeb_data(growth = data.frame(id = 1, time = 0:10,
  length = seq(0.1, 0.5, length.out = 11)))
mod <- bdeb_model(dat, type = "individual")
pp <- bdeb_prior_predictive(mod, n_draws = 100, seed = 1)
plot(pp, n_draws = 30)
```

`bdeb_session_info` *Reproducibility Report*

Description

Prints a concise summary of the software environment used for a BayesianDEB analysis. Useful for supplementary materials in publications or for diagnosing cross-machine discrepancies.

Usage

```
bdeb_session_info(fit = NULL)
```

Arguments

`fit`	Optional `bdeb_fit()` object. If provided, includes model-specific details (type, chains, iterations, adapt_delta, seed).

Value

Invisibly returns a named list with all reported information.

Examples

```
bdeb_session_info()
```

bdeb_tox *DEBtox Model Specification*

Description

Convenience wrapper for `bdeb_model()` that sets `type = "debtox"` and provides a `stress` argument to select the physiological mode of action (PMoA) of the toxicant. The underlying TKTD framework follows Jager et al. (2006) and the GUTS-RED-SD simplification of Jager & Zimmer (2012):

Usage

```
bdeb_tox(
  data,
  stress = c("assimilation", "maintenance", "growth_cost"),
  priors = list(),
  ...
)
```

Arguments

`data`	A `bdeb_data()` object with `concentration` specified.
`stress`	Physiological mode of action. Currently only `"assimilation"` is implemented.
`priors`	Named list of priors. Missing entries filled from `prior_default("debtox")`. The toxicological parameters k_d, z_w, b_w default to weakly informative log-normal priors.
`...`	Additional arguments passed to `bdeb_model()`.

Details

Toxicokinetics. Scaled internal damage D_w tracks the external concentration with first-order kinetics:

$$\frac{dD_w}{dt} = k_d\big(\max(C_w - z_w,\, 0) - D_w\big)$$

where k_d is the dominant rate constant, z_w is the NEC (no-effect concentration), and C_w is the external concentration.

Stress on assimilation. The assimilation flux is reduced by a factor $\max(1 - b_w D_w, 0)$, where b_w is the effect intensity. At steady state ($D_w = C_w - z_w$), the EC_{50} for 50\ $z_w + 0.5/b_w$.

Value

A `bdeb_model` object of type `"debtox"`.

Note

Lifecycle: beta. Stress on assimilation is implemented and tested; `"maintenance"` and `"growth_cost"` modes are planned.

Important limitation: the DEBtox model fits **one ODE per concentration group**, not per individual. If your data contain multiple individuals per concentration, they are automatically aggregated to group means per time point (with a warning). This is appropriate for group-level summary data but does not capture within-group individual variation. A hierarchical DEBtox extension is planned for a future version.

References

Jager, T., Heugens, E.H.W. and Kooijman, S.A.L.M. (2006). Making sense of ecotoxicological test results: towards application of process-based models. *Ecotoxicology*, 15(3), 305–314. doi:10.1007/s106460060060x

Jager, T. and Zimmer, E.I. (2012). Simplified Dynamic Energy Budget model for analysing ecotoxicity data. *Ecological Modelling*, 225, 74–81. doi:10.1016/j.ecolmodel.2011.11.012

Examples

```
# R-side specification only (no Stan sampling)
data(debtox_growth)
# one replicate per concentration avoids aggregation warning
dt <- debtox_growth[debtox_growth$id %in% c(1, 11, 21, 31), ]
dat <- bdeb_data(growth = dt, concentration = c(0, 20, 80, 200))
mod <- bdeb_tox(dat, stress = "assimilation")
```

`coef.bdeb_fit` *Extract Point Estimates from a BDEB Fit*

Description

Returns posterior medians of all model parameters as a named numeric vector, consistent with the S3 `coef()` convention.

Usage

```
## S3 method for class 'bdeb_fit'
coef(object, type = c("median", "mean"), ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`type`	One of `"median"` (default) or `"mean"`.
`...`	Ignored.

Value

Named numeric vector of point estimates.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  coef(fit)
}
```

`confint.bdeb_fit` *Posterior Credible Intervals for BDEB Model Parameters*

Description

Bayesian counterpart to `stats::confint()`: returns the posterior central credible interval for each model parameter.

Usage

```
## S3 method for class 'bdeb_fit'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`parm`	Character vector of parameter names. Default: all model parameters.
`level`	Probability mass of the credible interval. Default 0.95.
`...`	Ignored.

Value

A matrix with one row per parameter and two columns named by their percentile (e.g. `"2.5%"`, `"97.5%"` for `level = 0.95`).

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  confint(fit, level = 0.90)
}
```

`deb_fluxes` *Compute DEB Energy Fluxes*

Description

Given current state (E, V) and the core DEB parameters, computes all standard energy fluxes defined by the κ-rule (Kooijman, 2010, Eqs. 2.3–2.12):

Usage

```
deb_fluxes(E, V, f, p_Am, p_M, kappa, v, E_G, k_J = 0, E_Hp = 0)
```

Arguments

`E`	Reserve energy (J).
`V`	Structural volume (cm^3).
`f`	Scaled functional response $f \in [0, 1]$.
`p_Am`	Surface-area-specific maximum assimilation rate $\{p_{Am}\}$ (J d^{-1} cm^{-2}).
`p_M`	Volume-specific somatic maintenance rate $[p_M]$ (J d^{-1} cm^{-3}).
`kappa`	Allocation fraction to soma $\kappa \in (0, 1)$.
`v`	Energy conductance (cm d^{-1}).
`E_G`	Specific cost of structure $[E_G]$ (J cm^{-3}).
`k_J`	Maturity maintenance rate coefficient k_J (d^{-1}). Default 0.
`E_Hp`	Maturity at puberty E_H^p (J). Default 0.

Details

$\dot{p}_A$ Assimilation: $f\{p_{Am}\}L^2$.

$\dot{p}_C$ Mobilisation: $EvL/(E + [E_G]V)$.

$\dot{p}_M$ Somatic maintenance: $[p_M]V$.

$\dot{p}_G$ Growth: $\max(\kappa\dot{p}_C - \dot{p}_M, 0)$.

$\dot{p}_J$ Maturity maintenance: $k_J E_H^p$.

$\dot{p}_R$ Reproduction: $\max((1-\kappa)\dot{p}_C - \dot{p}_J, 0)$.

Value

Named list with fluxes p_A, p_C, p_M, p_G, p_J, p_R, structural length L ($V^{1/3}$), and scaled reserve density e ($E/([E_m]V + 10^{-12})$). The 10^{-12} stabilisation prevents division by zero when $V = 0$; in that edge case e returns a small finite number rather than Inf or NA.

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press, Ch. 2. doi:10.1017/CBO9780511805400

Examples

```
# Energy fluxes for a 1 cm^3 organism with typical Eisenia parameters
deb_fluxes(E = 1, V = 1, f = 1, p_Am = 5, p_M = 0.5,
  kappa = 0.75, v = 0.2, E_G = 4400)
```

---

deb_simulate *Simulate DEB Growth Trajectory*

Description

Forward-integrates the standard 2-state DEB ODE (reserve E, structure V) using Euler's method. This is a standalone simulator independent of Stan — useful for exploring parameter space, generating synthetic data, teaching, and prior predictive checks.

Usage

```
deb_simulate(t_max, p_Am, p_M, kappa, v, E_G, E0, L0, f = 1, dt = 0.5)
```

Arguments

t_max	End time (days).
p_Am	Surface-area-specific assimilation rate $\{p_{Am}\}$ (J d^{-1} cm^{-2}).
p_M	Volume-specific somatic maintenance $[p_M]$ (J d^{-1} cm^{-3}).
kappa	Allocation fraction to soma.
v	Energy conductance (cm d^{-1}).
E_G	Specific cost of structure (J cm^{-3}).
E0	Initial reserve density (J cm^{-3}).
L0	Initial structural length (cm).
f	Scaled functional response $f \in [0, 1]$. Default 1.
dt	Integration step size (days). Default 0.5.

Value

Data frame with columns time, E (reserve), V (volume), L (structural length).

Numerical note

This uses the LSODA solver from **deSolve**, which automatically switches between stiff (BDF) and non-stiff (Adams) methods. This matches the BDF solver used in the Stan models, ensuring numerical consistency between R-side simulation and Stan-side inference. The `dt` parameter controls output resolution, not integration accuracy (governed by `rtol`/`atol` = 1e-6).

Examples

```
# Simulate E. fetida growth for 84 days
traj <- deb_simulate(t_max = 84, p_Am = 5, p_M = 0.5,
  kappa = 0.75, v = 0.2, E_G = 400, E0 = 1, L0 = 0.1)
plot(traj$time, traj$L, type = "l", xlab = "Days", ylab = "L (cm)")
```

---

debtox_growth *Simulated DEBtox Growth Data*

Description

Simulated growth data under toxicant exposure for 4 concentration groups (0, 20, 80, 200 arbitrary units) with 10 individuals per group measured weekly over 6 weeks. Designed for fitting the DEBtox (TKTD) model in `bdeb_tox()`. The toxicant acts through stress on assimilation: the effective assimilation rate is reduced by a factor $\max(1 - b_w \max(C_w - z_w, 0),\ 0)$ following the DEBtox framework of Jager et al. (2006).

Usage

```
debtox_growth
```

Format

A data frame with 280 rows and 4 columns:

id Individual identifier (integer, 1–40)

concentration External toxicant concentration (arbitrary units)

time Observation time in days (0, 7, 14, ..., 42)

length Structural length in cm (with measurement error)

Source

Simulated with NEC $z_w = 15$, effect intensity $b_w = 0.003$, and base DEB parameters identical to eisenia_growth.

References

Jager, T., Heugens, E.H.W. and Kooijman, S.A.L.M. (2006). Making sense of ecotoxicological test results: towards application of process-based models. *Ecotoxicology*, 15(3), 305–314. doi:10.1007/s106460060060x

Examples

```
data(debtox_growth)
head(debtox_growth)
```

`debtox_simulate` *Simulate DEBtox Growth Under Toxicant Exposure*

Description

Forward-integrates the 4-state DEBtox ODE (E, V, E_R, D_w) with stress on assimilation. Standalone simulator independent of Stan.

Usage

```
debtox_simulate(
  t_max,
  p_Am,
  p_M,
  kappa,
  v,
  E_G,
  E0,
  L0,
  k_d,
  z_w,
  b_w,
  C_w,
  f = 1,
  dt = 0.5
)
```

Arguments

`t_max`	End time (days).
`p_Am`	Surface-area-specific assimilation rate $\{p_{Am}\}$ (J d^{-1} cm^{-2}).
`p_M`	Volume-specific somatic maintenance $[p_M]$ (J d^{-1} cm^{-3}).
`kappa`	Allocation fraction to soma.
`v`	Energy conductance (cm d^{-1}).
`E_G`	Specific cost of structure (J cm^{-3}).
`E0`	Initial reserve density (J cm^{-3}).
`L0`	Initial structural length (cm).
`k_d`	Damage recovery rate (d^{-1}).
`z_w`	No-effect concentration (NEC).
`b_w`	Effect intensity.

C_w	External toxicant concentration.
f	Scaled functional response $f \in [0, 1]$. Default 1.
dt	Integration step size (days). Default 0.5.

Value

Data frame with columns time, E, V, L, R (reproduction buffer), Dw (scaled damage).

Examples

```
traj <- debtox_simulate(t_max = 42, p_Am = 5, p_M = 0.5,
  kappa = 0.75, v = 0.2, E_G = 400, E0 = 1, L0 = 0.1,
  k_d = 0.3, z_w = 15, b_w = 0.003, C_w = 80)
plot(traj$time, traj$L, type = "l")
```

---

eisenia_cd *Eisenia andrei Cadmium Toxicity Data (Van Gestel 1991)*

Description

Real experimental growth data for *Eisenia andrei* exposed to cadmium in natural soil at 23 °C from Van Gestel et al. (1991), obtained via the EGrowth database (Mathieu, 2018). Five concentration groups (0, 10, 32, 100, 320 mg Cd/kg) with 7 time points each over 85 days. Group-mean body mass was converted to structural length via $L = (W/d_V)^{1/3}$ with $d_V = 1.05$ g cm^{-3}.

Usage

```
eisenia_cd
```

Format

A data frame with 35 rows and 4 columns:

id Concentration group identifier (1–5)

time Time in days (0–85)

length Structural length in cm

concentration Cadmium concentration (mg Cd/kg soil)

Source

EGrowth database, curve IDs gr0119–gr0123. https://github.com/JeromeMathieuEcology/EGrowth

References

Van Gestel, C.A.M., Van Dis, W.A., Dirven-Van Breemen, E.M., Sparenburg, P.M. and Baerselman, R. (1991). Influence of cadmium, copper, and pentachlorophenol on growth and sexual development of *Eisenia andrei* (Oligochaeta; Annelida). *Biology and Fertility of Soils*, 12, 117–121. doi:10.1007/BF00341486

Examples

```
data(eisenia_cd)
plot(eisenia_cd$time, eisenia_cd$length,
     col = as.factor(eisenia_cd$concentration),
     xlab = "Days", ylab = "Structural length (cm)")
```

`eisenia_growth` *Simulated Eisenia fetida Growth Data*

Description

Simulated growth dataset for 21 individuals of the earthworm *Eisenia fetida* measured weekly over 12 weeks under controlled laboratory conditions (20 °C, ad libitum feeding), following a standard OECD 222 test protocol design. DEB parameters are based on the *E. fetida* entry in the Add-my-Pet collection (AmP; Marques et al., 2018). Individual variation in $\{p_{Am}\}$ was drawn from a lognormal distribution with CV $\approx$ 10\ Gaussian measurement error $\sigma_L = 0.015$ cm was added to the structural length.

Usage

```
eisenia_growth
```

Format

A data frame with 273 rows and 3 columns:

id Individual identifier (integer, 1–21)

time Observation time in days (0, 7, 14, . . . , 84)

length Structural length in cm (= $V^{1/3}$), with measurement error

Source

Simulated from the standard DEB model (Kooijman, 2010, Eqs. 2.4–2.6) with parameters: $\{p_{Am}\} = 5.0$ J/d/cm^2, $[p_M] = 0.5$ J/d/cm^3, $\kappa = 0.75$, $v = 0.2$ cm/d, $[E_G] = 400$ J/cm^3. Individual variation: $\{p_{Am}\}_j \sim \mathrm{LogNormal}(\log 5.0,\ 0.1)$, $L_{0,j} \sim \mathrm{LogNormal}(\log 0.1,\ 0.05)$. Observation error: $\sigma_L = 0.015$ cm.

References

Kooijman, S.A.L.M. (2010). *Dynamic Energy Budget Theory for Metabolic Organisation*. 3rd edition. Cambridge University Press. doi:10.1017/CBO9780511805400

Marques, G.M., Augustine, S., Lika, K., Pecquerie, L., Domingos, T. and Kooijman, S.A.L.M. (2018). The AmP project: comparing species on the basis of dynamic energy budget parameters. *PLOS Computational Biology*, 14(5), e1006100. doi:10.1371/journal.pcbi.1006100

OECD (2016). Test No. 222: Earthworm Reproduction Test (*Eisenia fetida* / *Eisenia andrei*). OECD Guidelines for the Testing of Chemicals.

Examples

```
data(eisenia_growth)
head(eisenia_growth)
```

`eisenia_neuhauser` *Eisenia fetida Growth Data (Neuhauser 1980)*

Description

Real experimental growth data for *Eisenia fetida* on activated sludge at 25 °C from Neuhauser, Hartenstein & Kaplan (1980), obtained via the EGrowth database (Mathieu, 2018). The dataset comprises 37 group-mean body mass measurements (20 worms per replicate) over 250 days, from hatchling (~8 mg) to adult (~2350 mg). Wet mass was converted to structural length via $L = (W/d_V)^{1/3}$ with tissue density $d_V = 1.05$ g cm^{-3}.

Usage

```
eisenia_neuhauser
```

Format

A data frame with 37 rows and 3 columns:

id Individual/group identifier (always 1)

time Time in days since start (1–250)

length Structural length in cm (0.20–1.31)

Source

EGrowth database, curve ID gr0226. https://github.com/JeromeMathieuEcology/EGrowth

References

Neuhauser, E.F., Hartenstein, R. and Kaplan, D.L. (1980). Growth of the earthworm *Eisenia foetida* in relation to population density and food rationing. *Oikos*, 35, 93–98. doi:10.2307/3544730

Mathieu, J. (2018). EGrowth: a global database on intraspecific body growth variability in earthworm. *Soil Biology and Biochemistry*, 122, 71–80. doi:10.1016/j.soilbio.2018.04.004

Examples

```
data(eisenia_neuhauser)
plot(eisenia_neuhauser$time, eisenia_neuhauser$length,
     xlab = "Days", ylab = "Structural length (cm)")
```

`fitted.bdeb_fit` *Fitted Values from a BDEB Fit*

Description

Posterior point estimate (median or mean) of the latent length $\hat{L}_i$ at each observation. Bayesian counterpart of `stats::fitted()`.

Usage

```
## S3 method for class 'bdeb_fit'
fitted(object, type = c("median", "mean"), ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`type`	One of `"median"` (default) or `"mean"`.
`...`	Ignored.

Details

Currently supported for `"individual"` and `"growth_repro"` models. For `"hierarchical"` and `"debtox"` use `bdeb_predict()` instead.

Value

Named numeric vector of fitted values, one per observation.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  fitted(fit)
}
```

`folsomia_repro` *Simulated Folsomia candida Reproduction Data*

Description

Simulated 28-day reproduction dataset for the springtail *Folsomia candida* exposed to 5 cadmium concentrations (0, 10, 50, 100, 500 mg Cd/kg dry soil) with 6 replicates per concentration. The experimental design follows ISO 11267 (springtail reproduction test). Toxicant effects were simulated using a simple NEC-based model: expected offspring $= 120 \times \max(1 - b_w \max(C - z_w, 0), 0)$, with Poisson sampling around the expected value.

Usage

```
folsomia_repro
```

Format

A data frame with 30 rows and 5 columns:

id Replicate identifier (integer, 1–30)

concentration Cadmium concentration in mg/kg dry soil

t_start Start of observation interval (day 0)

t_end End of observation interval (day 28)

count Number of juveniles produced in the interval

Source

Simulated with NEC $z_w = 30$ mg/kg, effect intensity $b_w = 0.005$ (mg/kg)$^{-1}$, and control offspring count ≈ 120.

References

ISO 11267 (2014). *Soil quality — Inhibition of reproduction of Collembola (Folsomia candida) by soil contaminants*.

Examples

```
data(folsomia_repro)
head(folsomia_repro)
```

`logLik.bdeb_fit` *Log-Likelihood of a BDEB Fit*

Description

Computes the log-pointwise predictive density (lppd):

$$\text{lppd} = \sum_{i=1}^{N} \log \left(\frac{1}{S} \sum_{s=1}^{S} \exp(\log p(y_i \mid \theta^{(s)})) \right),$$

the Bayesian analogue of `stats::logLik()`. This is the natural point summary of model fit reported by `bdeb_loo()` (which adds Pareto-smoothed importance sampling for cross-validation).

Usage

```
## S3 method for class 'bdeb_fit'
logLik(object, ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`...`	Ignored.

Details

Requires the Stan model to store per-observation `log_lik` in `generated quantities`. Currently available for `"individual"` and `"growth_repro"` models; `"hierarchical"` and `"debtox"` compute the likelihood inside `reduce_sum` and do not expose individual terms.

Value

A `logLik` object with attributes `df` (number of model parameters) and `nobs` (number of observations).

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  logLik(fit)
}
```

nobs.bdeb_fit *Number of Observations Used in a BDEB Fit*

Description

Returns the total count of growth (and, where relevant, reproduction) observations that contributed to the likelihood. Matches `stats::nobs()` in spirit.

Usage

```
## S3 method for class 'bdeb_fit'
nobs(object, ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`...`	Ignored.

Value

Integer scalar: total number of observations.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  nobs(fit)
}
```

observation_models *Observation Model Specifications*

Description

Observation Model Specifications

Usage

```
obs_normal()

obs_lognormal()

obs_student_t(nu = 5)

obs_poisson()

obs_negbinom()
```

Arguments

nu	Degrees of freedom. Default 5.

Value

A bdeb_obs object (list with class "bdeb_obs").

Functions

- obs_normal(): Gaussian observation error (default for growth)
- obs_lognormal(): Log-normal observation error (multiplicative)
- obs_student_t(): Student-t observation error (robust to outliers)
- obs_poisson(): Poisson observations (for count data)
- obs_negbinom(): Negative binomial observations (overdispersed counts)

Examples

```
obs_normal()
obs_lognormal()
obs_negbinom()
```

---

plot.bdeb_data *Plot a BDEB Data Object*

Description

Visualises the contents of a bdeb_data() object. Growth data are shown as observed length versus time, with one trace per individual. Reproduction data are shown as interval counts versus interval midpoint. When concentration information is available (DEBtox setups) individuals are coloured by group.

Usage

```
## S3 method for class 'bdeb_data'
plot(x, endpoint = NULL, ...)
```

Arguments

x	A bdeb_data() object.
endpoint	Which endpoint to plot: "growth" or "reproduction". Default NULL, which prefers growth when both are present.
...	Ignored.

Value

A ggplot2::ggplot object.

Examples

```
df <- data.frame(id = 1, time = 0:5, length = seq(0.1, 0.6, by = 0.1))
dat <- bdeb_data(growth = df)
plot(dat)
```

---

plot.bdeb_diagnostics *Plot Convergence Diagnostics*

Description

Visualises the per-parameter R-hat or ESS-bulk values from a bdeb_diagnose() object. A dashed red reference line is drawn at the Vehtari et al. (2021) threshold (R-hat = 1.01, ESS-bulk = 400).

Usage

```
## S3 method for class 'bdeb_diagnostics'
plot(x, type = c("rhat", "ess"), full = FALSE, ...)
```

Arguments

x	A bdeb_diagnostics object.
type	One of "rhat" (default) or "ess".
full	Logical; if TRUE, also plot the latent-state rows that are hidden by default.
...	Unused.

Details

To keep the plot short and readable, the per-time-point latent states (x_sol[i,j], L_hat[i], ...) are hidden by default and only the scalar model parameters are shown, mirroring print.bdeb_diagnostics(); set full = TRUE to plot every monitored quantity.

Value

A ggplot2::ggplot object.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  plot(bdeb_diagnose(fit), type = "ess")
}
```

`plot.bdeb_fit` *Plot a BDEB Fit*

Description

Plot a BDEB Fit

Usage

```
## S3 method for class 'bdeb_fit'
plot(
  x,
  type = c("trace", "posterior", "pairs", "trajectory", "prior_posterior"),
  pars = NULL,
  n_draws = 100,
  seed = NULL,
  ...
)
```

Arguments

`x`	A `bdeb_fit()` object.
`type`	Type of plot. One of: • `"trace"`: MCMC trace plots • `"posterior"`: marginal posterior densities • `"pairs"`: bivariate posterior scatter plots • `"trajectory"`: predicted trajectories with data overlay • `"prior_posterior"`: prior (red) vs posterior (blue) densities
`pars`	Character vector of parameters to plot. Default: core DEB parameters.
`n_draws`	Number of posterior draws for trajectory plots. Default 100.
`seed`	Integer seed for reproducible draw selection in trajectory plots. Default `NULL` (no seed).
`...`	Additional arguments passed to bayesplot functions.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  plot(fit, type = "trace")
  plot(fit, type = "trajectory", n_draws = 50)
}
```

---

plot.bdeb_model *Plot a BDEB Model Specification*

Description

Displays the prior densities for the model parameters as a faceted panel of curves. Each parameter prior is drawn between the 0.001 and 0.999 quantiles of its prior distribution; for bounded supports (Beta, half-normal, half-Cauchy, exponential) the lower bound is fixed at zero (or a small epsilon for densities that diverge there). Useful for sanity-checking prior choices before calling `bdeb_fit()`.

Usage

```
## S3 method for class 'bdeb_model'
plot(x, pars = NULL, n = 200, ...)
```

Arguments

`x`	A `bdeb_model()` object.
`pars`	Optional character vector of parameter names to plot. Default: all priors in the model.
`n`	Number of grid points per density curve. Default 200.
`...`	Ignored.

Value

A ggplot2::ggplot object.

Examples

```
df <- data.frame(id = 1, time = 0:5, length = seq(0.1, 0.6, by = 0.1))
dat <- bdeb_data(growth = df)
mod <- bdeb_model(dat, type = "individual")
plot(mod, pars = c("p_Am", "kappa"))
```

`plot.bdeb_ppc` *Plot Posterior Predictive Checks*

Description

Plot Posterior Predictive Checks

Usage

```
## S3 method for class 'bdeb_ppc'
plot(x, n_draws = 50, ...)
```

Arguments

`x`	A `bdeb_ppc()` object.
`n_draws`	Number of replicated trajectories to show. Default 50.
`...`	Ignored.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  plot(bdeb_ppc(fit, type = "growth"))
}
```

plot.bdeb_prediction *Plot Posterior Predictive Trajectories*

Description

Overlays a sample of the posterior trajectory draws stored in a bdeb_prediction object.

Usage

```
## S3 method for class 'bdeb_prediction'
plot(x, n_draws = 100, ...)
```

Arguments

x	A bdeb_prediction object.
n_draws	Maximum number of trajectories to overlay. Default 100.
...	Unused.

Value

A ggplot2::ggplot object.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  plot(predict(fit), n_draws = 50)
}
```

plot_dose_response *Plot DEBtox Dose-Response*

Description

Produces a dose-response curve by **forward-simulating** the full 4-state DEBtox ODE from the posterior in R (Euler integration). For each posterior draw, the ODE is solved at every concentration in a fine grid from 0 to $1.2 \times \max(C_w)$. The y-axis shows the predicted final structural length relative to the control (C = 0) at the same draw, so that each curve propagates the full parameter uncertainty through the dynamic model.

Usage

```
plot_dose_response(
  fit,
  endpoint = "growth",
  n_draws = 100,
  n_conc = 50,
  dt = 1,
  t_end = NULL,
  seed = NULL
)
```

Arguments

fit	A bdeb_fit() object from a DEBtox model.
endpoint	Which endpoint to plot. Currently only "growth".
n_draws	Number of posterior draws to use. Default 100.
n_conc	Number of concentration points in the continuous grid. Default 50.
dt	Euler integration step size (days). Default 1.0. Smaller values are more accurate but slower. The Stan model uses BDF with adaptive stepping; this is an approximation for visualisation.
t_end	End time for the simulation (days). Default NULL (uses the last observation time from the fitted data).
seed	Integer seed for reproducible draw selection. Default NULL.

Details

This is a **visualisation tool**, not exact Stan inference. The R-side Euler integrator (step size dt) is an approximation of the BDF solver used during fitting. For quantitative results, use bdeb_ec50() which extracts the analytically computed EC_{50} and NEC directly from the Stan posterior.

Performance note. Each draw requires n_conc ODE integrations, so n_draws * n_conc total. With default settings (100 draws, 50 concentrations) this takes a few seconds. Reduce n_draws for faster interactive use.

Value

A ggplot2 object.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(debtox_growth)
  dat <- bdeb_data(growth = debtox_growth,
                   concentration = c("1" = 0, "11" = 20,
                                     "21" = 80, "31" = 200))
  fit <- bdeb_fit(bdeb_tox(dat, stress = "assimilation"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
```

```
                  init = 0.5, refresh = 0)
  plot_dose_response(fit, n_draws = 50)
}
```

`predict.bdeb_fit` *Predict from a BDEB Fit*

Description

When `newdata = NULL`, extracts the fitted trajectories (`L_hat`) from the Stan posterior (exact). When `newdata` is provided, performs **R-side forward simulation** (Euler integration) from the posterior draws under new environmental conditions or extended time horizons. The Euler integrator is an **approximation** — for quantitative results at the original data conditions, prefer `newdata = NULL`.

Usage

```
## S3 method for class 'bdeb_fit'
predict(object, newdata = NULL, n_draws = 200, dt = 0.5, seed = NULL, ...)

bdeb_predict(fit, newdata = NULL, n_draws = 200, dt = 0.5, seed = NULL)
```

Arguments

`object`	A `bdeb_fit()` object.
`newdata`	Optional named list with new conditions. Supported fields: `t_predict` Numeric vector of prediction time points (days). Required when `newdata` is not `NULL`. `f_food` Scaled functional response (default: value from the fitted model). `concentration` External toxicant concentration (scalar; only for `"debtox"` models; default 0).
`n_draws`	Number of posterior draws to use. Default 200.
`dt`	Euler step size for forward simulation (days). Default 0.5.
`seed`	Integer seed for reproducible draw selection. Default `NULL` (no seed).
`...`	Ignored.
`fit`	A `bdeb_fit()` object.

Value

A `bdeb_prediction` object with components `t` (time vector), `L_hat` (matrix: draws x time points), `n_draws`, `model_type`.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  pred <- predict(fit, n_draws = 200)
  summary(pred); plot(pred)
}
```

---

print.bdeb_data *Print a BDEB Data Object*

Description

Print a BDEB Data Object

Usage

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

Arguments

x	A bdeb_data() object.
...	Ignored.

Value

The input object, invisibly.

---

print.bdeb_diagnostics *Print a BDEB Diagnostics Report*

Description

Default printing for bdeb_diagnose() output. Displays divergence / treedepth / E-BFMI alerts, R-hat and ESS warnings, and a compact parameter summary table. To keep the on-screen output short, the per-time-point latent states (x_sol[i,j], L_hat[i], ...) are hidden by default; the scalar model parameters are always shown. Output uses cli::cli alerts and is therefore silenceable via cli::cli_inform() sinks.

Usage

```
## S3 method for class 'bdeb_diagnostics'
print(x, full = FALSE, ...)
```

Arguments

x	A bdeb_diagnostics object.
full	Logical; if TRUE, also print the latent-state rows that are hidden by default. The complete table is always available via summary(x)$table.
...	Unused.

Value

The input object, invisibly.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  print(bdeb_diagnose(fit))
}
```

---

print.bdeb_fit *Print a BDEB Fit*

Description

Print a BDEB Fit

Usage

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

Arguments

x	A bdeb_fit() object.
...	Ignored.

Value

The input object, invisibly.

print.bdeb_model *Print a BDEB Model Specification*

Description

Print a BDEB Model Specification

Usage

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

Arguments

x	A bdeb_model() object.
...	Ignored.

Value

The input object, invisibly.

print.bdeb_ppc *Print a BDEB Posterior Predictive Check*

Description

Print a BDEB Posterior Predictive Check

Usage

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

Arguments

x	A bdeb_ppc() object.
...	Ignored.

Value

The input object, invisibly.

print.bdeb_prediction *Print a BDEB Prediction*

Description

One-line metadata printer for `predict.bdeb_fit()` output.

Usage

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

Arguments

`x`	A `bdeb_prediction` object.
`...`	Unused.

Value

The input object, invisibly.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  print(predict(fit))
}
```

prior_default *Default Priors for DEB Parameters*

Description

Returns a named list of weakly informative priors for standard DEB parameters. The defaults are calibrated against the parameter ranges observed in the Add-my-Pet (AmP) collection (Marques et al., 2018) for standard ecotoxicological test species. All positive rate parameters use log-normal priors whose 95 \ roughly one order of magnitude around typical values; κ uses $\mathrm{Beta}(2, 2)$ which is symmetric on $(0, 1)$ with prior mean 0.5.

Usage

```
prior_default(type = c("individual", "growth_repro", "hierarchical", "debtox"))
```

Arguments

type	One of "individual", "growth_repro", "hierarchical", "debtox".

Value

Named list of bdeb_prior objects.

References

Marques, G.M., Augustine, S., Lika, K., Pecquerie, L., Domingos, T. and Kooijman, S.A.L.M. (2018). The AmP project: comparing species on the basis of dynamic energy budget parameters. *PLOS Computational Biology*, 14(5), e1006100. doi:10.1371/journal.pcbi.1006100

Examples

```
prior_default("individual")
```

---

prior_species *Species-Specific Priors from the AmP Collection*

Description

Returns priors calibrated to a specific species using parameter estimates from the Add-my-Pet (AmP) collection (Marques et al., 2018). The log-normal priors are centred on the AmP point estimate (log scale) with a moderate spread ($\sigma = 0.3$) that places 95\ value.

Usage

```
prior_species(
  species,
  type = c("individual", "growth_repro", "hierarchical", "debtox")
)
```

Arguments

species	Character string: species name with underscore separator (e.g., "Eisenia_fetida"). Case-insensitive.
type	Model type for filling model-specific defaults.

Details

Currently supported species (more will be added):

`Eisenia_fetida` Compost earthworm; AmP entry `Eisenia_fetida`.

`Eisenia_andrei` Sibling species of *E. fetida*; shares similar DEB parameters.

`Folsomia_candida` Springtail; standard ISO reproduction test species.

`Daphnia_magna` Water flea; classic aquatic ecotox model.

`Lumbricus_rubellus` Field earthworm; common biomonitoring species.

Value

Named list of `bdeb_prior` objects, suitable for the `priors` argument of `bdeb_model()` or `bdeb_tox()`.

References

Marques, G.M., Augustine, S., Lika, K., Pecquerie, L., Domingos, T. and Kooijman, S.A.L.M. (2018). The AmP project: comparing species on the basis of dynamic energy budget parameters. *PLOS Computational Biology*, 14(5), e1006100. doi:10.1371/journal.pcbi.1006100

Examples

```
# Use AmP-calibrated priors for E. fetida
prior_species("Eisenia_fetida")

# Combine with model specification (R-side, no Stan sampling)
data(eisenia_growth)
dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
mod <- bdeb_model(dat, type = "individual",
  priors = prior_species("Eisenia_fetida"))
```

---

priors *Prior Distribution Specifications for BDEB Models*

Description

Constructor functions for prior distribution objects used in `bdeb_model()`. Each returns a lightweight `bdeb_prior` S3 object that encodes the distribution family and its hyperparameters. The prior is evaluated inside the Stan `model` block; hyperparameters are passed as data so that changing priors does not require recompilation.

Usage

```
prior_lognormal(mu = 0, sigma = 1)

prior_normal(mu = 0, sigma = 1)

prior_beta(a = 2, b = 2)

prior_halfnormal(sigma = 1)

prior_halfcauchy(sigma = 1)

prior_exponential(rate = 1)
```

Arguments

mu	Mean.
sigma	Scale of the half-Cauchy.
a	Shape parameter alpha.
b	Shape parameter beta.
rate	Rate parameter (1/mean).

Details

The choice of prior family follows the recommendations for DEB parameters in Hackenberger (2025, Ch. 6) and general guidelines from Gelman et al. (2013, Sec. 2.9):

- Positive rate parameters ($\{p_{Am}\}$, $[p_M]$, v, $[E_G]$) — log-normal, because the log-transform maps the strictly positive domain to the real line.
- Allocation fraction $\kappa \in (0, 1)$ — Beta, the natural conjugate for bounded parameters.
- Observation-error standard deviations — half-normal (or half-Cauchy for heavier tails), which place zero mass on negative values (Gelman, 2006).
- Hierarchical standard deviations — exponential, following the advice of Betancourt & Girolami (2015) for non-centred parameterisations.

Value

A `bdeb_prior` object (list with class `"bdeb_prior"`).

Functions

- `prior_lognormal()`: Log-normal prior (for positive parameters: p_Am, p_M, v, E_G, etc.)
- `prior_normal()`: Normal prior (for unconstrained parameters)
- `prior_beta()`: Beta prior (for (0,1)-constrained parameters: kappa)
- `prior_halfnormal()`: Half-normal prior (for scale parameters: sigma_L, etc.)
- `prior_halfcauchy()`: Half-Cauchy prior (for scale parameters, heavier tails)
- `prior_exponential()`: Exponential prior (for variance components in hierarchical models)

References

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3), 515–534. doi:10.1214/06BA117A

Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A. and Rubin, D.B. (2013). *Bayesian Data Analysis*. 3rd edition. Chapman & Hall/CRC.

Examples

```
# Log-normal prior on assimilation rate: median ~ exp(1.5) ~ 4.5
prior_lognormal(mu = 1.5, sigma = 0.5)

# Beta(2,2) prior on kappa — symmetric, favouring 0.5
prior_beta(a = 2, b = 2)
```

---

repro_to_intervals *Convert Cumulative Reproduction to Intervals*

Description

Many ecotoxicological protocols (e.g., ISO 11267 for *Folsomia candida*, OECD 222 for *Eisenia fetida*) report cumulative offspring counts at successive observation times. The DEB reproduction buffer model, however, requires interval counts $\Delta R = R(t_{\mathrm{end}}) - R(t_{\mathrm{start}})$ so that the negative-binomial likelihood can be applied to each counting period. This function computes the first-difference per individual.

Usage

```
repro_to_intervals(df)
```

Arguments

`df`	Data frame with columns: `id`, `time`, `cumulative`.

Value

Data frame with columns: `id`, `t_start`, `t_end`, `count`.

Examples

```
cumul <- data.frame(
  id = rep(1, 5),
  time = c(0, 7, 14, 21, 28),
  cumulative = c(0, 10, 30, 60, 100)
)
repro_to_intervals(cumul)
```

`residuals.bdeb_fit` *Residuals from a BDEB Fit*

Description

Observed minus fitted length for each observation, using the posterior point estimate from `fitted()`. Bayesian counterpart of `stats::residuals()`.

Usage

```
## S3 method for class 'bdeb_fit'
residuals(object, type = "response", ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`type`	Currently only `"response"` is supported (raw residuals).
`...`	Ignored.

Details

Currently supported for `"individual"` and `"growth_repro"` models.

Value

Named numeric vector of residuals.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  residuals(fit)
}
```

summary.bdeb_data *Summary of a BDEB Data Object*

Description

Returns a compact list of summary statistics describing a `bdeb_data()` object: number of observations, number of unique individuals, time range, presence of growth/reproduction endpoints, the functional response, and (for DEBtox data) the unique concentration levels.

Usage

```
## S3 method for class 'bdeb_data'
summary(object, ...)

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

Arguments

`object`	A `bdeb_data()` object.
`...`	Ignored.
`x`	A `summary.bdeb_data` object.

Value

An object of class `summary.bdeb_data` (a list).

Examples

```
df <- data.frame(id = 1, time = 0:5, length = seq(0.1, 0.6, by = 0.1))
dat <- bdeb_data(growth = df)
summary(dat)
```

summary.bdeb_diagnostics *Compact Summary of a BDEB Diagnostics Report*

Description

Returns counts of problematic parameters (divergences, treedepth saturations, low-EBFMI chains, R-hat > 1.01, ESS-bulk < 400) suitable for a one-line health check or programmatic gating.

Usage

```
## S3 method for class 'bdeb_diagnostics'
summary(object, ...)
```

Arguments

object	A bdeb_diagnostics object.
...	Unused.

Value

An object of class summary.bdeb_diagnostics (a list).

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  summary(bdeb_diagnose(fit))
}
```

---

summary.bdeb_fit *Posterior Summary for a BDEB Fit*

Description

Returns a tidy summary table of posterior draws for model parameters (and optionally derived quantities), analogous to stats::summary.lm() for frequentist fits.

Usage

```
## S3 method for class 'bdeb_fit'
summary(object, pars = NULL, prob = 0.9, ...)
```

Arguments

object	A bdeb_fit() object.
pars	Character vector of parameter names. Default: all model parameters (excludes log_lik, L_hat, L_rep, R_hat, R_rep, lp__, and the internal p_Am_new).
prob	Probability for the central credible interval. Default 0.90 (5th/95th percentiles).
...	Ignored.

Value

A posterior::draws_summary data frame with columns variable, mean, sd, median, two quantile columns named by their percentile (e.g. "5%" / "95%" for prob = 0.90), rhat, ess_bulk, and ess_tail.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  summary(fit, pars = c("p_Am", "kappa"), prob = 0.95)
}
```

---

summary.bdeb_model *Summary of a BDEB Model Specification*

Description

Returns a list summarising the structure of a `bdeb_model()` object: model type, Stan program name, names of model parameters, prior families, observation models, and (when present) the temperature correction settings. Useful for one-glance verification before calling `bdeb_fit()`.

Usage

```
## S3 method for class 'bdeb_model'
summary(object, ...)

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

Arguments

`object`	A `bdeb_model()` object.
`...`	Ignored.
`x`	A `summary.bdeb_model` object.

Value

An object of class `summary.bdeb_model` (a list).

summary.bdeb_prediction *Tabulate Posterior Predictive Quantiles*

Description

Reduces the matrix of posterior trajectory draws stored in a bdeb_prediction object to per-time-point credible intervals.

Usage

```
## S3 method for class 'bdeb_prediction'
summary(object, prob = 0.9, ...)
```

Arguments

object	A bdeb_prediction object.
prob	Credible interval coverage in (0, 1). Default 0.90.
...	Unused.

Value

A data frame with columns time, lower, median, upper. Carries class summary.bdeb_prediction and attribute prob.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  summary(predict(fit), prob = 0.95)
}
```

vcov.bdeb_fit *Posterior Covariance Matrix of BDEB Model Parameters*

Description

Computes the empirical covariance matrix of the posterior draws for all model parameters. Bayesian counterpart of `stats::vcov()`.

Usage

```
## S3 method for class 'bdeb_fit'
vcov(object, ...)
```

Arguments

`object`	A `bdeb_fit()` object.
`...`	Ignored.

Value

A symmetric numeric matrix with model parameters on rows and columns.

Examples

```
if (requireNamespace("cmdstanr", quietly = TRUE) &&
    nzchar(tryCatch(cmdstanr::cmdstan_path(), error = function(e) ""))) {
  data(eisenia_growth)
  dat <- bdeb_data(growth = eisenia_growth[eisenia_growth$id == 1, ])
  fit <- bdeb_fit(bdeb_model(dat, type = "individual"),
                  chains = 2, iter_warmup = 200, iter_sampling = 200,
                  refresh = 0)
  vcov(fit)
}
```

Index

∗ **datasets**
  debtox_growth, 25
  eisenia_cd, 27
  eisenia_growth, 28
  eisenia_neuhauser, 29
  folsomia_repro, 31

arrhenius, 3
arrhenius(), *14*

bdeb_data, 4
bdeb_data(), *14*, *20*, *34, 35*, *42*, *51*
bdeb_derived, 5
bdeb_diagnose, 7
bdeb_diagnose(), *11, 12*, *16*, *35*, *42*
bdeb_ec50, 9
bdeb_ec50(), *40*
bdeb_fit, 10
bdeb_fit(), *6*, *8, 9*, *13, 14*, *16*, *19*, *21, 22*, *30*, *32, 33*, *36, 37*, *40, 41*, *43*, *50*, *52, 53*, *55*
bdeb_loo, 12
bdeb_model, 14
bdeb_model(), *5*, *10*, *18*, *20*, *37*, *44*, *47*, *53*
bdeb_plots, 15
bdeb_ppc, 16
bdeb_ppc(), *18*, *38*, *44*
bdeb_predict *(*predict.bdeb_fit*)*, 41
bdeb_predict(), *30*
bdeb_prior-methods, 17
bdeb_prior_predictive, 18
bdeb_session_info, 19
bdeb_tox, 20
bdeb_tox(), *5*, *25*, *47*

cli::cli, *42*
cli::cli_inform(), *42*
cli::cli_verbatim(), *9*
coef(), *21*
coef.bdeb_fit, 21
confint.bdeb_fit, 22

deb_fluxes, 23
deb_simulate, 24
debtox_growth, 25
debtox_simulate, 26

eisenia_cd, 27
eisenia_growth, *25*, 28
eisenia_neuhauser, 29

fitted(), *50*
fitted.bdeb_fit, 30
folsomia_repro, 31

ggplot2::ggplot, *17*, *35*, *37*, *39*

logLik.bdeb_fit, 32
loo::loo(), *13*

message(), *9*

nobs.bdeb_fit, 33

obs_lognormal *(*observation_models*)*, 33
obs_negbinom *(*observation_models*)*, 33
obs_normal *(*observation_models*)*, 33
obs_poisson *(*observation_models*)*, 33
obs_student_t *(*observation_models*)*, 33
observation_models, 33

plot(), *8*
plot.bdeb_data, 34
plot.bdeb_diagnostics, 35
plot.bdeb_diagnostics(), *8*
plot.bdeb_fit, 36
plot.bdeb_model, 37
plot.bdeb_ppc, 38
plot.bdeb_ppc(), *16*
plot.bdeb_prediction, 39
plot.bdeb_prior *(*bdeb_prior-methods*)*, 17

plot_dose_response, 39
plot_dose_response(), *16*
posterior::draws_df, *6*
posterior::summarise_draws(), *8*
predict.bdeb_fit, 41
predict.bdeb_fit(), *45*
print(), *8*
print.bdeb_data, 42
print.bdeb_diagnostics, 42
print.bdeb_diagnostics(), *8*, *35*
print.bdeb_fit, 43
print.bdeb_model, 44
print.bdeb_ppc, 44
print.bdeb_prediction, 45
print.bdeb_prior *(*bdeb_prior-methods*)*, 17
print.summary.bdeb_data *(*summary.bdeb_data*)*, 51
print.summary.bdeb_model *(*summary.bdeb_model*)*, 53
print.summary.bdeb_prior *(*bdeb_prior-methods*)*, 17
prior_beta *(*priors*)*, 47
prior_beta(), *14*, *17*
prior_default, 45
prior_default(), *14*
prior_exponential *(*priors*)*, 47
prior_exponential(), *17*
prior_halfcauchy *(*priors*)*, 47
prior_halfcauchy(), *17*
prior_halfnormal *(*priors*)*, 47
prior_halfnormal(), *17*
prior_lognormal *(*priors*)*, 47
prior_lognormal(), *14*, *17*
prior_normal *(*priors*)*, 47
prior_normal(), *17*
prior_species, 46
priors, 47

repro_to_intervals, 49
repro_to_intervals(), *5*
residuals.bdeb_fit, 50

stats::confint(), *22*
stats::fitted(), *30*
stats::logLik(), *32*
stats::nobs(), *33*
stats::residuals(), *50*
stats::summary.lm(), *52*
stats::vcov(), *55*
summary(), *8*
summary.bdeb_data, 51
summary.bdeb_diagnostics, 51
summary.bdeb_diagnostics(), *8*
summary.bdeb_fit, 52
summary.bdeb_model, 53
summary.bdeb_prediction, 54
summary.bdeb_prior *(*bdeb_prior-methods*)*, 17
suppressMessages(), *9*

vcov.bdeb_fit, 55