Package: bayesTLS (via r-universe)

July 21, 2026

Title Joint Bayesian 4PL Models for Thermal Load Sensitivity

Version 1.0.0

Description Fits joint Bayesian four-parameter logistic (4PL) models to thermal-tolerance proportion data, extracts the classical thermal load sensitivity quantities (z, CTmax at 1 hour, T_crit) with full posterior uncertainty, and predicts heat-injury accumulation and survival under fluctuating temperature regimes with optional Sharpe-Schoolfield repair. Models are fitted with 'Stan' via the 'brms' package. Implements the framework described in Noble, Arnold, Nakagawa and Pottier (in preparation).

License CC BY 4.0

Encoding UTF-8

RoxygenNote 7.3.3

URL https://github.com/daniel1noble/bayesTLS

BugReports https://github.com/daniel1noble/bayesTLS/issues

Imports brms, dplyr, ggplot2, MASS, methods, patchwork, posterior, stats, tibble, utils

Suggests cmdstanr, glmmTMB, here, pkgload, readxl, testthat (>= 3.0.0), tidybayes, tidyr

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

Config/testthat/edition 3

Depends R (>= 4.1.0)

LazyData true

NeedsCompilation no

Author Daniel W. A. Noble [aut, cre], Pieter A. Arnold [aut], Shinichi Nakagawa [aut], Patrice Pottier [aut]

Maintainer Daniel W. A. Noble <daniel.noble@anu.edu.au>

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

Date/Publication 2026-07-21 11:10:19 UTC

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

RemoteRef HEAD

RemoteSha 65c95f03246889af60673544aaf220a465c8e555

Contents

aphid_tdt . . . 3
bayes_R2_tls . . . 4
clock_to_minutes . . . 5
derive_tdt_curve . . . 5
derive_tdt_landscape . . . 7
derive_temperature_for_duration . . . 8
derive_z . . . 10
diagnose_tdt_fit . . . 11
dsuzukii . . . 12
extract_4pl_pars . . . 13
fit_4pl . . . 15
format_interval . . . 17
get_4pl_est . . . 18
get_brmsfit . . . 19
get_hi_draws . . . 20
get_surv_draws . . . 20
get_tls_est . . . 21
has_fit . . . 22
make_4pl_formula . . . 22
make_4pl_priors . . . 25
make_temperature_scenarios . . . 27
planted_dose_from_trace . . . 29
plot.bayes_tls . . . 30
plot_heat_injury . . . 30
plot_repair_rate . . . 31
plot_survival_curves . . . 32
plot_tdt_curve . . . 33
plot_tdt_landscape . . . 34
plot_temperature_density . . . 36
plot_temperature_scenarios . . . 36
predict_heat_injury . . . 37
predict_survival_curves . . . 39
print.bayes_tls . . . 40
repair_rate_schoolfield . . . 41
snowgum_psii . . . 42
standardize_data . . . 44
summarise_observed_survival . . . 46
summary.bayes_tls . . . 47
tdt_parameter_table . . . 48
tdt_quantile . . . 49
theme_tdt . . . 49

tls 50
ts_ci 52
ts_curve 53
ts_stage1 54
ts_stage2 55
zebrafish_o2 56

Index **58**

`aphid_tdt` *Cereal-aphid lethal-TDT data, three species across three ages*

Description

Survival of three cereal-aphid species across a broad range of stressful high and low temperatures, the model-ready frame for the multi-species case study. Aphids of three ages (2, 6, 12 days old) were exposed to a heat branch (34–40 degrees C) or a cold branch (-11 to -3 degrees C) for a range of durations and scored alive/dead after recovery. One row per assay group; `branch` flags the heat vs cold series. Subset to one `branch` (and typically one `age`) and fit `CTmax` and `z` as functions of `species` in one joint 4PL to compare species with full posterior uncertainty.

Usage

```
aphid_tdt
```

Format

A data frame with 3041 rows and 7 variables:

species Species, a factor with levels `M_dirhodum` (*Metopolophium dirhodum*), `S_avenae` (*Sitobion avenae*), `R_padi` (*Rhopalosiphum padi*).

age Age in days, a factor with levels `2`, `6`, `12`.

branch Stress branch, a factor with levels `heat` (34–40 degrees C), `cold` (-11 to -3 degrees C).

temp Assay temperature (degrees C).

duration_min Exposure duration (minutes).

n_total Number of aphids treated.

n_surv Number surviving after treatment and recovery.

Source

Li Y-J, Chen S-Y, Jørgensen LB, Overgaard J, Renault D, Colinet H, Ma C-S (2023). Data for: Interspecific differences in thermal tolerance landscape explain aphid community abundance under climate change. Dryad, doi:10.5061/dryad.mcvdnck4j (Dryad CC0). Associated article: *Journal of Thermal Biology* 114: 103583, doi:10.1016/j.jtherbio.2023.103583. Raw file: `system.file("extdata", "data_lethal_TDT_aphid.csv", package = "bayesTLS")`.

Examples

```
## Not run:
a <- subset(aphid_tdt, branch == "heat" & age == "6")
std <- standardize_data(a, temp = "temp", duration = "duration_min",
                        n_total = "n_total", n_surv = "n_surv",
                        duration_unit = "minutes")
wf <- fit_4pl(std, ctmax = ~ 0 + species, z = ~ 0 + species, t_ref = 60)
tls(wf, by = "species", lethal = TRUE)   # z, CTmax, T_crit per species

## End(Not run)
```

`bayes_R2_tls` *Bayesian R^2 for a fitted TDT workflow*

Description

A thin tidy wrapper around `brms::bayes_R2()` for a fitted `bayes_tls` workflow: it pulls the underlying `brmsfit` (erroring clearly on an unfitted workflow, via `get_brmsfit()`) and returns the posterior median R^2 with its standard error and 95% credible interval as a one-row tibble, rather than the bare matrix `brms::bayes_R2()` returns. Pass `...` straight through to `brms::bayes_R2()` (e.g. `re_formula = NA` to exclude group-level effects, or `ndraws =` to subsample). Map over a named list of workflows to build a multi-fit table.

Usage

```
bayes_R2_tls(workflow, ...)
```

Arguments

`workflow`	A fitted `bayes_tls` workflow returned by `fit_4pl()`.
`...`	Further arguments passed to `brms::bayes_R2()`.

Value

A one-row tibble with columns `estimate`, `est_error`, `lower`, and `upper` (the lower and upper credible bounds, 2.5% and 97.5% by default; controlled by `probs` passed via `...`).

See Also

`brms::bayes_R2()`, `diagnose_tdt_fit()`

Examples

```
## Not run:
wf <- fit_4pl(std)
bayes_R2_tls(wf)
# Multiple fits in one table:
purrr::imap_dfr(list(binom = wf1, beta = wf2),
```

```
                ~ cbind(model = .y, bayes_R2_tls(.x)))

## End(Not run)
```

`clock_to_minutes` *Convert various clock formats to minutes*

Description

Accepts POSIXt, hms / difftime, numeric fractions of a day (Excel time) or bare numeric minutes, and character strings: `"HH:MM:SS"`, `"HH:MM"`, bare numeric strings (minutes), and durations beyond 24 h (e.g. `"25:30:00"`). Character strings are parsed element-wise; malformed entries become `NA`.

Usage

```
clock_to_minutes(x)
```

Arguments

`x`	Time value(s).

Value

Numeric vector of minutes.

Examples

```
clock_to_minutes("08:30:00")
clock_to_minutes("25:30")   # 25 h 30 min = 1530 min
clock_to_minutes(0.5)       # half a day = 720 min
```

`derive_tdt_curve` *Posterior LT_x curve: time to reach a survival target at each temperature*

Description

Returns the per-draw duration at which population-level survival crosses the chosen threshold, at each temperature in `temp_grid`.

Usage

```
derive_tdt_curve(
  workflow,
  temp_grid,
  duration_grid = NULL,
  target_surv = "relative",
  ndraws = NULL,
  probs = c(0.025, 0.5, 0.975),
  time_multiplier = NULL,
  output_time_unit = "min",
  by = NULL
)
```

Arguments

workflow	Fitted bayes_tls.
temp_grid	Numeric vector of temperatures (°C).
duration_grid	Numeric vector of durations along which to search. Only used in "absolute" mode. Default: 350 log-spaced values spanning 0.2× to 5× the training data's duration range.
target_surv	Threshold mode. "relative" (default; = (low + up)/2), "absolute" (= 0.5), or a numeric in (0, 1).
ndraws	Posterior draws to use; NULL (default) uses the full posterior. Subsampling can shift results slightly off the model, so the default keeps derived quantities consistent with the fit; pass an integer for speed.
probs	Quantile probabilities for the summary. Default c(0.025, 0.5, 0.975).
time_multiplier	Multiplier from model time units to output_time_unit (e.g. 60 for an hours model → min). NULL (default) derives it automatically from the workflow's duration_unit and output_time_unit, so a minutes model and an hours model both give the correct result without manual tuning. Pass a value to override.
output_time_unit	Label for the output time unit. Default "min".
by	Optional moderator column(s) for per-group LT curves. NULL (default) uses the fit's recorded moderators; a single-condition fit returns one ungrouped curve.

Details

Two threshold modes are supported via target_surv:

- "relative" (default): the duration at which survival reaches the midpoint between the fitted lower and upper asymptotes, i.e. (low + up)/2. This is the 4PL mid parameter on the natural time axis, returned directly from posterior_linpred(nlpar = "mid") — no numerical inversion. When low is about 0 and up about 1 it coincides with the classical LT50.

- `"absolute"` (or a numeric `p` in `(0, 1)`): the duration at which survival crosses the literal probability `p` (0.5 by default). The inversion is numerical — predict survival on a dense duration grid, then `approx()` through `p`.

This is the **horizontal** read of the survival surface: fix a survival threshold, read off the time required to reach it at each temperature.

Value

A list with `draws` (per-draw threshold durations; `target_surv` column is a character label), `summary` (quantile summary by temperature, plus moderator columns for grouped fits), `target_surv` (the label), `time_multiplier`, `output_time_unit`, and `by`.

Examples

```
## Not run:
wf  <- fit_4pl(std)
crv <- derive_tdt_curve(wf, temp_grid = c(38, 40, 42))  # relative LT curve
crv$summary

## End(Not run)
```

`derive_tdt_landscape` *Posterior survival landscape across the temperature × duration plane*

Description

Returns the posterior median (and 95% credible band) of survival probability on a 2-D grid covering the experimental range. Suitable for plotting as a heatmap or contour with `plot_tdt_landscape()` (Phase 1d). For curves at a handful of temperatures, use `predict_survival_curves()` instead.

Usage

```
derive_tdt_landscape(
  workflow,
  temp_grid = NULL,
  duration_grid = NULL,
  ndraws = NULL,
  probs = c(0.025, 0.5, 0.975),
  by = NULL
)
```

Arguments

`workflow`	Fitted `bayes_tls`.
`temp_grid`	Numeric vector of temperatures (°C). Default: 120 equally spaced values across the training-data range.

duration_grid	Numeric vector of durations. Default: 120 equally spaced values across the training-data range. A linear (rather than log-spaced) default keeps the grid regular on the linear duration axis that plot_tdt_landscape() uses by default, so the survival surface renders without uneven-raster artefacts.
ndraws	Posterior draws to use; NULL (default) uses the full posterior. Pass an integer to subsample for speed.
probs	Quantile probabilities. Default c(0.025, 0.5, 0.975).
by	Optional moderator column(s) for per-group landscapes. NULL (default) uses the fit's moderators; a single-condition fit returns one ungrouped landscape.

Value

A list with the same shape as predict_survival_curves() (a summary tibble with temp, duration, survival_median, survival_lower, survival_upper, plus any moderator columns for grouped fits, draws_matrix and grid).

Examples

```
## Not run:
wf  <- fit_4pl(d, ...)
lsp <- derive_tdt_landscape(wf)
lsp$summary

## End(Not run)
```

---

derive_temperature_for_duration

Temperature at which survival equals a target after a fixed exposure

---

Description

The **vertical** read of the survival surface: fix an exposure duration, find the temperature at which the posterior survival reaches the chosen threshold. Returns one temperature per posterior draw.

Usage

```
derive_temperature_for_duration(
  workflow,
  exposure_duration,
  temp_grid,
  target_surv = "relative",
  ndraws = NULL,
  probs = c(0.025, 0.5, 0.975),
  seed = NULL,
  by = NULL
)
```

Arguments

workflow	Fitted bayes_tls.
exposure_duration	Numeric scalar — the fixed duration (model units).
temp_grid	Numeric vector of temperatures to search over.
target_surv	Threshold mode. "relative" (default), "absolute", or a numeric in (0, 1).
ndraws	Posterior draws to use; NULL (default) uses the full posterior. Pass an integer to subsample.
probs	Quantile probabilities. Default c(0.025, 0.5, 0.975).
seed	Optional integer seeding the draw subsample for reproducibility. NULL (default) leaves the RNG alone.
by	Optional moderator column(s) for per-group results. NULL (default) uses the fit's moderators; a single-condition fit returns one ungrouped result.

Details

Threshold modes (via target_surv) match derive_tdt_curve():

- "relative" (default) → temperature at which mid(T) = log10(exposure_duration) per draw. The inversion is done analytically per draw: extract posterior_linpred(nlpar = "mid") over temp_grid, then approx() to the target log10-time.
- "absolute" (= 0.5) or numeric p in (0, 1) → existing numerical inversion of the 4PL survival surface at the literal probability p.

This is the primitive used by tls() to derive CTmax at t_ref.

Value

A list with draws (per-draw threshold temperatures; target_surv column is a character label), summary (quantile summary), exposure_duration, target_surv (the label), target_mode, target_prob. A grouped fit adds the moderator column(s).

Examples

```
## Not run:
wf <- fit_4pl(std)
# Temperature giving the relative midpoint threshold after a 60-unit exposure:
tt <- derive_temperature_for_duration(wf, exposure_duration = 60,
                                      temp_grid = seq(36, 44, by = 0.1))
tt$summary

## End(Not run)
```

`derive_z` *Per-draw thermal sensitivity z directly from the joint posterior*

Description

Derives $z = -1/(\mathrm{d}/\mathrm{d}T \log_{10} \mathrm{LT}(T))$ per posterior draw, read straight from the fitted 4PL coefficients — **no regression**. There are two regimes:

Usage

```
derive_z(
  workflow,
  target_surv = "relative",
  temp_grid = NULL,
  ndraws = NULL,
  probs = c(0.025, 0.5, 0.975),
  h = 0.001,
  seed = NULL,
  by = NULL
)
```

Arguments

`workflow`	Fitted `bayes_tls`.
`target_surv`	Threshold mode: `"relative"` (default; = `(low + up)/2`), `"absolute"` (= 0.5), or a numeric in `(0, 1)`.
`temp_grid`	Temperatures at which to evaluate local z and over which to pool. Default: the observed (unique) assay temperatures — pooling only where the data inform the curve.
`ndraws`	Posterior draws to subsample, or `NULL` (default) for all.
`probs`	Quantile probabilities for the summaries. Default `c(0.025, 0.5, 0.975)`.
`h`	Temperature step (°C) for the central finite difference. Default `1e-3`. (For a linear midpoint — the relative threshold — the central difference is exact regardless.)
`seed`	Optional integer seeding the draw subsample (relevant only when `ndraws` is set) for reproducibility. `NULL` (default) leaves the RNG untouched.
`by`	Optional moderator column(s) for per-group z. `NULL` (default) uses the fit's moderators (`meta$group_vars`); a single-condition fit then returns one ungrouped result.

Details

- **Relative threshold** (default; the $(\ell + u)/2$ midpoint): $\log_{10} \mathrm{LT}_{\mathrm{rel}}(T) = \mathrm{mid}(T) = \beta_0 + \beta_1(T - \bar{T})$ is exactly linear, so $z = -1/\beta_1$ where β_1 is the temperature slope on `mid` (`b_mid_temp_c`). The asymptotes ℓ, u and slope k do not enter (the midpoint cancels the curve asymmetry). z is constant in temperature.

- **Absolute threshold** p: the LT curve gains the asymmetry-correction term, $\log_{10} \mathrm{LT}_p(T) = \mathrm{mid}(T) + \frac{1}{k(T)} \log \frac{u(T)-p}{p-\ell(T)}$. When ℓ, u or k carry temperature effects this bends the curve, so z varies with temperature. A **local** $z(T) = -1/m(T)$ is computed at each assay temperature, where the local slope $m(T)$ is obtained by a central finite difference of the closed-form LT curve (step `h`). When the shape parameters are constant in T the correction is flat and this reduces to $-1/\beta_1$.

The returned **pooled** z (the default single summary) is the per-draw mean of the local $z(T)$ over `temp_grid`. The full per-temperature local $z(T)$ is also returned. z is invariant to the time unit (a constant time-multiplier shifts the LT intercept, not its slope), so no `time_multiplier` is needed here.

Value

A list with:

- `draws`: tibble `(.draw, z)` — pooled per-draw z.
- `summary`: tibble `(z_median, z_lower, z_upper)`.
- `local_draws`: tibble `(.draw, temp, z)` — local z(T) per draw.
- `local_summary`: tibble `(temp, z_median, z_lower, z_upper)`.
- `target_surv`, `temp_grid`. For a grouped fit each tibble gains the moderator column(s).

Examples

```
## Not run:
wf <- fit_4pl(std)
z  <- derive_z(wf)             # relative: z = -1 / b_mid_temp_c per draw
z$summary
derive_z(wf, target_surv = "absolute")$local_summary  # local z(T)

## End(Not run)
```

`diagnose_tdt_fit` *Sampling diagnostics for a fitted TDT workflow*

Description

Returns a tibble with the per-fit summary numbers a reviewer will want at a glance: max Rhat, minimum bulk and tail ESS, divergent transitions, tree-depth saturations, minimum BFMI per chain, and pass flags for each criterion plus a combined `all_pass`. Healthy values:

- Rhat < 1.01
- ESS bulk and tail > 400
- zero divergent transitions
- zero tree-depth saturations
- BFMI > 0.3 per chain

Usage

```
diagnose_tdt_fit(workflow)
```

Arguments

workflow	A fitted bayes_tls.

Details

The treedepth_max field is the model's max_treedepth setting (the ceiling passed to brms::brm(), default 10); a saturation is a post-warmup transition that hit that ceiling. A fit that merely tops out *below* the ceiling has zero saturations. Per-chain BFMI is computed from the energy diagnostic in brms::nuts_params(.) following the standard Stan definition (Var(Delta E)/Var(E)).

Value

A tibble with one row of diagnostic statistics.

Examples

```
## Not run:
diagnose_tdt_fit(wf)

## End(Not run)
```

---

dsuzukii *Drosophila suzukii multi-trait thermal-tolerance data*

Description

Per-individual thermal-tolerance assays for the spotted-wing fly (*Drosophila suzukii*), one row per fly, carrying three thermal-tolerance endpoints measured under static heat exposures at 34–38 degrees C: a lethal endpoint (dead), a sublethal knockdown time-to-event (t_coma), and a sublethal reproductive endpoint (prod). The model-ready frame for Case Study 4; aggregate dead to counts for the beta-binomial lethal fit, or use t_coma / prod directly for the sublethal endpoints. lvl indexes the exposure-duration grid as a percentage of the estimated median time-to-coma from the authors' initial TDT curves; time is the realised duration in minutes.

Usage

```
dsuzukii
```

Format

A data frame with 1407 rows and 9 variables:

id Unique individual identifier (`temp-lvl-sex-rep`).

temp Assay temperature (degrees C).

lvl Exposure duration as a percentage of the estimated median time-to-coma from the authors' initial TDT curves.

time Exposure duration (minutes).

sex Sex, a factor with levels `F`, `M`.

rep Replicate vial within a temperature x lvl x sex cell.

prod Reproductive productivity (offspring per female per day).

dead Mortality indicator: `1` = died, `0` = survived.

t_coma Time to heat coma (minutes); `NA` where no coma was recorded for that individual.

Source

Ørsted M, Willot Q, Olsen AK, Kongsgaard V, Overgaard J (2024). Data for: Thermal limits of survival and reproduction depend on stress duration: a case study of *Drosophila suzukii*. Zenodo, doi:10.5281/zenodo.10602268 (distributed under CC BY 4.0). Associated article: doi:10.1111/ele.14421. Raw file: `system.file("extdata", "data_multitrait_TDT_drosophila_suzukii.csv", package = "bayesTLS")`.

Examples

```
# Lethal endpoint: aggregate per-individual deaths to cell counts, then
# standardise for the beta-binomial 4PL.
cells <- dplyr::summarise(
  dplyr::group_by(dsuzukii, temp, time, sex),
  n_total = dplyr::n(), n_dead = sum(dead), .groups = "drop")
std <- standardize_data(cells, temp = "temp", duration = "time",
                        n_total = "n_total", n_dead = "n_dead",
                        random_effects = "sex", duration_unit = "minutes")
```

`extract_4pl_pars` *Per-draw natural-scale 4PL parameters from a fitted workflow*

Description

Faithfully extracts ALL FOUR natural-scale 4PL sub-parameters (`low`, `up`, `k`, `mid`) per posterior draw, at one or more assay temperatures and per moderator group, retaining whatever temperature and/or group structure each sub-parameter was fit with (`~ 1`, `~ temp_c`, `~ temp_c * group`, random effects, ...). It surfaces the shared TDT engine (`tls_eval_subpars()` over a `tls_build_grid()` temperature × moderator grid via `brms::posterior_linpred(nlpar=)`), so the result is parameterisation- and coding-agnostic — no coefficient-name parsing.

Usage

```
extract_4pl_pars(
  workflow,
  temps = NULL,
  by = NULL,
  re_formula = NA,
  ndraws = NULL
)
```

Arguments

workflow	Fitted bayes_tls.
temps	Numeric vector of assay temperatures (°C, uncentred) to evaluate the curve parameters at. NULL (default) uses the fit's observed unique assay temperatures.
by	Optional moderator column(s) for per-group parameters. NULL (default) uses the fit's moderators (all levels); a single-condition fit returns the ungrouped tibble. A grouped fit prepends the moderator column(s).
re_formula	Passed to brms::posterior_linpred(). NA (default) marginalises out group-level (random) effects so the result is a population-level curve; NULL conditions on the fitted random effects.
ndraws	Posterior draws to use; NULL (default) uses the full posterior. Pass an integer to subsample (the caller is responsible for set.seed() if a reproducible subsample is wanted).

Details

mid is the natural-scale midpoint sub-parameter (the log10 time at the per-draw relative threshold) evaluated at each temperature; on an absolute fit it is still the bare mid nlpar (the asymmetry correction is applied downstream by the threshold-specific readers, not folded in here).

This replaces the earlier constant-shape extractor that discarded the temperature slopes on low/up/k. The constant-in-T view the heat-injury integral needs (low, up, k at the centring temperature plus a linear mid_int/mid_temp) now lives in the internal helper hi_pars(), which predict_heat_injury() calls for shape = "constant".

Value

A long tibble with (.draw, temp, low, up, k, mid) columns (plus the moderator column(s) for a grouped fit, immediately after .draw), one row per draw × temperature × group, filtered to rows producing valid parameter values (k > 0, up > low, all finite). temp is the uncentred assay temperature (°C).

See Also

hi_pars() for the constant-shape view used by the heat-injury integral; get_4pl_est() which wraps this for the "draws" view.

Examples

```
## Not run:
pars <- extract_4pl_pars(wf)                # observed temps, fit's groups
extract_4pl_pars(wf, temps = c(30, 33, 36))  # at chosen temperatures
head(pars)

## End(Not run)
```

`fit_4pl` *Fit the joint Bayesian 4PL to standardised TDT data*

Description

Wraps `make_4pl_formula()` + `make_4pl_priors()` + `brms::brm()`. Returns a workflow object containing the fit, data, formula, prior, and metadata that downstream helpers (e.g. `tls()`) read from.

Usage

```
fit_4pl(
  data,
  random_effects = NULL,
  temp_effects = c("low", "up", "k", "mid"),
  lower = 0,
  upper = 1,
  family = NULL,
  prior = NULL,
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  cores = chains,
  seed = 123,
  backend = "cmdstanr",
  control = list(adapt_delta = 0.95, max_treedepth = 12),
  init = 0,
  file = NULL,
  file_refit = "on_change",
  fit = TRUE,
  bounds = NULL,
  ctmax = NULL,
  z = NULL,
  up = NULL,
  low = NULL,
  k = NULL,
  mid = NULL,
  by = NULL,
  threshold = c("relative", "absolute"),
```

```
  p = 0.5,
  t_ref = 60,
  ...
)
```

Arguments

data	Output of standardize_data().
random_effects	Optional character vector of grouping variables for random intercepts on mid.
temp_effects	Which 4PL sub-parameters carry a temp_c slope; a subset of c("low", "up", "k", "mid") (default all four). "mid" is always required. The all-four default needs roughly >= 15 cells per fixed effect to be stable; for sparse designs (few temperatures x durations, small n) prefer temp_effects = "mid" (constant-shape), which estimates only the midpoint's temperature slope (= -1/z) and shares the asymptotes and steepness across temperature. Passed to make_4pl_formula().
lower, upper	Response-scale bounds for the asymptotes. Default 0, 1; pass lower = 0.85, upper = 1 for sublethal data bounded above zero.
family	brms family. NULL (default) picks the family from the data's response_type metadata: beta_binomial(link = "identity") for count data and brms::Beta(link = "identity") for a continuous proportion response. Pass an explicit family to override (e.g. binomial(link = "identity") for the no-overdispersion count case).
prior	Optional brmsprior object. If NULL (default), make_4pl_priors() builds defaults from data, lower, upper, and random_effects.
chains, iter, warmup, cores, seed	Sampling arguments passed to brms::brm().
backend	Either "cmdstanr" (default) or "rstan".
control	List of HMC control parameters.
init	Initial values for the sampler. Default 0 initialises every unconstrained parameter at zero.
file	Optional path (without extension) to cache the fit.
file_refit	Passed to brms::brm(). Default "on_change".
fit	Logical. If FALSE, returns the workflow spec without fitting — useful for inspecting the formula and priors.
bounds	Length-2 c(lower, upper) asymptote interval; supersedes lower/upper when supplied.
ctmax, z, up, low, k	One-sided formulas selecting the **direct** CTmax/z parameterisation (see make_4pl_formula()). Supplying ctmax and/or z fits CTmax and z directly; up/low/k are inherited from them when omitted. up/low/k also take explicit formulas in the **midpoint** parameterisation.
mid, by	**Midpoint** parameterisation only (see make_4pl_formula()). mid is an explicit one-sided formula for the midpoint (must carry temp_c); by is a moderator column-name shortcut applying the moderator to all four sub-parameters

	(low/up/k/mid) at once, so a per-group CTmax/z fit needs only by = "<moderator>". Either records the moderator in meta$group_vars, so tls() auto-derives per-group quantities with by = . Both error in direct mode.
threshold	"relative" (default) or "absolute"; the threshold the fitted CTmax/z refer to (direct mode only).
p	Survival fraction for the "absolute" threshold (direct mode). Default 0.5 (LT50). The disjoint-bounds reparam splits the asymptotes at the bounds' midpoint, so an absolute p is only achievable *near that midpoint* (with the default bounds [0, 1] the achievable range is roughly (0.499, 0.501)); fit_4pl() errors otherwise. For any other LTx (LT60, LT90, ...) or a sub-unit-bounded response, fit with threshold = "relative" and derive it afterwards via tls() with target_surv = .
t_ref	Reference exposure for CTmax, in **minutes** (default 60, i.e. one hour). Converted to the model's log10-time scale via the data's duration_unit.
...	Further arguments passed to brms::brm().

Details

By default all four 4PL sub-parameters get a temp_c slope, and random intercepts attach to mid only. Use temp_effects = "mid" for the classical constant-shape TDT model (recommended for sparse designs — see temp_effects). To fit separate categories (life stages, species, populations), filter data per category and call this function once per subset.

Value

A list of class "bayes_tls" with elements fit, data, formula, prior, meta. See print.bayes_tls(), summary.bayes_tls(), and plot.bayes_tls() for the available methods.

Examples

```
## Not run:
d  <- standardize_data(raw, ...)
wf <- fit_4pl(d, fit = FALSE)         # inspect spec only
wf <- fit_4pl(d, file = "output/models/my_fit")
wf <- fit_4pl(d, lower = 0.85)        # sublethal data

## End(Not run)
```

---

format_interval *Format a posterior median plus credible interval as a single string*

Description

Format a posterior median plus credible interval as a single string

Usage

```
format_interval(median, lower, upper, digits = 2)
```

Arguments

median, lower, upper	Numeric (scalar or vector).
digits	Integer rounding precision.

Value

Character like "5.12 [4.87, 5.4]". A non-finite median yields NA_character_ (rather than "NA [...]"); a non-finite bound is shown as an en dash, so the strings stay table-ready.

Examples

```
format_interval(5.123, 4.872, 5.401)
format_interval(NA, 1, 2)   # -> NA
```

---

get_4pl_est *Posterior estimates of the natural-scale 4PL parameters (draws or summary)*

Description

The 4PL-shape companion to get_tls_est(). Returns the posterior of the natural-scale 4PL curve parameters from a fitted fit_4pl() workflow, as either per-draw values or a median + 95% credible-interval summary, per moderator group. Whereas get_tls_est() operates on a tls() object and returns the derived TLS quantities (z, CTmax, Tcrit), this operates on the fitted workflow and returns the raw curve parameters those quantities are derived from. A thin wrapper: "summary" calls tdt_parameter_table(), "draws" calls extract_4pl_pars().

Usage

```
get_4pl_est(workflow, what = c("summary", "draws"), by = NULL, temps = NULL)
```

Arguments

workflow	A fitted fit_4pl() workflow (workflow$fit not NULL).
what	"summary" (default; one row per parameter with median + 95% CrI at the centring temperature T_bar, via tdt_parameter_table() – this also includes the derived z slope row) or "draws" (per-draw natural- scale low, up, k, mid at each assay temperature, via extract_4pl_pars()).
by	Optional moderator column(s) to group by, passed through. NULL (default) uses the fit's own grouping for a grouped fit, else one group.
temps	For what = "draws", numeric vector of assay temperatures (°C) to evaluate the curve parameters at, passed to extract_4pl_pars(). NULL (default) uses the fit's observed unique assay temperatures. Ignored for what = "summary" (which always reports the T_bar slice).

Value

A tibble (plus the moderator column(s) for a grouped fit): for `"summary"`, one row per parameter with median and credible bounds at `T_bar`; for `"draws"`, one row per posterior draw × temperature with `temp`, `low`, `up`, `k`, `mid` (and `.draw`).

See Also

`get_tls_est()` for the derived TLS quantities; `tdt_parameter_table()` and `extract_4pl_pars()`, which this wraps.

Examples

```
## Not run:
wf <- fit_4pl(standardize_data(my_data, temp = "temp", duration = "time",
                               n_total = "n", n_dead = "dead"))
get_4pl_est(wf, "summary")                # low/up/k/mid median + 95% CrI
get_4pl_est(wf, "draws", by = "species")   # per-draw params, per group

## End(Not run)
```

---

get_brmsfit *Extract the fitted brms model from a workflow*

Description

Returns the underlying brms::brmsfit object held in a `bayes_tls` workflow, ready for any brms / posterior / bayesplot helper (e.g. `brms::bayes_R2()`, `brms::mcmc_plot()`, `posterior::as_draws_df()`). Prefer this over reaching into `workflow$fit` directly: it errors clearly on an unfitted workflow instead of silently returning `NULL`. Pairs with the predicate `has_fit()`.

Usage

```
get_brmsfit(workflow)
```

Arguments

`workflow`	A fitted `bayes_tls` workflow returned by `fit_4pl()`.

Value

The `brmsfit` object stored in the workflow.

Examples

```
## Not run:
wf  <- fit_4pl(std)
fit <- get_brmsfit(wf)
brms::bayes_R2(fit)

## End(Not run)
```

get_hi_draws *Posterior draws of heat-injury trajectory*

Description

Long-format tibble of per-draw HI, dose, survival and mortality at every time step of the supplied temperature trace. Requires that `predict_heat_injury()` was called with `save_draws = TRUE`.

Usage

```
get_hi_draws(hi)
```

Arguments

`hi`	The list returned by `predict_heat_injury(..., save_draws = TRUE)`.

Value

A tibble with columns `.draw`, `time`, `temp`, `dose`, `hi`, `survival`, `mortality`.

Examples

```
## Not run:
hi <- predict_heat_injury(trace, wf, save_draws = TRUE)
get_hi_draws(hi)

## End(Not run)
```

get_surv_draws *Posterior draws of survival*

Description

Accepts either a `predict_survival_curves()` result (static survival on a temperature × duration grid) or a `predict_heat_injury()` result (dynamic survival along a temperature trace). In the heat-injury case requires `save_draws = TRUE`.

Usage

```
get_surv_draws(x)
```

Arguments

`x`	A list returned by `predict_survival_curves()` or by `predict_heat_injury()`.

Value

A long-format tibble. For `predict_survival_curves()`: `.draw`, `temp`, `duration`, `survival`. For `predict_heat_injury()`: `.draw`, `time`, `temp`, `survival`. For a grouped fit the moderator column(s) are also carried (so per-group draws are not collapsed).

Examples

```
## Not run:
psc <- predict_survival_curves(wf, temps = c(32, 34),
                               durations = c(0.5, 1))
get_surv_draws(psc)

hi <- predict_heat_injury(trace, wf, save_draws = TRUE)
get_surv_draws(hi)

## End(Not run)
```

`get_tls_est` *Extract TLS estimates (draws or summary) from a* `tls` *object*

Description

Pulls the posterior `draws` or the `summary` (median + 95% credible interval) for any or all thermal-load-sensitivity quantities (`z`, `CTmax`, and, when the fit was extracted with `lethal = TRUE`, `Tcrit`) from a `tls()` result. Draws are returned in tidy-long form (`quantity`, `.draw`, `value`, plus any grouping column), so a posterior contrast is a join on `.draw` followed by a difference of `value`.

Usage

```
get_tls_est(x, what = c("summary", "draws"), params = NULL)
```

Arguments

`x`	A `tls` object, the result of `tls()`.
`what`	Either `"summary"` (the default; median + lower/upper per quantity x group) or `"draws"` (one row per posterior draw per quantity x group).
`params`	Optional character vector selecting quantities to return, matched case-insensitively against `"z"`, `"CTmax"`, `"Tcrit"`. `NULL` (default) returns all available quantities.

Value

A tibble. For `what = "summary"`: columns `quantity`, `median`, `lower`, `upper` (plus grouping columns for a grouped fit). For `what = "draws"`: columns `quantity`, `.draw`, `value` (plus grouping columns).

See Also

`tls()`, which produces the object this reads.

Examples

```
## Not run:
fit <- fit_4pl(standardize_data(my_data, temp = "temp", duration = "time",
                                n_total = "n", n_dead = "dead"))
est <- tls(fit, lethal = TRUE)
get_tls_est(est, "summary")             # all quantities: median + 95% CrI
zd  <- get_tls_est(est, "draws", "z")   # z draws, ready for a contrast

## End(Not run)
```

---

has_fit *Whether a workflow has been fitted*

Description

Whether a workflow has been fitted

Usage

```
has_fit(workflow)
```

Arguments

workflow	Object returned by fit_4pl().

Value

Logical scalar.

Examples

```
wf <- list(fit = NULL); class(wf) <- "bayes_tls"
has_fit(wf)
```

---

make_4pl_formula *Build the brms formula for the joint Bayesian 4PL*

Description

Constructs the joint 4PL with the disjoint-bounds reparameterisation on the asymptotes (low, up) and a positivity reparam on k. All four 4PL sub-parameters (lowraw, upraw, logk, mid) get a temp_c slope — the model assumes temperature can affect every aspect of the dose-response curve. Random intercepts (if any) attach to mid.

Usage

```
make_4pl_formula(
  random_effects = NULL,
  bounds = NULL,
  family = brms::beta_binomial(link = "identity"),
  response_var = "survival",
  temp_effects = c("low", "up", "k", "mid"),
  ctmax = NULL,
  z = NULL,
  up = NULL,
  low = NULL,
  k = NULL,
  mid = NULL,
  by = NULL,
  threshold = c("relative", "absolute"),
  p = 0.5,
  log10_tref = 0,
  lower = 0,
  upper = 1
)
```

Arguments

random_effects	Optional character vector of grouping variables for random intercepts on mid.
bounds	Length-2 numeric c(lower, upper) giving the response-scale asymptote interval. Defaults to c(lower, upper). Preferred over the separate lower/upper (which it supersedes — see those).
family	brms family. Default beta_binomial(link = "identity"). Pass binomial(link = "identity") for the simpler no-overdispersion case, or brms::Beta(link = "identity") for a continuous proportion response. The count families (binomial, beta_binomial) require link = "identity" — the 4PL produces a probability directly, so any other link would double-transform it (an error is thrown). A known family name passed as a string (e.g. "binomial") is coerced to its identity-link constructor.
response_var	Response column name for a Beta (continuous proportion) fit. Ignored for count families. Default "survival" (the column standardize_data() writes for a proportion response).
temp_effects	Character vector naming which 4PL sub-parameters carry a temp_c slope: any subset of c("low", "up", "k", "mid"). Default is all four — temperature can affect every aspect of the dose-response curve. "mid" must always be present (its slope is $-1/z$, the core TDT quantity). Restricting to "mid" gives the classical *constant-shape* TDT model (asymptotes and slope shared across temperature), which is the right choice for sparse designs where the richer all-four model over-fits. Ignored in the direct CTmax/z parameterisation.
ctmax, z	One-sided formulas for the **direct** parameterisation. Supplying either switches make_4pl_formula() to fit CTmax (as CTmaxdev = CTmax - mean(temp)) and z (as logz) directly as nonlinear parameters, with mid reconstructed from them.

Carry the fixed and random effects for tolerance and sensitivity, e.g. `~ life_stage + (1 | Date)`. Omit `z` for a single pooled `z` (`logz ~ 1`).

`up`, `low`, `k` — Optional one-sided formulas for the asymptote and steepness sub-parameters. In **direct** mode, when omitted they inherit the `ctmax`/`z` **fixed** effects crossed with `temp_c` (random effects are not inherited); intercept-only `ctmax`/`z` gives `~ temp_c`. In **midpoint** mode they default to the `temp_effects` structure (or `by`, see below); an explicit formula always overrides.

`mid` — **Midpoint** mode only. Optional one-sided formula for the midpoint sub-parameter (whose `temp_c` slope is $-1/z$). Must carry `temp_c`. Defaults to the `temp_effects`/`by` structure. Errors in direct mode (where `mid` is reconstructed from `ctmax`/`z`).

`by` — **Midpoint** mode only. Optional character vector of moderator column(s). A shortcut that fits `~ temp_c * by` on **all four** sub-parameters (`low`/`up`/`k`/`mid`) at once – i.e. every aspect of the curve, plus the midpoint's temperature slope ($-1/z$), varies by group. `by` is the more specific instruction, so it overrides `temp_effects`. Equivalent to writing the four `~ temp_c * by` formulas by hand; use explicit formulas for finer per-parameter control. Errors in direct mode (group there via `ctmax = ~ 0 + <moderator>`).

`threshold` — `"relative"` (default) fits the midpoint backbone so `CTmax`/`z` are the relative-threshold quantities; `"absolute"` bakes the asymmetry correction $C(T) = (1/k)\log((up-p)/(p-low))$ into `mid` so they are the absolute (p-survival) quantities. The disjoint-bounds reparam splits the asymptotes at the bounds' midpoint, so `"absolute"` is only well defined for `p` *near that midpoint* (with the default bounds `[0, 1]` the achievable range is roughly `(0.499, 0.501)`); other LTx values (LT60, LT90, ...) must be obtained from a `"relative"` fit plus `tls()` with `target_surv = `, which inverts the fitted surface post hoc for any `p`.

`p` — Survival fraction defining the absolute threshold. Default `0.5`. Only meaningful when `threshold = "absolute"`, and (see `threshold`) only achievable for `p` near the bounds' midpoint – use the relative threshold + `target_surv` for any other LTx.

`log10_tref` — `log10` of the reference exposure (in the model's time unit) at which `CTmax` is defined. Default `0` (one time unit). `fit_4pl()` derives this from `t_ref` and the data's duration unit.

`lower`, `upper` — Response-scale bounds for the asymptotes. Default `0, 1` for proportion data; set narrower for sublethal data whose response sits well above `0`.

Details

The response term depends on the family:

- **Count families** (`binomial`, `beta_binomial`): the response is `n_surv | trials(n_total) ~ <4PL>`.
- `Beta` **family** (continuous proportion): the response is `<response_var> ~ <4PL>` with no `trials()` term. With `link = "identity"` the 4PL value is the mean directly (safe because the reparam keeps it in `(0, 1)`); with `link = "logit"` the 4PL is wrapped in `logit()` so the mean is still the 4PL value.

If a parameter has no real temperature effect, its `temp_c` slope shrinks toward zero under the prior and the fit reduces cleanly to the simpler case.

Value

A brmsformula object.

Examples

```
make_4pl_formula()
make_4pl_formula(family = binomial(link = "identity"))
make_4pl_formula(temp_effects = "mid")            # constant-shape TDT
make_4pl_formula(family = brms::Beta(link = "identity"),
                 response_var = "survival")
make_4pl_formula(ctmax = ~ life_stage, z = ~ life_stage)  # direct CTmax/z
```

---

make_4pl_priors *Default weakly informative priors for the joint Bayesian 4PL*

Description

Priors are weakly informative and do not adapt to the data (which would understate uncertainty). They are constructed to match make_4pl_formula()'s disjoint-bounds reparameterisation low = low_min + inv_logit(lowraw) * low_w and up = up_min + inv_logit(upraw) * up_w for the asymptote intervals derived from lower and upper by compute_4pl_bounds().

Usage

```
make_4pl_priors(
  data,
  lower = 0,
  upper = 1,
  random_effects = NULL,
  prior_random_sd = "exponential(2)",
  prior_phi = "gamma(2, 0.1)",
  bounds = NULL,
  ctmax = NULL,
  z = NULL,
  up = NULL,
  low = NULL,
  k = NULL
)
```

Arguments

data	Output of standardize_data(). Only logd is used, to centre the midpoint Intercept prior.
lower, upper	Response-scale bounds for the asymptotes. Default 0, 1 for proportion data. For sublethal data bounded above 0 (e.g. photosystem-II between 0.85 and 1), set lower accordingly.

`random_effects` Optional character vector of grouping variables for random intercepts on `mid`. Adds one `prior_random_sd` row per group.

`prior_random_sd`
Stan prior string for random-effect SDs. Default `"exponential(2)"`.

`prior_phi` Stan prior string for the beta-binomial precision parameter `phi`. Default `"gamma(2, 0.1)"`. Pass `NULL` to omit the `phi` prior — needed when fitting with `family = binomial()` (no overdispersion).

`bounds` Length-2 `c(lower, upper)` asymptote interval; supersedes `lower`/`upper` when supplied.

`ctmax`, `z`, `up`, `low`, `k`
One-sided formulas selecting the **direct** CTmax/z priors (matching `make_4pl_formula()`). Supplying `ctmax`/`z` switches to direct-mode priors: the same centred asymptote priors (per factor level for cell-means terms), coding-aware weakly-informative `CTmaxdev`/`logz` priors (the level prior lands on the Intercept or each factor level, while between-group contrasts and temperature slopes are centred on zero, so the per-group z/CTmax prior **mean** is invariant to cell-means vs treatment coding — this removes the log(3) shrinkage bias). The prior **variance** is not fully coding-invariant: under treatment coding a non-reference group's implied prior convolves the Intercept and contrast SDs (e.g. logz SD `sqrt(0.7^2 + 0.7^2)` ~ `0.99` vs `0.70` under cell-means), and `normalize_cellmeans()` only rewrites single-factor formulas. Prefer cell-means (`0 + G`) for multi-factor models to keep groups symmetric. Plus random-effect SD priors on `CTmaxdev`/`logz` only (never on the shape sub-parameters). `up`/`low`/`k` follow the same inheritance resolution as the formula.

Details

Each of the four 4PL sub-parameters (`lowraw`, `upraw`, `logk`, `mid`) gets an Intercept-specific prior plus a general `class = "b"` prior covering the `temp_c` slope. Random-effect SDs on `mid` get one `prior_random_sd` row per grouping variable.

Value

A `brmsprior` object ready to pass to `fit_4pl()`.

Examples

```
raw <- data.frame(
  temperature_C = rep(c(30, 32, 34), each = 4),
  exposure_h    = rep(c(1, 2, 4, 8), times = 3),
  n             = 30L,
  alive         = c(29, 28, 25, 5, 30, 27, 18, 2, 28, 22, 10, 1)
)
d <- standardize_data(raw, temp = "temperature_C", duration = "exposure_h",
                      n_total = "n", n_surv = "alive")
make_4pl_priors(d)
```

`make_temperature_scenarios` *Build reference temperature traces for heat-injury validation*

Description

Returns a list of four tibbles, each with columns `time` and `temp`, representing the canonical validation scenarios:

Usage

```
make_temperature_scenarios(
  baseline = 20,
  spike_temp = 30,
  n_hours = 96,
  dt_hours = 1,
  spike_times_single = 24,
  spike_times_multi = c(24, 48, 72),
  diurnal_n_days = 7,
  diurnal_night_temp = baseline,
  diurnal_day_peaks = NULL,
  diurnal_peak_fwhm = 6,
  diurnal_noise_sd = 0.3,
  diurnal_seed = 1L
)
```

Arguments

`baseline`	Numeric. Constant baseline temperature (°C). Default 20.
`spike_temp`	Numeric. Temperature of each spike (°C); each spike is one time step (`dt_hours`) wide. Default 30.
`n_hours`	Integer. Total length of each trace, in hours. Default 96.
`dt_hours`	Numeric. Time step, in hours. Default 1 (hourly).
`spike_times_single`	Integer vector. Hours at which the single-spike trace has a spike. Default `24`.
`spike_times_multi`	Integer vector. Hours at which the multi-spike trace has a spike. Default `c(24, 48, 72)`.
`diurnal_n_days`	Integer. Number of days in the diurnal scenario. Default 7.
`diurnal_night_temp`	Numeric vector of length `diurnal_n_days` (or length 1, recycled): night-time baseline temperature (°C) each day, applied at the cool end of that day's diurnal cycle. Passing a vector lets nights on hot days be warmer than nights on cool days (a common natural pattern). Default `baseline` (constant across days).

diurnal_day_peaks
: Numeric vector of length diurnal_n_days (or length 1, recycled): the peak temperature each day reaches near 14:00. If NULL (default) the function alternates cool (baseline + 5) and warm (baseline + 8) days so some days accrue HI and others do not.

diurnal_peak_fwhm
: Numeric. Full-width-half-maximum of the daily Gaussian temperature peak, in hours. Default 6 (about 4-5 hours near the daily peak).

diurnal_noise_sd
: Numeric. Standard deviation (°C) of smooth hourly fluctuations added on top of the diurnal cycle. Default 0.3.

diurnal_seed
: Integer. Seed for the hourly-noise RNG so the diurnal trace is reproducible. Default 1L.

Details

- flat — constant baseline temperature for n_hours hours.
- single_spike — flat baseline with one time step (dt_hours wide) at spike_temp placed at each of spike_times_single (default one spike).
- multi_spike — flat baseline with one time step (dt_hours wide) at spike_temp placed at each of spike_times_multi (default three spikes).
- diurnal — multi-day diurnal cycle with day-to-day variation in peak temperature and small hourly fluctuations on top. Useful as a stand-in for a natural thermal regime where some days exceed T_crit and accrue injury while others stay below.

Each spike is one time step (dt_hours) wide, so the analytical dose it delivers is the instantaneous rate times the step width, $100 \cdot 10^{(T_{spike} - CT_{max,1hr})/z} \cdot dt_{hours}$ % LT50-dose. With the default dt_hours = 1 this reduces to $100 \cdot 10^{(T_{spike} - CT_{max,1hr})/z}$, so setting spike_temp = CTmax_1hr delivers ~100% LT50-dose per spike — a calibration target for predict_heat_injury() validation. For dt_hours != 1 the dose scales by dt_hours.

Value

A named list of four tibbles (flat, single_spike, multi_spike, diurnal), each with columns time (numeric, hours from start) and temp (°C). Note the traces differ in length: flat, single_spike, and multi_spike span n_hours (default 96) while diurnal spans diurnal_n_days * 24 (default 168) hours.

Examples

```
scens <- make_temperature_scenarios()
lapply(scens, head)
# Diurnal calibrated for a shrimp-like organism (T_crit ~ 25 °C):
make_temperature_scenarios(
  baseline = 18,
  diurnal_n_days = 7,
  diurnal_night_temp = 19,
  diurnal_day_peaks  = c(24, 27, 24.5, 27.5, 23, 26.5, 25.5)
)$diurnal
```

`planted_dose_from_trace` *Analytical heat-injury accumulation along a temperature trace*

Description

Implements the classical HI integral exactly (Equation 8 of the manuscript), given known TDT parameters. Use this as the **truth** that `predict_heat_injury()` should recover when fed the same trace and a model fit whose posterior is consistent with `(z, CTmax_1hr, T_c)`.

Usage

```
planted_dose_from_trace(trace, z, CTmax_1hr, T_c)
```

Arguments

`trace`	Tibble with columns `time` and `temp`, output of `make_temperature_scenarios()`.
`z`	Thermal sensitivity, °C per 10-fold change in time.
`CTmax_1hr`	Static temperature at which LT50 = 1 hour, °C.
`T_c`	Damage threshold (°C); contributions below `T_c` are zero.

Details

Integrated by forward Euler from zero: the interval ending at point i accrues the rate at its start (point $i-1$) times that interval's width,

$$\Delta HI_i = 100 \cdot 10^{(T_{i-1} - CT_{max,1hr})/z} \cdot (t_i - t_{i-1})$$

when `T_{i-1} > T_c`, and zero otherwise (and `hi_inc[1] = 0`). Cumulative HI is the running sum. This matches `predict_heat_injury()`'s integrator.

Value

The input trace augmented with two new columns: - `hi_inc` — instantaneous HI contribution at each hour (%). - `hi_cumulative` — running sum of `hi_inc` (%).

Examples

```
scens <- make_temperature_scenarios(spike_temp = 30)
planted_dose_from_trace(scens$single_spike, z = 5, CTmax_1hr = 32, T_c = 25)
```

plot.bayes_tls *MCMC mixing plot for a fitted* bayes_tls *workflow*

Description

Delegates to `plot()` on the underlying `brmsfit` (via `get_brmsfit()`), which produces brms's default `mcmc_combo` layout (per-parameter density on the left, post-warmup trace on the right, chains coloured). Use this to eyeball chain mixing alongside the numeric Rhat / ESS in `summary.bayes_tls()`.

Usage

```
## S3 method for class 'bayes_tls'
plot(x, ...)
```

Arguments

`x`	A fitted `bayes_tls` workflow.
`...`	Passed through to `brms`'s `plot.brmsfit` method (e.g. `variable`, `combo`, `N`, `ask`).

Value

The brms plot output (invisibly): a list of `bayesplot_grid` grid objects, one per plotted page.

Examples

```
## Not run:
plot(wf)
plot(wf, variable = "^b_mid", regex = TRUE)
plot(wf, combo = c("dens_overlay", "trace"))

## End(Not run)
```

plot_heat_injury *Plot HI and survival trajectories from* predict_heat_injury()

Description

Two stacked panels: cumulative heat injury (%) on top, predicted survival fraction on the bottom, both with 95% credible bands.

Usage

```
plot_heat_injury(hi, lt50_threshold = 100)
```

Arguments

hi	Output of predict_heat_injury().
lt50_threshold	Numeric. Horizontal dashed line on the HI panel marking the LT50 dose threshold. Default 100.

Value

A ggplot object (combined via patchwork).

Examples

```
## Not run:
hi <- predict_heat_injury(scens$single_spike, wf)
plot_heat_injury(hi)

## End(Not run)
```

---

plot_repair_rate *Plot a Sharpe-Schoolfield repair TPC*

Description

Plot a Sharpe-Schoolfield repair TPC

Usage

```
plot_repair_rate(temp_grid, repair_pars)
```

Arguments

temp_grid	Numeric vector of temperatures (degC) to evaluate.
repair_pars	Named list of Sharpe-Schoolfield parameters (TA, TAL, TAH, TL, TH, TREF, r_ref).

Value

A ggplot object.

Examples

```
rp <- list(TA = 14065, TAL = 50000, TAH = 120000,
           TL = 10.5 + 273.15, TH = 22.5 + 273.15,
           TREF = 17 + 273.15, r_ref = 0.01)
plot_repair_rate(seq(5, 30, length.out = 200), rp)
```

`plot_survival_curves` *Plot posterior survival curves*

Description

Plots posterior median ± 95% CrI of survival as a function of duration, one coloured line per assay temperature. Optionally overlays observed survival proportions per replicate.

Usage

```
plot_survival_curves(
  pred,
  observed = NULL,
  log_time = FALSE,
  palette = "viridis",
  response_label = "Survival probability",
  clip_to_observed = TRUE,
  facet_scales = c("auto", "fixed", "free", "free_x", "free_y")
)
```

Arguments

`pred`	Output of `predict_survival_curves()`.
`observed`	Optional standardised data tibble (from `standardize_data()`) to overlay as points (jittered, transparent).
`log_time`	Logical. If `TRUE`, the exposure-duration axis is log10. Default `FALSE` (linear).
`palette`	Character. Viridis option for the temperature colour scale. Default `"viridis"`; pass `NULL` to keep ggplot's default hue scale.
`response_label`	Character. y-axis label for the modelled response. Default `"Survival probability"`; set to e.g. `"Retained PSII function"` for a continuous-proportion (Beta) endpoint where the response is not survival.
`clip_to_observed`	Logical. If `TRUE` (the default) and `observed` is supplied, each temperature/group curve is capped at the longest exposure actually observed for that cell — shorter durations are still predicted (the curve there is well-behaved, rising to the upper asymptote), but the long tail is not extrapolated past the data (e.g. designs that test much shorter exposures at hotter temperatures). Prediction cells with no matching observed temperature/group combination are dropped. A no-op when `observed` is not supplied. Set `FALSE` for the full grid.
`facet_scales`	Facet scale behaviour for grouped fits. `"auto"` (default) frees the duration (x) axis per group when `clip_to_observed = TRUE` and `observed` is supplied, so each group fills its own tested range (survival stays shared at `[0, 1]`); otherwise fixed. Or pass any `facet_wrap()` scale (`"fixed"`, `"free"`, `"free_x"`, `"free_y"`).

Details

Colour mapping uses the viridis palette by default (yellow at the highest temperature, dark purple at the lowest); the exposure-duration axis is linear by default, with the classical log-time TDT view available via `log_time = TRUE`.

Value

A ggplot object.

Examples

```
## Not run:
pred <- predict_survival_curves(wf, temps = c(30, 32, 34, 36))
plot_survival_curves(pred, observed = wf$data)
plot_survival_curves(pred, log_time = TRUE)  # classical TDT view

## End(Not run)
```

`plot_tdt_curve` *Plot a posterior LT_x curve*

Description

Plots posterior median +/- 95% CrI of the duration to reach `target_surv` across temperature. By default, returns a two-panel figure: a linear-time panel (the absolute time to reach the target) alongside a `log10`-time panel (the classical TDT view where the relationship is near-linear with slope `-1/z`). Both panels share posterior draws.

Usage

```
plot_tdt_curve(
  ltx,
  panels = c("both", "linear", "log"),
  colour = "#146C7C",
  observed = NULL,
  clip_to_observed = TRUE
)
```

Arguments

`ltx`	Output of `derive_tdt_curve()`.
`panels`	One of `"both"` (default), `"linear"`, or `"log"`.
`colour`	Line/ribbon colour. Default `"#146C7C"`.
`observed`	Optional standardised data tibble (from `standardize_data()`). Only used, together with `clip_to_observed`, to trim the temperature axis.

clip_to_observed

Logical. If TRUE (the default) and observed is supplied, each group's curve is drawn only over its observed assay-temperature range (the temperature grid is chosen by derive_tdt_curve(), so this only trims a grid drawn wider than the data). A no-op without observed.

Details

Grouped output from derive_tdt_curve() with by = ... is faceted automatically so each moderator level gets its own TDT curve panel.

Internal y-axis behaviour: if the 95% credible interval of the LT_x curve extends above 48 hours (converted to ltx$output_time_unit), both panels' y-axes are clamped to 48 hours so the readable part of the curve isn't compressed by an extreme upper tail at cool temperatures. If the data already stay below 48 hours, no limits are applied and ggplot2 chooses a data-driven range. Callers can override either way by appending their own + ggplot2::coord_cartesian(ylim = ...) to the returned plot.

Value

A ggplot object (single panel) or patchwork composition (two panels).

Examples

```
## Not run:
ltx <- derive_tdt_curve(wf, temp_grid = seq(29, 37, by = 0.5))
plot_tdt_curve(ltx)                  # two-panel default
plot_tdt_curve(ltx, panels = "log")  # classical TDT view only

# Override the internal 48-h cap by appending a coord_cartesian:
plot_tdt_curve(ltx) +
  ggplot2::coord_cartesian(ylim = c(0, 240))

## End(Not run)
```

---

plot_tdt_landscape *Plot the TDT survival landscape as a heatmap*

Description

Exposure-duration axis is linear by default; pass log_time = TRUE for the classical log-time TDT presentation.

Usage

```
plot_tdt_landscape(
  landscape,
  observed = NULL,
  contours = c(0.25, 0.5, 0.75),
```

```
  log_time = FALSE,
  clip_to_observed = TRUE,
  facet_scales = c("auto", "fixed", "free", "free_x", "free_y")
)
```

Arguments

landscape	Output of derive_tdt_landscape().
observed	Optional standardised data tibble to overlay.
contours	Numeric vector of survival levels to draw contours at. Default c(0.25, 0.5, 0.75).
log_time	Logical. If TRUE, exposure-duration y-axis is log10. Default FALSE (linear).
clip_to_observed	Logical. If TRUE (the default) and observed is supplied, each group's heatmap is restricted to the observed temperature range and capped at the longest observed exposure (shorter durations are still shown), so the surface is not drawn beyond the data; grouped panels use free axes to avoid plotting unsupported blank space. A no-op when observed is not supplied. Set FALSE for the full grid.
facet_scales	Facet scale behaviour for grouped landscapes. "auto" (default) uses free axes when clip_to_observed = TRUE and observed is supplied, otherwise fixed axes. Use "fixed" to force identical axes across groups, or any facet_wrap() scale option ("free", "free_x", "free_y").

Details

A grouped fit (a moderator on CTmax/z, e.g. oxygen or species) produces one heatmap panel per group, mirroring plot_survival_curves(); a single-condition fit returns the one ungrouped heatmap.

Value

A ggplot object.

Examples

```
## Not run:
lsp <- derive_tdt_landscape(wf)
plot_tdt_landscape(lsp, observed = wf$data)

## End(Not run)
```

plot_temperature_density

Plot a posterior temperature density (e.g. CTmax or T_crit)

Description

Density plot of a per-draw temperature posterior, with a horizontal segment marking the 95% CrI and a point at the median. Useful for visualising `derive_temperature_for_duration()` output.

Usage

```
plot_temperature_density(temp_post, truth = NULL, x_label = "Temperature (°C)")
```

Arguments

`temp_post`	Output of `derive_temperature_for_duration()`: a list with `$draws` (carrying a per-draw `temp` column) and `$summary`.
`truth`	Optional numeric scalar: a true value to mark with a dashed vertical line.
`x_label`	X-axis label. Default `"Temperature (°C)"`.

Value

A ggplot object.

Examples

```
## Not run:
tp <- derive_temperature_for_duration(wf, exposure_duration = 60,
                                      temp_grid = seq(36, 44, by = 0.1))
plot_temperature_density(tp)

## End(Not run)
```

plot_temperature_scenarios

Plot the three reference temperature scenarios

Description

One panel per scenario stacked vertically, with optional horizontal dashed line at the damage threshold `T_c`.

Usage

```
plot_temperature_scenarios(scens, T_c = NULL)
```

Arguments

scens	Named list from make_temperature_scenarios().
T_c	Optional damage threshold to mark.

Value

A ggplot object.

Examples

```
scens <- make_temperature_scenarios()
plot_temperature_scenarios(scens, T_c = 24)
```

---

predict_heat_injury *Predict heat injury and survival under a fluctuating temperature trace*

Description

Propagates the model's full posterior through an Eulerian damage- accumulation integration along the supplied temperature trace, returning the posterior median and 95% credible band of:

Usage

```
predict_heat_injury(
  trace,
  workflow,
  target_surv = "relative",
  T_c = NULL,
  trace_unit = "hours",
  ndraws = NULL,
  repair = FALSE,
  repair_pars = NULL,
  repair_scales_with_survival = TRUE,
  irreversible_mortality = TRUE,
  save_draws = FALSE,
  seed = NULL,
  by = NULL,
  shape = c("constant", "varying")
)
```

Arguments

trace	Tibble with columns time (numeric time from start, in trace_unit) and temp (°C), in time order. Requires >= 2 rows.
workflow	Fitted bayes_tls.

target_surv
: Threshold defining "1 dose". `"relative"` (default; `(low + up)/2`), `"absolute"` (= 0.5), or a numeric in `(0, 1)`. The default coincides with the classical LT50 when `low` is about 0 and `up` about 1; with sub-unit asymptotes the relative threshold is the more biologically meaningful anchor for dose accounting.

T_c
: Optional damage-accumulation threshold (°C). When supplied, the damage rate is forced to zero at `temp <= T_c` (matches the heat-injury integral, Equation 8 of the manuscript). Default `NULL` lets the rate fall naturally with T.

trace_unit
: Time unit of the trace's `time` column: one of `"hours"` (default), `"minutes"`, `"seconds"`, `"days"`. Reconciled internally with the model's fitted `duration_unit`, so the result is correct for any combination of model and trace time units.

ndraws
: Posterior draws to use; `NULL` (default) uses the full posterior, keeping the integral consistent with the model. Pass an integer to subsample – useful for speed, since the full posterior over a long temperature trace can be slow.

repair
: Logical. If `TRUE`, add Sharpe-Schoolfield repair. Default `FALSE`.

repair_pars
: Required when `repair = TRUE`. Named list with elements `TA, TAL, TAH, TL, TH, TREF, r_ref` passed straight to `repair_rate_schoolfield()`. `r_ref` should be in "doses per hour" so it matches the damage-rate units.

repair_scales_with_survival
: Logical. If `TRUE` (default), repair rate at each step is scaled by `survival / up` so dead organisms do not contribute to repair.

irreversible_mortality
: Logical. If `TRUE` (default), survival can only decrease over time — i.e. once the population's predicted survival reaches a value, it cannot rebound even if cumulative dose subsequently decreases.

save_draws
: Logical. If `TRUE`, return the full per-draw trajectories. Default `FALSE`.

seed
: Optional integer seeding the posterior-draw subsample for reproducibility. `NULL` (default) leaves the RNG untouched.

by
: Optional moderator column(s) for per-group injury. `NULL` (default) uses the fit's moderators; a single-condition fit returns the ungrouped result. A grouped fit runs the dose integral through each group's 4PL and `summary` gains the moderator column(s).

shape
: How the 4PL asymptotes/steepness enter the dose integral. `"constant"` (default) holds `low, up, k` at the centring temperature `T_bar` and lets only the midpoint shift with temperature – the classical constant-shape heat-injury assumption (via `hi_pars()`). `"varying"` feeds the temperature-local `low(T)`, `up(T)`, `k(T)` (and `mid(T)`) into both the damage rate and the dose→survival mapping (via `extract_4pl_pars()`). In `"varying"` mode survival is accumulated as a monotone state by an incremental decrement: each interval's mortality increment is the *local* curve's drop in survival over the dose added that interval (`g(D_j; shape_j) - g(D_{j-1}; shape_j)`), so the local shape sets the RATE of survival loss without re-mapping the global cumulative dose through a step-local curve (which would make survival jump when temperature changes with no added exposure). When the shape is flat in temperature the two modes coincide to numerical tolerance. See vignette-free note `notes/2026-06-26-shape-varying-heat-injury.qmd`.

Details

- **HI(t)** — cumulative heat injury, in percent of an LT_target_surv dose. When `HI(t) = 100`, the population has accumulated one full dose at the chosen survival threshold (default 50% mortality).
- **S(t)** — predicted survival fraction, mapped from the cumulative dose through the fitted 4PL.

Optionally adds a temperature-dependent repair rate via `repair_rate_schoolfield()`.

Value

A list with elements:

- `summary`: tibble with `time`, `temp`, and posterior median + 95% CrI for `hi`, `survival`, and `mortality` at each time step (plus the moderator column(s) for a grouped fit).
- `draws`: optional per-draw trajectories (when `save_draws = TRUE`).
- `meta`: list of inputs used.

Examples

```
## Not run:
scens <- make_temperature_scenarios()
hi    <- predict_heat_injury(scens$single_spike, wf)
hi$summary

## End(Not run)
```

---

predict_survival_curves

Posterior survival curves on a temperature × duration grid

Description

Predicts survival probability at every combination of `temps` × `durations` under a fitted 4PL, then returns the posterior median and 95% credible interval at each grid point. Random effects are marginalised out.

Usage

```
predict_survival_curves(
  workflow,
  temps = NULL,
  durations = NULL,
  ndraws = NULL,
  probs = c(0.025, 0.5, 0.975),
  by = NULL
)
```

Arguments

workflow	A fitted bayes_tls.
temps	Numeric vector of temperatures (°C). Default: unique assay temperatures in the training data.
durations	Numeric vector of durations. Default: 250 log-spaced values spanning 0.2× to 5× the observed range.
ndraws	Posterior draws to use; NULL (default) uses the full posterior. Pass an integer to subsample for speed.
probs	Numeric length-3 quantile probabilities (lower, median, upper). Default c(0.025, 0.5, 0.975).
by	Optional moderator column(s) for per-group curves. NULL (default) uses the fit's moderators; a single-condition fit returns the ungrouped curves, a grouped fit one block per group with the moderator column(s) prepended to summary.

Details

Use this for *curves* (a handful of temperatures, dense in duration). For a 2-D survival heatmap call derive_tdt_landscape() instead.

Value

A list with elements summary (tibble of temp, duration, survival_median, survival_lower, survival_upper; plus the moderator column(s) for a grouped fit), draws_matrix (the raw posterior matrix [ndraws x grid points] of survival probabilities), and grid (the prediction grid tibble from new_tdt_grid()). Use get_surv_draws() for the long-tibble form of the draws.

Examples

```
## Not run:
wf <- fit_4pl(d, ...)
pred <- predict_survival_curves(wf, temps = c(30, 32, 34))

## End(Not run)
```

---

print.bayes_tls *Compact print method for a* bayes_tls *workflow*

Description

One-screen header: data shape, asymptote/response bounds, parameterisation (midpoint, or direct CTmax/z with its threshold and t_ref; for grouped direct fits, the moderator(s)), random-effect grouping (if any), and fit status (draws if fitted, "spec only" otherwise). Call summary() for parameter posteriors and HMC diagnostics, plot() for MCMC trace plots.

Usage

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

Arguments

x	A bayes_tls object returned by fit_4pl().
...	Ignored.

Value

The object, invisibly.

Examples

```
## Not run:
wf <- fit_4pl(std)
print(wf)

## End(Not run)
```

---

repair_rate_schoolfield

Sharpe-Schoolfield thermal performance curve for repair rate

Description

Computes a temperature-dependent repair rate using the Sharpe-Schoolfield formulation (Sharpe & Schoolfield, 1981). All Arrhenius temperatures inside this function are in Kelvin; the user-facing temp_celsius argument is in degrees Celsius and converted internally.

Usage

```
repair_rate_schoolfield(temp_celsius, TA, TAL, TAH, TL, TH, TREF, r_ref)
```

Arguments

temp_celsius	Numeric vector of temperatures in °C.
TA	Arrhenius activation energy (K).
TAL	Low-temperature inactivation activation energy (K).
TAH	High-temperature inactivation activation energy (K).
TL	Low-temperature inactivation midpoint, in Kelvin.
TH	High-temperature inactivation midpoint, in Kelvin.
TREF	Reference temperature, in Kelvin.

`r_ref` The Arrhenius (uninhibited) rate scale, in the same units the user wants the output expressed in. Note this is NOT exactly the realised rate at `TREF`: the inactivation denominator suppresses it, so `rate(TREF) = r_ref / (1 + exp(TAL(1/TREF - 1/TL)) + exp(TAH(1/TH - 1/TREF)))` (a few % below `r_ref` when `TREF` sits well between `TL` and `TH`, more as it approaches either). To set the realised rate at `TREF` to a target value, divide your target by that denominator.

Details

The Sharpe-Schoolfield rate at temperature `T_K` (in Kelvin) is

$$r(T_K) = \frac{r_{ref} \cdot \exp\big(T_A(T_{ref}^{-1} - T_K^{-1})\big)}{1 + \exp\big(T_{AL}(T_K^{-1} - T_L^{-1})\big) + \exp\big(T_{AH}(T_H^{-1} - T_K^{-1})\big)}$$

where the numerator is the Arrhenius enzymatic rate at the reference temperature, and the two terms in the denominator suppress the rate at temperatures below `TL` and above `TH` respectively (the low- and high-temperature inactivation arms).

Value

Numeric vector of repair rates at the supplied temperatures, in the same units as `r_ref`. Negative or non-finite values are coerced to zero.

Examples

```
# Default shrimp-style parameters from the prototype: optimum ~17 °C
repair_rate_schoolfield(
  temp_celsius = seq(5, 30, by = 5),
  TA  = 14065, TAL = 50000, TAH = 120000,
  TL  = 10.5 + 273.15, TH = 22.5 + 273.15, TREF = 17 + 273.15,
  r_ref = 0.01
)
```

---

snowgum_psii *Snow gum leaf PSII functional-impairment thermal-tolerance data*

Description

Chlorophyll-fluorescence (F_v/F_m) measurements on excised snow gum (*Eucalyptus pauciflora*) leaf sections before and 16–24 h after heat exposure, from Experiment 1 (post-heat light vs dark recovery) of Arnold et al. (2026). Short branches were cut from six mature trees grown outdoors in Canberra, ACT; $\sim$1 cm^2 leaf sections were dark-adapted, given an initial F_v/F_m, then submerged in a temperature-controlled water bath under sub-saturating light across a grid of assay temperatures (30–56 degrees C) and exposure durations (5–120 min). After heat, paired arrays were held for 90 min in moderate light (`recovery = "Light"`) or in darkness (`recovery = "Dark"`); a final F_v/F_m was taken 16–24 h later. The response is the continuous proportion `fvfm_prop` (post/pre ratio), modelled with a Beta likelihood. The model-ready frame for the leaf PSII case study (sublethal, continuous-proportion endpoint). The light/dark contrast is a two-group categorical moderator; in the source experiment post-heat light lowered apparent heat tolerance.

Usage

```
snowgum_psii
```

Format

A data frame with 394 rows and 8 variables:

Temp Assay temperature (degrees C); 30, 35, 40, 44, 48, 52, 56.

Time Exposure duration (minutes); 5, 15, 30, 60, 120.

recovery Post-heat recovery light condition: `"Dark"` (darkness immediately after heat) or `"Light"` (90 min moderate light post-heat). A two-level moderator.

plant Replicate mature tree (factor, 6 levels); the natural random-effect grouping.

meas_day Assay day (factor, 2 levels). Two levels only, so a poor random-effect grouping; better treated as fixed or omitted.

initial_fvfm F_v/F_m measured before heat exposure.

final_fvfm F_v/F_m measured 16–24 h after heat exposure.

fvfm_prop Retained PSII function, `final_fvfm / initial_fvfm` (a proportion in the unit interval; 0 indicates complete loss of measurable PSII function).

Details

Two of the 396 raw rows have post/pre F_v/F_m marginally above 1 (both Dark, low dose) where a leaf measured slightly higher after heat than before; retained function cannot exceed 1, so these are treated as measurement noise and excluded, leaving 394 rows.

Source

Arnold PA, Harris RJ, Aitken SM, Hoek MM, Cook AM, Leigh A, Nicotra AB (2026) Towards a standard approach to investigating the thermal load sensitivity of photosystem II via chlorophyll fluorescence. bioRxiv doi:10.64898/2026.04.09.717599 (CC BY-NC 4.0), Experiment 1, snow gum slice. Raw file: `system.file("extdata", "data_function_PSII_TDT_snowgum.csv", package = "bayesTLS")`.

Examples

```
std <- standardize_data(snowgum_psii, temp = "Temp", duration = "Time",
                        proportion = "fvfm_prop",
                        random_effects = "plant",
                        duration_unit = "minutes")
```

`standardize_data` *Standardise a raw survival / proportion dataset for the TDT function library*

Description

Rewrites user column names into a single project-standard schema and attaches metadata used by every downstream fitting and prediction helper. This is the single entry point for raw data — everything else in the library assumes the output of this function.

Usage

```
standardize_data(
  data,
  temp,
  duration,
  n_total = NULL,
  n_surv = NULL,
  n_dead = NULL,
  survival = NULL,
  mortality = NULL,
  proportion = NULL,
  proportion_eps = 0.001,
  random_effects = NULL,
  duration_unit = "hours",
  temp_mean = NULL
)
```

Arguments

`data`	Raw data frame or tibble.
`temp`	Column name of the assay temperature (°C).
`duration`	Column name of the exposure duration. The unit is whatever is in the source data; record it via `duration_unit`.
`n_total`	Column name for total individuals per replicate. Required for count responses; leave `NULL` (default) for a continuous `proportion` response.
`n_surv`	Column name for survivor counts.
`n_dead`	Column name for death counts. Converted to `n_surv` via `n_surv = n_total - n_dead`.
`survival`	Column name for survival proportions in `[0, 1]`. Converted to integer counts via `n_total`.
`mortality`	Column name for mortality proportions in `[0, 1]`. Converted to `n_surv = round((1 - mortality) * n_total)`.

proportion	Column name for a continuous proportion response in [0, 1] with no denominator (modelled with a Beta likelihood). Mutually exclusive with the count arguments above.
proportion_eps	Half-open clamp applied to proportion so values sit strictly inside (0, 1) (the Beta density is undefined at exactly 0 or 1). Default 0.001.
random_effects	Optional character vector of grouping variables for random effects, e.g. c("Date", "Tank"). These columns are converted to factors and stored in metadata for the fitter to read.
duration_unit	Label for the unit of duration, stored in metadata. Default "hours".
temp_mean	Value to subtract from temp to form temp_c. NULL (default) uses mean(temp). Supply a fixed value to align multiple datasets to a common centre.

Details

Two response types are supported:

- **Count data** (binomial / beta-binomial): supply n_total plus **exactly one** of n_surv, n_dead, survival, or mortality. The other counts are derived and the standardised columns include n_total, n_surv, n_dead, survival. response_type is recorded as "count".
- **Continuous proportion** (Beta), e.g. a chlorophyll-fluorescence F_v/F_m ratio with no denominator: supply proportion and omit the count arguments. The value is stored in survival (clamped into the open interval (proportion_eps, 1 - proportion_eps) so the Beta likelihood is finite); no n_total/n_surv columns are created. response_type is recorded as "proportion".

If the dataset spans multiple categories (life stages, species, populations, etc.), filter to one category before calling this function and fit a separate model per subset — the fitter does not estimate category-level effects.

Value

A tibble with the standardised columns plus a "tdt_meta" attribute storing temp_mean, duration_unit, random_effects, response_type ("count" or "proportion"), response_var (the response column name for a proportion fit, else NULL), and proportion_eps (the clamp half-width used for a proportion fit, else NULL).

Examples

```
# Count data
raw <- data.frame(
  temperature_C = rep(c(30, 32, 34), each = 4),
  exposure_h    = rep(c(1, 2, 4, 8), times = 3),
  n             = 30L,
  alive         = c(29, 28, 25, 5, 30, 27, 18, 2, 28, 22, 10, 1)
)
standardize_data(raw,
                 temp     = "temperature_C",
                 duration = "exposure_h",
                 n_total  = "n",
```

```
                  n_surv   = "alive")

# Continuous proportion (Beta) data
raw_p <- data.frame(
  temperature_C = rep(c(30, 32, 34), each = 4),
  exposure_h    = rep(c(1, 2, 4, 8), times = 3),
  fvfm_ratio    = c(0.95, 0.9, 0.7, 0.2, 0.92, 0.6, 0.3, 0, 0.8, 0.4, 0.1, 0)
)
standardize_data(raw_p,
                 temp       = "temperature_C",
                 duration   = "exposure_h",
                 proportion = "fvfm_ratio")
```

`summarise_observed_survival` *Summarise observed survival counts to mean ± SE per (temp, duration) cell*

Description

Useful as an overlay on plotted posterior curves. Returns one row per temperature × duration combination, with the mean observed survival proportion across replicates and a standard error.

Usage

```
summarise_observed_survival(observed)
```

Arguments

`observed`	A tibble from `standardize_data()`.

Value

A tibble with columns `temp`, `duration`, `survival_mean`, `survival_se`, `survival_lower`, `survival_upper`, `n_units`, `n_total_sum`.

Examples

```
raw <- data.frame(T = rep(30, 6), hrs = rep(c(1, 5), each = 3),
                  n = 30, alive = c(28, 27, 29, 10, 12, 8))
d <- standardize_data(raw, "T", "hrs", n_total = "n", n_surv = "alive")
summarise_observed_survival(d)
```

summary.bayes_tls *Summarise a fitted* bayes_tls *workflow*

Description

Delegates to summary() on the underlying brmsfit (via get_brmsfit()), so the returned object is a brms brmssummary with the population-level coefficient table, group-level standard deviations, family parameters, and HMC diagnostics already laid out by brms.

Usage

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

Arguments

object	A fitted bayes_tls workflow.
...	Passed through to brms::summary.brmsfit() (e.g. prob, mc_se, priors, robust).

Details

The high-level workflow context (data shape, centred-temperature mean, asymptote bounds, random-effect grouping, draw count) is available via print.bayes_tls(). For natural-scale 4PL parameters (low, up, k, z), use tdt_parameter_table(). For the TDT quantities (z, CTmax_1hr, optionally T_crit), use tls().

Value

A brms brmssummary object (brms handles printing).

Examples

```
## Not run:
summary(wf)
summary(wf, prob = 0.9, robust = TRUE)

## End(Not run)
```

`tdt_parameter_table` *Posterior parameter table on the natural scale*

Description

Pulls the population-level posterior of the reparameterised 4PL parameters, transformed back to the natural scale via the model's bounds, as a one-row-per-parameter tibble with median + 95% CrI.

Usage

```
tdt_parameter_table(workflow, by = NULL)
```

Arguments

`workflow`	A fitted `bayes_tls`.
`by`	Optional character vector of moderator columns to report per group. `NULL` (default) uses the fit's moderators (`meta$group_vars`); a single-condition fit then returns one block (no group column).

Details

For the **midpoint** parameterisation the rows are `low`, `up`, `k`, the `mid` intercept and temperature slope, and `z = -1 / mid_temp`. For the **direct** CTmax/z parameterisation the rows are `low`, `up`, `k`, `CTmax` (at the model's reference dose) and `z`. All quantities are read by evaluating `brms::posterior_linpred(nlpar=)` at `temp_c = 0` and `temp_c = 1` (so the result is parameterisation- and coding-agnostic – no coefficient-name parsing). For a fit whose CTmax/z vary by a moderator the table is returned per group, with the moderator column(s) prepended.

Value

A tibble with columns `parameter`, `median`, `lower`, `upper` (plus the moderator column(s) for a grouped fit).

Examples

```
## Not run:
tdt_parameter_table(wf)

## End(Not run)
```

tdt_quantile *Quantile wrapper with TDT-friendly defaults*

Description

Quantile wrapper with TDT-friendly defaults

Usage

```
tdt_quantile(x, probs = c(0.025, 0.5, 0.975))
```

Arguments

x	Numeric vector.
probs	Numeric vector of quantile probabilities.

Value

Numeric vector of length `length(probs)`.

Examples

```
tdt_quantile(rnorm(100))
```

theme_tdt *Project ggplot theme*

Description

A minimal classic theme with bold axis titles, used by all plotting helpers in this library. Caller can override with `+ theme(...)`.

Usage

```
theme_tdt(base_size = 13)
```

Arguments

base_size	Base font size, passed to `theme_classic()`. Default 13.

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(mpg, hp)) + geom_point() + theme_tdt()
```

`tls` *Thermal load sensitivity summaries from any fitted 4PL model*

Description

One call that derives the classical TDT quantities — thermal sensitivity `z`, `CTmax`, and (for lethal endpoints) `T_crit` — **per moderator group** from a fitted joint 4PL, including hand-written `brms` models with moderators (sex, life stage, clone, ...) on any sub-parameter. The four sub-parameters are evaluated at a moderator x temperature grid with `brms::posterior_linpred()`, then `z`, `CTmax` and `T_crit` are derived from the same posterior draws (so they are mutually consistent and `T_crit` pairs `CTmax` and `z` per draw).

Usage

```
tls(
  object,
  by = NULL,
  params = "all",
  target_surv = "relative",
  t_ref = NULL,
  time_multiplier = NULL,
  lethal = FALSE,
  TC_rate_range = c(0.1, 1),
  temp = "temp_c",
  temp_mean = NULL,
  temp_grid = NULL,
  re_formula = NA,
  lower = 0,
  upper = 1,
  nlpars = c("lowraw", "upraw", "logk", "mid"),
  ndraws = NULL,
  newdata = NULL,
  probs = c(0.025, 0.5, 0.975),
  seed = NULL,
  mode = NULL,
  p = NULL
)

tls_z(object, ...)

tls_ctmax(object, ...)

tls_tcrit(object, ...)
```

Arguments

`object`	A fitted `bayes_tls` workflow or a `brmsfit` 4PL whose non-linear parameters are the (reparameterised) asymptotes, steepness and midpoint.

by	Character vector of moderator columns defining the groups reported separately (e.g. "sex"). NULL (default) pools to a single group.
params	Quantities to return: "all" (z, CTmax, and T_crit when lethal = TRUE), or a subset of c("z", "ctmax", "tcrit").
target_surv	Survival threshold the LT curve is read at: "relative" (default; the per-draw midpoint (low + up)/2), "absolute" (literal LT50), or a numeric p in (0, 1) for an absolute LTp.
t_ref	Reference exposure duration for CTmax, in minutes. If NULL (default), inherit the fit's own reference exposure (meta$t_ref) and report it via a message; a raw brmsfit with no recorded reference falls back to 60. Converted to the model's time scale via time_multiplier.
time_multiplier	Multiplier from the model's time unit to minutes. NULL (default) derives it from a bayes_tls workflow's duration_unit (e.g. 60 for an hours model); a raw brmsfit with no metadata defaults to 1, so t_ref is then in the model's own time units.
lethal	Logical; with TRUE, T_crit (rate-multiplier) is available. T_crit is meaningful only for damage-accumulation (lethal) endpoints.
TC_rate_range	Length-2 HI-rate floor range (% per hour) for T_crit.
temp	Name of the centred temperature column. Default "temp_c".
temp_mean	Centring constant mapping temp back to temperature, needed for CTmax/T_crit. Taken from a bayes_tls workflow's metadata when available; supply it for a raw brmsfit.
temp_grid	Temperatures (on the temp scale) at which the LT curve is evaluated: z is the local z(T) pooled over this grid and CTmax is the per-draw crossing of t_ref inverted on it. Default: 11 points over the observed range. For an absolute threshold, ensure the grid brackets the CTmax crossing (extend it if CTmax falls outside the observed range).
re_formula	Passed to brms::posterior_linpred(); default NA (population level). Include group-level terms (e.g. clone) for per-group draws.
lower, upper	Response bounds of the disjoint-bounds reparameterisation.
nlpars	Non-linear parameter names, ordered low / up / k / mid (raw scale).
ndraws	Optional number of posterior draws to subsample.
newdata	Optional explicit moderator x temperature grid; overrides the by/temp_grid construction.
probs	Summary quantiles (lower, median, upper). Default c(.025, .5, .975).
seed	Optional integer seeding the draw subsample (ndraws) and the T_crit rate draws for reproducibility. NULL (default) leaves the RNG alone.
mode, p	**Deprecated**, superseded by target_surv. If supplied, mode = "relative" maps to target_surv = "relative" and mode = "absolute" (with p) to target_surv = p. Kept for back-compat.
...	Additional arguments passed on to tls() (used by the tls_z(), tls_ctmax() and tls_tcrit() convenience wrappers).

Details

tls() derives z, CTmax and T_crit from the same posterior draws, so on a common temperature grid they are mutually consistent for every threshold. z is the central-difference local z(T) (negative reciprocal of the local slope of log10(LT) on temperature) pooled over temp_grid; CTmax is the per-draw temperature at which the fitted LT curve crosses t_ref, from the exact closed form when the relative midpoint is linear and from a true per-draw inversion of the (possibly bent) curve otherwise. Under target_surv = "relative" (the default) log10(LT) = mid(T) is linear, so z is constant in temperature and equals -1 / b_mid_temp_c; under an absolute threshold with temperature effects on the asymptotes or steepness the curve bends and both the local z(T) and the inversion account for it, with no linear approximation.

Value

A tls object: $summary (per-group, per-quantity median + interval), $draws (per-group, per-quantity posterior draws), and $meta.

Examples

```
## Not run:
tls(joint_sex_fit, by = "sex", lethal = TRUE, temp_mean = 36.1)  # z, CTmax, T_crit per sex
tls(wf_leaf, params = "z")                                        # z only, workflow

## End(Not run)
```

---

ts_ci *Uncertainty for the classical two-stage TDT quantities*

Description

Two propagation methods on the Stage-2 fit:

- "delta" — delta-method standard errors for z and CTmax, with **both Normal and t quantiles** (the t-quantile is the small-sample correction for the few Stage-2 residual degrees of freedom). This is the method the bias simulation reports.
- "mvn" — slope-CI inversion for z (defined only when the slope CI is wholly negative) plus MVN simulation of the Stage-2 coefficients for CTmax and T_crit, and a predict.lm confidence band for the LT-vs-T line over temp_grid. This is the method the case studies report.

Usage

```
ts_ci(
  stage2,
  method = c("delta", "mvn"),
  level = 0.95,
  t_ref = 60,
  time_multiplier = 1,
  TC_rate_range = c(0.1, 1),
```

```
  temp_grid = NULL,
  n_sim = 1000,
  seed = 123
)
```

Arguments

stage2	Output of ts_stage2().
method	"delta" or "mvn".
level	Confidence level. Default 0.95.
t_ref, time_multiplier, TC_rate_range	As in ts_stage2().
temp_grid	Temperatures for the line band ("mvn" only).
n_sim	MVN draws ("mvn" only). Default 1000.
seed	RNG seed ("mvn" only). Default 123.

Value

For "delta", a list with z and CTmax_1hr, each list(point, lower, upper, lower_t, upper_t, se), plus df_resid. As in ts_stage2(), the CTmax_1hr block reports CTmax at t_ref despite its name (the two coincide at the default t_ref = 60). For "mvn", a list with summary_ci (z/CTmax/T_crit bounds) and curve_ci (per-temp_grid line band).

Examples

```
d <- data.frame(
  temp = rep(c(30, 32, 34, 36, 38), each = 12),
  dur  = rep(c(1, 5, 15, 45, 135, 405), times = 10),
  surv = rbinom(60, 20, 0.4), tot = 20)
s2 <- ts_stage2(ts_stage1(d, "temp", "dur", "surv", "tot"))
ts_ci(s2, method = "delta")$z
```

---

ts_curve *Median LT-vs-temperature line from a two-stage fit*

Description

Median LT-vs-temperature line from a two-stage fit

Usage

```
ts_curve(stage2, temp_grid, time_multiplier = 1)
```

Arguments

stage2	Output of ts_stage2().
temp_grid	Temperatures (°C) to evaluate.
time_multiplier	Multiplier to minutes. Default 1.

Value

A tibble with temp and duration_median (minutes).

Examples

```
d <- data.frame(
  temp = rep(c(30, 34, 38), each = 12),
  dur  = rep(c(1, 5, 15, 45), times = 9),
  surv = rbinom(36, 20, 0.4), tot = 20)
ts_curve(ts_stage2(ts_stage1(d, "temp", "dur", "surv", "tot")),
         temp_grid = seq(30, 38, 1))
```

---

ts_stage1 *Stage 1 of the classical two-stage TDT pipeline*

Description

Fits a separate logistic dose-response curve at each assay temperature and reads off log10(LT50) (the duration at 50% survival). The binomial family uses stats::glm; the beta-binomial family uses glmmTMB::glmmTMB with a betabinomial family (overdispersion at Stage 1).

Usage

```
ts_stage1(
  data,
  temp = "temp",
  duration = "duration",
  n_surv = "n_surv",
  n_total = "n_total",
  family = c("binomial", "betabinomial")
)
```

Arguments

data	Data frame with one row per (temperature, duration) replicate.
temp, duration, n_surv, n_total	Column names (strings) for assay temperature (°C), exposure duration, survivors, and trials.
family	"binomial" or "betabinomial".

Details

Two validity flags are returned so callers can choose their own success rule: `finite_ok` (finite coefficients, negative non-trivial slope) and `bracket_ok` (the fitted LT50 lies within the observed duration range, padded by 0.5 on the log10 scale). `stage1_ok` is their conjunction.

Value

A tibble with one row per temperature: `temp`, `log10_lt50`, `se_log10_lt50`, `slope`, `phi` (beta-binomial precision, else `NA`), `finite_ok`, `bracket_ok`, `stage1_ok`.

Examples

```
d <- data.frame(
  temp = rep(c(30, 34, 38), each = 12),
  dur  = rep(rep(c(1, 5, 15, 45), 3), times = 3),
  surv = rbinom(36, 20, 0.5), tot = 20)
ts_stage1(d, "temp", "dur", "surv", "tot", family = "binomial")
```

---

ts_stage2 *Stage 2 of the classical two-stage TDT pipeline*

Description

Regresses Stage-1 `log10(LT50)` on assay temperature by ordinary least squares and derives the classical quantities. `z = -1/slope`; `CTmax(t_ref) = (log10(t_ref) - intercept) / slope`; `T_crit` follows the rate-multiplier definition, `CTmax(1h) + z * mean(log10(TC_rate_range/100))`.

Usage

```
ts_stage2(
  stage1,
  t_ref = 60,
  time_multiplier = 1,
  TC_rate_range = c(0.1, 1),
  rows = c("stage1_ok", "finite_ok")
)
```

Arguments

`stage1`	Output of `ts_stage1()`.
`t_ref`	Reference exposure duration for CTmax (minutes). Default 60. Must be > 0.
`time_multiplier`	Multiplier from the Stage-1 duration unit to minutes (e.g. 60 if durations are in hours). Default 1. Must be > 0.
`TC_rate_range`	Length-2 HI-rate range (% per hour) for T_crit.
`rows`	Which Stage-1 rows to keep: `"stage1_ok"` (bracketing validation, the case-study default) or `"finite_ok"` (finite/negative slope only, the simulation's looser rule).

Details

The Stage-2 regression is **unweighted** ordinary least squares: it ignores the Stage-1 standard errors (`se_log10_lt50`) by design. This is the generated-regressor bias the manuscript demonstrates, so `se_log10_lt50` is exposed for diagnostics only and is not propagated into the Stage-2 quantities.

Value

`list(fit, summary)`; `fit` is `NULL` if fewer than 3 valid Stage-1 estimates remain. `summary` has `intercept`, `slope_T`, `z`, `CTmax_1hr`, `T_crit`, `r_squared`, `n_stage1`, `n_excluded`. Note that despite its name `CTmax_1hr` reports CTmax at `t_ref` (the two coincide at the default `t_ref = 60`); the 1 h anchor used for `T_crit` is computed separately and is invariant to `t_ref`.

Examples

```
d <- data.frame(
  temp = rep(c(30, 32, 34, 36, 38), each = 12),
  dur  = rep(c(1, 5, 15, 45, 135, 405), times = 10),
  surv = rbinom(60, 20, 0.4), tot = 20)
s1 <- ts_stage1(d, "temp", "dur", "surv", "tot")
ts_stage2(s1)$summary
```

---

zebrafish_o2 *Zebrafish lethal-TDT data across an oxygen gradient*

Description

Survival of zebrafish (*Danio rerio*) larvae assayed for upper thermal tolerance under three oxygen treatments, the model-ready frame for the oxygen-gradient case study. Diploid and triploid larvae were held at assay temperatures of 26 (control), 38, 39 and 40 degrees C for 3.8–240 minutes under hypoxia, normoxia or hyperoxia, and scored alive/dead. One row per assay group; `oxygen` is the categorical moderator. Fit `CTmax` and `z` as functions of `oxygen` (optionally `ploidy`) in one joint 4PL to compare thermal tolerance across the gradient with full posterior uncertainty.

Usage

```
zebrafish_o2
```

Format

A data frame with 905 rows and 10 variables:

cohort Larval cohort identifier.

ploidy Ploidy, a factor with levels `diploid`, `triploid`.

oxygen Oxygen treatment, a factor with levels `hypoxia`, `normoxia`, `hyperoxia` (the modelling moderator).

o2_nominal Nominal oxygen target as percent air saturation (25 / 100 / 225).

o2_measured Measured oxygen level (percent air saturation).

temp Target assay temperature (degrees C).

temp_measured Measured assay temperature (degrees C).

duration_min Exposure duration (minutes).

n_total Number of larvae in the assay group.

n_surv Number surviving after exposure and recovery.

Source

Saruhashi S, Boerrigter JGJ, Hooymans MHL, Sinclair BJ, Verberk WCEP (2026). Data and code for: Oxygen availability and oxygen delivery but not oxidative stress shape heat tolerance in diploid and triploid zebrafish larvae. Zenodo, doi:10.5281/zenodo.20075355 (distributed under CC BY 4.0). Associated article: *Journal of Experimental Biology* 229(10): jeb251548, doi:10.1242/jeb.251548. Raw file: `system.file("extdata", "data_lethal_TDT_zebrafish_oxygen.csv", package = "bayesTLS")`.

Examples

```
## Not run:
std <- standardize_data(zebrafish_o2, temp = "temp", duration = "duration_min",
                        n_total = "n_total", n_surv = "n_surv",
                        duration_unit = "minutes")
wf <- fit_4pl(std, ctmax = ~ 0 + oxygen, z = ~ 0 + oxygen, t_ref = 60)
tls(wf, by = "oxygen", lethal = TRUE)   # z, CTmax, T_crit per oxygen treatment

## End(Not run)
```

Index

∗ **datasets**
  aphid_tdt, 3
  dsuzukii, 12
  snowgum_psii, 42
  zebrafish_o2, 56

aphid_tdt, 3

bayes_R2_tls, 4
brms::bayes_R2(), *4*, *19*
brms::brm(), *15–17*
brms::brmsfit, *19*
brms::mcmc_plot(), *19*
brms::posterior_linpred(), *14*, *50*, *51*
brms::summary.brmsfit(), *47*

clock_to_minutes, 5
compute_4pl_bounds(), *25*

derive_tdt_curve, 5
derive_tdt_curve(), *9*, *33*, *34*
derive_tdt_landscape, 7
derive_tdt_landscape(), *35*, *40*
derive_temperature_for_duration, 8
derive_temperature_for_duration(), *36*
derive_z, 10
diagnose_tdt_fit, 11
diagnose_tdt_fit(), *4*
dsuzukii, 12

extract_4pl_pars, 13
extract_4pl_pars(), *18*, *19*, *38*

fit_4pl, 15
fit_4pl(), *4*, *18*, *19*, *22*, *24*, *26*, *41*
format_interval, 17

get_4pl_est, 18
get_4pl_est(), *14*
get_brmsfit, 19
get_brmsfit(), *4*, *30*, *47*
get_hi_draws, 20
get_surv_draws, 20
get_surv_draws(), *40*
get_tls_est, 21
get_tls_est(), *18*, *19*
glmmTMB::glmmTMB, *54*

has_fit, 22
has_fit(), *19*
hi_pars(), *14*, *38*

make_4pl_formula, 22
make_4pl_formula(), *15*, *16*, *25*, *26*
make_4pl_priors, 25
make_4pl_priors(), *15*, *16*
make_temperature_scenarios, 27
make_temperature_scenarios(), *29*, *37*

new_tdt_grid(), *40*

planted_dose_from_trace, 29
plot(), *40*
plot.bayes_tls, 30
plot.bayes_tls(), *17*
plot_heat_injury, 30
plot_repair_rate, 31
plot_survival_curves, 32
plot_survival_curves(), *35*
plot_tdt_curve, 33
plot_tdt_landscape, 34
plot_tdt_landscape(), *7*, *8*
plot_temperature_density, 36
plot_temperature_scenarios, 36
posterior::as_draws_df(), *19*
predict_heat_injury, 37
predict_heat_injury(), *14*, *20*, *21*, *29*, *31*
predict_survival_curves, 39
predict_survival_curves(), *7*, *8*, *20*, *21*, *32*
print.bayes_tls, 40

print.bayes_tls(), *17*, *47*

repair_rate_schoolfield, 41
repair_rate_schoolfield(), *38, 39*

snowgum_psii, 42
standardize_data, 44
standardize_data(), *16*, *23*, *25*, *32, 33*, *46*
stats::glm, *54*
summarise_observed_survival, 46
summary(), *40*
summary.bayes_tls, 47
summary.bayes_tls(), *17*, *30*

tdt_parameter_table, 48
tdt_parameter_table(), *18, 19*, *47*
tdt_quantile, 49
theme_tdt, 49
tls, 50
tls(), *9*, *15*, *17, 18*, *21*, *24*, *47*, *51*
tls_ctmax *(*tls*)*, 50
tls_tcrit *(*tls*)*, 50
tls_z *(*tls*)*, 50
ts_ci, 52
ts_curve, 53
ts_stage1, 54
ts_stage1(), *55*
ts_stage2, 55
ts_stage2(), *53, 54*

zebrafish_o2, 56