Package 'freqTLS'

July 21, 2026

Title Frequentist Inference for Thermal Load Sensitivity Models

Version 0.1.0

Description A maximum-likelihood implementation of the thermal-load-sensitivity framework for thermal death-time modelling introduced by Noble, Arnold and Pottier in the 'bayesTLS' package, providing the frequentist counterpart to that Bayesian workflow. The modelling idea and the four-parameter logistic parameterisation are theirs; 'freqTLS' contributes a 'Template Model Builder' ('TMB') likelihood whose midpoint is written directly in terms of critical thermal maximum ('CTmax') and
thermal sensitivity (z) so both headline quantities are estimable, and reports uncertainty through a unified trio of frequentist intervals -- Wald (delta), profile-likelihood, and bootstrap -- for binomial and beta-binomial survival counts and beta-distributed proportions. Column and formula interfaces support fixed and grouped designs plus limited independent random intercepts; prediction includes survival curves and deterministic heat-injury scenarios. Equivalence claims are restricted to the matched relative-threshold, constant-shape 'bayesTLS' configuration.

License GPL (>= 3)

Copyright See file inst/COPYRIGHTS.

URL https://github.com/itchyshin/freqTLS,
https://itchyshin.github.io/freqTLS/

BugReports https://github.com/itchyshin/freqTLS/issues

Encoding UTF-8

Depends R (>= 4.2)

Imports cli, ggplot2, MASS, parallel, rlang, stats, tibble, TMB, utils

LinkingTo RcppEigen, TMB

Suggests glmmTMB, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Shinichi Nakagawa [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-7765-5182>), Pieter A. Arnold [aut] (ORCID: <https://orcid.org/0000-0002-6158-7752>, co-author of the bayesTLS framework), Patrice Pottier [aut] (ORCID: <https://orcid.org/0000-0003-2106-6597>, co-author of the bayesTLS framework), Daniel W. A. Noble [aut] (ORCID: <https://orcid.org/0000-0001-9460-8743>, senior author of the bayesTLS thermal-load-sensitivity framework)

Maintainer Shinichi Nakagawa <itchyshin@gmail.com>

Repository CRAN

Date/Publication 2026-07-21 11:00:10 UTC

Contents

`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 profile-likelihood confidence intervals on those direct parameters.

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 = "freqTLS")`.

Examples

```
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
```

---

check_tls *Report identifiability diagnostics for a fitted model*

Description

`check_tls()` re-runs the data-adequacy diagnostics on a fitted `profile_tls` object, including the two post-fit checks that `fit_tls()` cannot run before the model exists: whether any fitted `CTmax` is extrapolated beyond the assayed temperatures (item 7), and whether `phi` has reached the binomial limit (item 8). Each concern is emitted as a `cli::cli_warn()`. Use it to audit a fit, or after `suppressWarnings()` around `fit_tls()`.

Usage

```
check_tls(fit)
```

Arguments

`fit`	A `profile_tls` fit from `fit_tls()`, or a `freq_tls` workflow from `fit_4pl()`.

Details

The profile-geometry diagnostics (items 9-12) are emitted by `confint()` and `profile()` when those are called, not here, because they require the profile likelihood.

Value

Invisibly, a character vector of the diagnostic codes that fired.

Recovery guide

The warning code returned by `check_tls()` identifies the next action:

- `temps`: assay at least three distinct temperatures spanning the survival transition.
- `durations` / `durations_per_temp`: assay at least three distinct durations per temperature, with times on both sides of the transition.
- `no_mortality`: extend to hotter or longer exposures until mortality occurs.
- `all_mortality`: add cooler or shorter exposures that retain survivors.
- `threshold`: extend the design until observed survival straddles 0.5.
- `up_not_approached` / `low_not_approached`: add milder / harsher conditions approaching survival 1 / 0, or do not interpret that asymptote.
- `ctmax_extrapolated`: expand the assayed temperature range to bracket CTmax; otherwise report it explicitly as extrapolated.
- `phi_binomial_limit`: consider the simpler binomial family.

After changing the design or family, refit and rerun `check_tls()`. For an existing data set that cannot be augmented, use the warning to limit the scientific claim; `vignette("profile-likelihood")` explains the strict `fallback = FALSE` diagnostic and the default bootstrap recovery attempt.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
check_tls(fit)
```

---

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
```

`confint.profile_tls` *Confidence intervals for a fitted thermal-load-sensitivity model*

Description

`confint()` returns confidence intervals for the natural-scale parameters of a `fit_tls()` model. Three methods are available:

Usage

```
## S3 method for class 'profile_tls'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = c("profile", "wald", "bootstrap"),
  npoints = 30L,
  trace = FALSE,
  fallback = TRUE,
  nboot = 1000L,
  boot_seed = NULL,
  cores = 1L,
  ...
)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`parm`	Character vector of target names (for example `"CTmax"`, `"z"`, `"log_z"`, `"low"`, `"k"`, `"phi"`, grouped names such as `"CTmax:A"`, or contrasts such as `"dCTmax:A-B"`, which means group A minus group B). `NULL` (default) returns intervals for the natural-scale parameters of the fit.
`level`	Confidence level (default `0.95`).
`method`	One of `"profile"` (default), `"wald"`, or `"bootstrap"`.
`npoints`	Number of grid points used per profile (default `30`); ignored for `method = "wald"` and `method = "bootstrap"`.
`trace`	Logical; print inner-optimisation progress (profile/bootstrap).
`fallback`	Logical; when `method = "profile"` (the default), fall back to the parametric bootstrap for any parameter whose profile does not close, and for all parameters when the fit's Hessian is not positive definite (`pdHess = FALSE`). Default `TRUE`.

nboot	Number of bootstrap replicates for method = "bootstrap" or the fallback (default 1000).
boot_seed	Optional integer seed making the bootstrap reproducible without disturbing the caller's random stream (default NULL).
cores	Number of CPU cores for the bootstrap refits (default 1, maximum 2). Requests above two warn and use two. cores > 1 refits replicates in parallel by forking (Unix; sequential on Windows). Results are identical for a given boot_seed regardless of cores.
...	Reserved; must be empty.

Details

- method = "profile" (default) computes profile-likelihood confidence intervals by inverting the likelihood-ratio test: the interval is {psi : D(psi) <= qt(1 - alpha/2, df)^2}, found by stats::uniroot() on each side of the MLE on the unconstrained internal coordinate, with the endpoints transformed to the natural scale. The cutoff is the squared profile-t quantile on df = n - p residual degrees of freedom (Bates-Watts profile-t), not qchisq(level, 1); the two coincide as df -> Inf. These intervals are prior-free and respect asymmetry. They are equivariant under monotone reparameterisation, so the z interval equals exp() of the internal log_z interval.
- method = "wald" reuses the Phase-2 Wald path: estimate +/- t * se (with t = qt(1 - alpha/2, df) on df = n - p) on the internal (link) scale, back-transformed.
- method = "bootstrap" returns prior-free parametric-bootstrap percentile intervals: survival counts are regenerated at the observed design from the fitted 4PL, the model is refitted nboot times, and the interval is the percentile range of the replicate estimates. This is the likelihood-path analogue of the bayesTLS posterior interval. It returns a finite interval only when enough stable, non-degenerate refits remain.

When a profile does not close on one side, or the fitted Hessian is not positive definite (pdHess = FALSE), confint() falls back to the parametric bootstrap for the affected parameters (with a message). The fallback can still return NA when too few valid refits remain. Set fallback = FALSE to keep the strict profile behaviour, which returns NA on the open side (never a fabricated bound) with a warning that the parameter is weakly identified (see vignette("profile-likelihood")). The upper asymptote up has its own coordinate beta_up under disjoint bounds but is not yet profiled, so it is reported with the delta-method Wald interval under the profile/Wald methods, with a message.

For a fit with a random intercept (CTmax ~ <fixed> + (1 | group)), method = "profile" profiles the fixed-effect coordinates by re-running the Laplace approximation at each grid point, which is slower than a fixed-effects profile. Variance components keep their log-scale Wald intervals under the profile method, and a non-closing random-effects profile falls back to Wald. method = "bootstrap" instead redraws every active random-intercept block and refits with the Laplace approximation, returning percentile intervals when enough stable refits remain.

Value

A tibble with one row per target and columns parameter, conf.low, conf.high, estimate, level, method, scale, and conf.status.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
confint(fit, "CTmax", method = "profile")
confint(fit, "z", method = "profile")
```

`derive_ctmax` *Derive the temperature giving a target survival at a fixed exposure*

Description

`derive_ctmax()` inverts the fitted 4PL for **temperature**: it returns the assay temperature at which survival equals a target `surv` after exposure `duration`. By default `surv` is the *relative* midpoint threshold `(low + up) / 2` and `duration` is `tref`, so `derive_ctmax(fit)` reproduces the fitted `CTmax`. Supplying an **absolute** `surv` gives the absolute- threshold critical temperature (the analogue of the `bayesTLS` `extract_tdt()` absolute mode), with the asymmetry correction `qlogis((surv - low) / (up - low)) / k` built in.

Usage

```
derive_ctmax(object, surv = NULL, duration = NULL, group = NULL)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`surv`	Target survival probability in `(low, up)`. `NULL` (default) uses the relative midpoint `(low + up) / 2`, reproducing `CTmax` at `tref`.
`duration`	Exposure duration(s) (native time unit; strictly positive). Defaults to the fit's `tref`.
`group`	Optional single group level (grouped fits only).

Details

Solving `surv = low + (up - low) * plogis(-k (log10(duration) - mid))` with `mid = log10(tref) - (temp - CTmax) / z` for the temperature gives

$$temp = CTmax - z\Big(\log_{10} duration - \log_{10} t_{ref} + \mathrm{qlogis}\big(\tfrac{surv-low}{up-low}\big)/k\Big).$$

The target `surv` must lie strictly between `low` and `up`. For a random-effects fit this is a population-level derived quantity; it does not add a group BLUP.

Value

A numeric vector of temperatures (degrees C), one per `duration`.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
derive_ctmax(fit)                                  # ~ CTmax (relative, at tref)
derive_ctmax(fit, surv = 0.5, duration = c(1, 4))  # absolute 50% survival
```

`derive_lt` *Derive the lethal / survival-threshold duration at a temperature*

Description

`derive_lt()` solves the fitted 4PL for the duration at which survival crosses a target probability `p` at a given temperature (an "LT" / lethal- time-style quantity, e.g. `p = 0.5` gives the median survival time). Because the threshold is interpreted *relative to the asymptotes*, the default `p = 0.5` returns the curve's midpoint duration, where `log10(duration) = mid` exactly.

Usage

```
derive_lt(object, p = 0.5, temp, group = NULL)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`p`	Target survival probability in `(low, up)` (default `0.5`).
`temp`	Numeric temperature(s) at which to solve.
`group`	Optional single group level (grouped fits only). Required when the fit is grouped.

Details

Survival follows `p = low + (up - low) * plogis(-k (log10(duration) - mid))`, so the duration at which survival equals a target `p` solves

$$\log_{10}(duration) = mid - \mathrm{qlogis}\left(\frac{p - low}{up - low}\right)/k.$$

The target must lie strictly between `low` and `up` for a finite crossing; otherwise the survival curve never reaches `p` and `derive_lt()` aborts with an explanatory message (confidence-language, never silent). For a random-effects fit this is a population-level derived quantity; it does not add a group BLUP.

Value

A numeric vector of durations (same length as `temp`) on the data's native time unit.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
# Median survival duration at 36 C:
derive_lt(fit, p = 0.5, temp = 36)
```

`derive_tcrit` *Derive the critical temperature at a damage-rate floor (T_crit)*

Description

`derive_tcrit()` returns the rate-multiplier critical temperature `T_crit`: the temperature at which the thermal-damage rate falls to a chosen low floor `rate`. It is the maximum-likelihood analogue of the `bayesTLS` `extract_tdt()` absolute-family `T_crit`, and follows directly from the fitted `CTmax` and `z`:

$$T_{crit} = CTmax + z \log_{10}(rate/100).$$

Because `rate < 100` makes `log10(rate / 100) < 0` and `z > 0`, `T_crit` sits below `CTmax`: it is the lower thermal threshold at which damage becomes negligible (the temperature cutoff a heat-injury accumulation model treats as "no damage").

Usage

```
derive_tcrit(object, rate = 1, group = NULL)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`rate`	Damage-rate floor(s), a percentage of the lethal dose per hour (strictly positive). A scalar or a vector; default `1`.
`group`	Optional single group level (grouped fits only; required when the fit is grouped).

Details

`rate` is a damage-rate floor expressed as a **percentage of the lethal dose per hour**; `bayesTLS` brackets observed breakpoints with a default range of `0.1–1` %/hour. Unlike the Bayesian path, which samples `rate` to fold an operational choice into the posterior, freqTLS treats `rate` as a fixed input and returns the deterministic transform of the fitted `CTmax` and `z` (combine their confidence intervals if you need to propagate uncertainty). For a random-effects fit this is a population-level derived quantity; it does not add a group BLUP.

`T_crit` assumes a **lethal endpoint**: it is a damage-accumulation concept, so for sublethal endpoints (knockdown, photosynthetic failure) the steeper `z` drives it implausibly low. `derive_tcrit()` says so, once per call.

Value

A numeric vector of critical temperatures (degrees C), one per `rate`.

See Also

`derive_ctmax()` for the absolute-threshold critical temperature.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
derive_tcrit(fit, rate = c(0.1, 1)) # lower thermal thresholds
```

---

diagnose_tdt_fit *Diagnose a freqTLS fit (frequentist twin of* `diagnose_tdt_fit`*)*

Description

The maximum-likelihood analogue of `bayesTLS::diagnose_tdt_fit()`: where the Bayesian version reports Rhat / ESS / divergences, the freqTLS version reports optimiser convergence, a positive-definite Hessian, and the gradient norm at the optimum, with a single `all_pass` flag.

Usage

```
diagnose_tdt_fit(object)
```

Arguments

`object`	A `freq_tls` fit from `fit_4pl()` (or a `profile_tls` fit).

Value

A one-row tibble of convergence diagnostics.

See Also

`fit_4pl()`, `check_tls()`

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(dat, family = "binomial", t_ref = 1, quiet = TRUE)
diagnose_tdt_fit(fit)
```

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`). Only `dead` is a valid freqTLS response: aggregate it to counts for the beta-binomial lethal fit. The `t_coma` and `prod` columns are retained to preserve the deposited record and provide study context; they require time-to-event and reproductive-response models that freqTLS does not fit. `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 = "freqTLS")`.

Examples

```
# Lethal endpoint: aggregate per-individual deaths to cell counts, then
# prepare the data for a beta-binomial 4PL.
cells <- stats::aggregate(
  cbind(n_total = rep.int(1L, nrow(dsuzukii)), n_dead = dead) ~
    temp + time + sex,
  data = dsuzukii,
  FUN = sum
)
std <- standardize_data(cells, temp = "temp", duration = "time",
                        n_total = "n_total", n_dead = "n_dead",
                        duration_unit = "minutes")
```

`extract_tdt` *Extract z, CTmax and (optionally) T_crit with bootstrap confidence intervals*

Description

The frequentist twin of `bayesTLS::extract_tdt()`. Runs a parametric bootstrap (via the freqTLS engine), derives the thermal-death-time quantities on each replicate, and returns the same nested `$z` / `$CTmax` / `$T_crit` structure (each a list of `draws` + `summary`). The per-replicate tables are the frequentist analogue of posterior draws; `*_median` is the maximum-likelihood point estimate and `*_lower` / `*_upper` are bootstrap percentiles.

Usage

```
extract_tdt(
  object,
  target_surv = "relative",
  lethal = FALSE,
  TC_rate_range = c(0.1, 1),
  nboot = 1000L,
  level = 0.95,
  seed = NULL,
  by = NULL
)
```

Arguments

`object`	A `freq_tls` fit from `fit_4pl()` (or a `profile_tls` fit).
`target_surv`	`"relative"` (curve midpoint, default), `"absolute"` (50% survival), or a numeric survival level in `(0, 1)` for an LTx CTmax.
`lethal`	If `TRUE`, also derive `T_crit` (the damage-rate-floor critical temperature, `CTmax + z * log10(rate / 100)`, `rate` log-uniform over `TC_rate_range`), anchored at the fit's reference time.

TC_rate_range	Damage-rate floor range (percent of lethal dose per hour) for T_crit. Default c(0.1, 1).
nboot	Number of bootstrap replicates (default 1000; smaller is faster).
level	Confidence level (default 0.95).
seed	Optional RNG seed for reproducible replicates / rate draws.
by	Optional name for the grouping column; defaults to the fit moderator.

Value

A list with $z, $CTmax, ($T_crit when lethal), and $meta. Each quantity is list(draws = <tibble>, summary = <tibble
Column names follow bayesTLS: z_median/z_lower/z_upper for z; temp_median/temp_lower/
temp_upper for CTmax and T_crit; per-draw value columns are z / temp.

See Also

fit_4pl(), tls(), get_z_summary(), get_ctmax_summary()

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(dat, family = "binomial", t_ref = 1, quiet = TRUE)
tdt <- extract_tdt(fit, nboot = 10, seed = 1)
tdt$CTmax$summary
```

---

fit_4pl *Fit the 4PL thermal-load-sensitivity model by maximum likelihood (TMB)*

Description

The frequentist twin of bayesTLS::fit_4pl(). Consumes standardize_data() output and fits the single-stage 4PL thermal death-time model, parameterised directly in CTmax and thermal sensitivity (z), via the freqTLS TMB engine. Returns a freq_tls workflow object; uncertainty (Wald / profile / bootstrap) is computed on demand by the quantity twins (tls(), confint()).

Usage

```
fit_4pl(
  data,
  ctmax = NULL,
  z = NULL,
  up = NULL,
```

```
  low = NULL,
  k = NULL,
  by = NULL,
  threshold = c("relative", "absolute"),
  p = 0.5,
  t_ref = 60,
  bounds = c(0, 1),
  family = NULL,
  method = c("profile", "wald", "bootstrap"),
  start = NULL,
  control = list(),
  trace = FALSE,
  quiet = FALSE
)
```

Arguments

data	Output of standardize_data().
ctmax, z, up, low, k, by	Direct-mode formula interface; see make_4pl_formula(). Supplying ctmax/z (or by) fits per-group CTmax/z; the two headline formulas must produce the same fixed-effect columns.
threshold	"relative" (default; CTmax/z at the curve midpoint) or "absolute" (at the p-survival level). The fitting backbone currently accepts only "relative"; obtain absolute-threshold quantities post fit with extract_tdt() and target_surv = "absolute".
p	Survival level for the absolute threshold (default 0.5).
t_ref	Reference exposure time (in the data's duration_unit) at which CTmax is reported. Default 60 (e.g. minutes); use t_ref = 1 for hours.
bounds	Asymptote range. Only c(0, 1) is currently accepted. Supply survival as a probability in [0, 1] and let the model estimate low and up within that range; non-default bounds stop with an error.
family	"beta_binomial" (default for counts), "binomial", or "beta". NULL picks beta for a proportion response, else beta-binomial.
method	Default interval method for downstream extraction ("profile", "wald", or "bootstrap"); stored in the object.
start, control, trace, quiet	Passed to the engine fit_tls().

Value

A freq_tls object: a list with $fit (the engine fit), $data, $formula, and $meta (threshold, t_ref, bounds, temp_mean, response_type, family, grouped, moderators, method).

Before interpretation

Before interpreting the fit, run `check_tls()`. Its help page gives a recovery action for each data-adequacy warning; `vignette("profile-likelihood")` explains strict open profiles and the default bootstrap fallback.

See Also

`standardize_data()`, `make_4pl_formula()`, `fit_tls()`, `check_tls()`

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(
  dat, family = "binomial", t_ref = 1, method = "wald", quiet = TRUE
)
coef(fit)
```

---

fit_tls *Fit a single-stage 4PL thermal-load-sensitivity model by maximum likelihood*

Description

`fit_tls()` fits the descending four-parameter logistic (4PL) thermal death-time model to survival-count data, parameterised **directly in** `CTmax` **and** `z` (thermal sensitivity) so that both headline quantities can be profiled. Survival is modelled as a function of `log10(duration)`; the midpoint moves with temperature through `CTmax` and `z` (see `vignette("model-math")`).

Usage

```
fit_tls(
  x,
  y,
  n,
  time,
  temp,
  group = NULL,
  family = c("beta_binomial", "binomial", "beta"),
  tref = 1,
  start = NULL,
  control = list(),
  trace = FALSE,
  quiet = FALSE,
  data = NULL
)
```

Arguments

x	Either a data frame (column interface) or a tls_formula from tls_bf() (formula interface). For back-compatibility the first argument is still positional, so fit_tls(my_data, y = survived, ...) continues to work.
y	<data-masked> Column of successes (survivors), or, for the beta family, the response proportion in (0, 1).
n	<data-masked> Column of trials (total individuals). Required for the binomial and beta-binomial families; omit it for the beta family, whose response is already a proportion.
time	<data-masked> Column of exposure durations in the data's native unit (e.g. hours); used as log10(duration) internally.
temp	<data-masked> Column of assay temperatures (degrees C).
group	<data-masked> Optional grouping column. When supplied, each group gets its own direct, profile-able CTmax and z with shared low, up, and k. Defaults to NULL (a single ungrouped fit).
family	One of "beta_binomial" (default), "binomial", or "beta" (a continuous proportion in (0, 1)), or a tls_family object from beta_binomial_tls() / binomial_tls() / beta_tls().
tref	Reference time at which CTmax is defined, in the same unit as time (default 1, i.e. CTmax at one time unit).
start	Optional named list of starting values on the internal (unconstrained) scale, overriding the defaults. Names must match the parameters in src/profile_tls.cpp (beta_low, beta_up, beta_logk, beta_CT, beta_logz, log_phi).
control	List of optimiser controls; optimizer is passed to stats::nlminb()'s control, and trace toggles optimiser output.
trace	Logical; print optimiser progress. A shortcut for control$trace.
quiet	Logical; if TRUE, suppress freqTLS's own data-adequacy and identifiability diagnostic warnings and messages (the few-groups, beta boundary-clamp, and same-grouping notes). Genuine errors and optimiser warnings still surface, and check_tls() reports the diagnostics on demand. Default FALSE.
data	Used only in the formula interface: the data frame the tls_bf() columns are resolved against. Ignored in the column interface (where the data frame is x).

Details

There are two equivalent interfaces. In the **column interface**, columns are referenced with tidy evaluation: pass the bare column names of data (as in dplyr), not strings. In the **formula interface**, pass a tls_bf() object as x and the data frame as data; the brms/drmTMB-style grammar names the response, the two axes, and the CTmax / log_z predictors. Both interfaces feed the same likelihood engine, so a grouped formula fit and the matching group = column fit are numerically identical.

Value

An object of class `c("profile_tls", "tls_fit")`: a list with the call, the resolved `family`, `tref`, `group_levels`, a `data_summary`, the internal-scale MLE `par`, an `estimates` data frame of natural-scale parameters with standard errors, the `vcov` of the internal coordinates, the `logLik`, residual `df`, `AIC`, a `convergence` list (`code`/`pdHess`/`message`), the `name_map`, and the underlying TMB `obj`, optimiser `opt`, and `sdreport`.

Before interpretation

Run `check_tls()` on the fitted object. Its help page maps every data-adequacy warning to a concrete design or analysis response. `vignette("profile-likelihood")` explains strict open profiles and the default bootstrap recovery attempt.

See Also

`check_tls()`, `confint.profile_tls()`

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
fit$estimates

# The same fit through the formula interface:
fit2 <- fit_tls(
  tls_bf(survived | trials(total) ~ time(duration) + temp(temp)),
  data = d, family = "binomial", tref = 1
)
```

---

format_interval *Format a point estimate plus confidence interval as a single string*

Description

Format a point estimate plus confidence interval as a single string

Usage

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

Arguments

`median`, `lower`, `upper`	Numeric (scalar or vector). `median` is the central value (a point estimate or the median of bootstrap replicates), `lower` and `upper` the confidence-interval endpoints.
`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_ctmax *Extract the CTmax estimate(s)*

Description

Extract the CTmax estimate(s)

Usage

```
get_ctmax(fit, conf.int = TRUE, conf.level = 0.95)
```

Arguments

fit	A profile_tls fit from fit_tls().
conf.int	Logical; include Wald conf.low / conf.high (default TRUE).
conf.level	Confidence level for the Wald interval (default 0.95).

Value

A tibble of the CTmax row(s) from tidy_parameters().

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
get_ctmax(fit)
```

get_shape *Extract the shape parameters (low, up, k, and phi)*

Description

Extract the shape parameters (low, up, k, and phi)

Usage

```
get_shape(fit, conf.int = TRUE, conf.level = 0.95)
```

Arguments

fit	A profile_tls fit from fit_tls().
conf.int	Logical; include Wald conf.low / conf.high (default TRUE).
conf.level	Confidence level for the Wald interval (default 0.95).

Value

A tibble of the shape rows (low, up, k, and phi for the beta-binomial family) from tidy_parameters().

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
get_shape(fit)
```

get_z *Extract the thermal-sensitivity (z) estimate(s)*

Description

Extract the thermal-sensitivity (z) estimate(s)

Usage

```
get_z(fit, conf.int = TRUE, conf.level = 0.95)
```

Arguments

fit	A profile_tls fit from fit_tls().
conf.int	Logical; include Wald conf.low / conf.high (default TRUE).
conf.level	Confidence level for the Wald interval (default 0.95).

Value

A tibble of the z row(s) from `tidy_parameters()`.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
get_z(fit)
```

`heat_injury_envelope` *Parametric-bootstrap confidence envelope for a heat-injury trajectory*

Description

`heat_injury_envelope()` is the uncertainty counterpart of `predict_heat_injury()`: it redraws the fitted curve parameters by parametric bootstrap (the same machinery `confint()` uses), re-integrates the survival trajectory under the temperature `trace` for each draw with the documented dose-accumulation map, and returns a pointwise confidence band around the point-estimate survival curve. The band is **prior-free** – it carries no prior and makes no probability statement about the parameters; it is the likelihood-path analogue of the `bayesTLS` posterior survival band, **not** a credible band.

Usage

```
heat_injury_envelope(
  object,
  trace,
  group = NULL,
  target_surv = NULL,
  t_c = NULL,
  repair = NULL,
  irreversible = TRUE,
  nboot = 1000L,
  conf.level = 0.95,
  seed = NULL
)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`, or a `freq_tls` workflow from `fit_4pl()`.
`trace`	A data frame with numeric columns `time` (strictly increasing, at least two rows) and `temp` (degrees C).
`group`	Optional single group level (grouped fits only; required when the fit is grouped).

target_surv	Optional absolute survival threshold defining one lethal dose: a single probability strictly between the fitted lower and upper asymptotes. NULL (default) uses the project-default relative threshold (the curve midpoint (low + up) / 2), matching derive_ctmax() and derive_lt().
t_c	Optional damage-cutoff temperature (degrees C): at or below it the damage rate is zero. NULL (default) applies no cutoff.
repair	Optional named list of Sharpe-Schoolfield repair parameters (see Details); NULL (default) means no repair.
irreversible	Logical; if TRUE (default) survival is monotone non-increasing (mortality does not reverse even if dose is repaired).
nboot	Number of bootstrap replicates (default 1000).
conf.level	Width of the pointwise confidence band (default 0.95).
seed	Optional integer seed; when supplied the bootstrap is reproducible without disturbing the caller's random stream.

Value

A tibble with time, temp, survival (the point-estimate trajectory from predict_heat_injury()), and conf.low / conf.high (the pointwise parametric-bootstrap confidence band).

See Also

predict_heat_injury() for the point trajectory, plot_heat_injury() to draw the band.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
trace <- data.frame(time = seq(0, 2, by = 0.1),
                    temp = 34 + 6 * sin(seq(0, 2, by = 0.1)))
heat_injury_envelope(fit, trace, nboot = 50, seed = 1)
```

---

make_4pl_formula *Build a freqTLS 4PL formula from the direct CTmax/z interface*

Description

Translates the bayesTLS-style direct-mode arguments (ctmax, z, up, low, k, by) into the engine's tls_bf() tls_formula object. Supplying a ctmax and/or z formula is the direct parameterisation; by is shorthand for grouping CTmax and z by a single moderator (~ 0 + by).

Usage

```
make_4pl_formula(
  ctmax = NULL,
  z = NULL,
  up = NULL,
  low = NULL,
  k = NULL,
  by = NULL,
  family = "beta_binomial"
)
```

Arguments

ctmax, z, up, low, k	One-sided formulas (or NULL). ctmax/z set the CTmax / thermal-sensitivity structure and must have the same fixed-effect model-matrix columns; up/low/k set the 4PL shape. Random intercepts are supported on ctmax, z, low, and k, but not up.
by	Optional single moderator column name; shorthand for ctmax = z = ~ 0 + by when those are not given explicitly.
family	"beta_binomial", "binomial", or "beta" (selects the response idiom: n_surv | trials(n_total) for counts, bare survival for the continuous-proportion beta family).

Details

Following the freqTLS constant-shape invariant, the asymptotes and steepness (up, low, k) default to **shared** (~ 1) so the temperature effect runs through the midpoint (CTmax / z) only; pass an explicit formula to let a shape vary. ctmax and z must produce the same fixed-effect model-matrix columns. Supported random intercepts go inside the ctmax/z/low/k formulas; up random effects are not supported. For example, ctmax = ~ 1 + (1 | batch) keeps the same intercept-only fixed design as the default z = ~ 1 while adding a CTmax random intercept.

Value

A tls_formula object (as built by tls_bf()).

See Also

fit_4pl(), tls_bf(), standardize_data()

Examples

```
make_4pl_formula()
make_4pl_formula(by = "population", family = "binomial")
```

`plot.profile_tls_profile` *Plot a profile-likelihood deviance curve*

Description

`plot()` for a `"profile_tls_profile"` object draws the likelihood-ratio deviance curve against the natural-scale parameter. A dotted horizontal line marks the profile-t cutoff `qt(1 - alpha/2, df)^2`; a solid vertical line marks the point estimate; dashed vertical lines mark the interval endpoints when they are finite. The wording is deliberately "confidence" – this is a likelihood curve, never a posterior. A non-closing side is annotated rather than drawn as a closed bound.

Usage

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

Arguments

`x`	A `"profile_tls_profile"` object from `profile.profile_tls()`.
`...`	Reserved; must be empty.

Details

This is the per-parameter profile curve; the full Confidence-Eye interval displays are added in Phase 4.

Value

A `ggplot` object (invisibly when printed for its side effect).

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
plot(profile(fit, "CTmax"))
```

plot_confidence_eye *Confidence-Eye (or line) display of headline confidence intervals*

Description

plot_confidence_eye() draws the freqTLS Confidence Eye for one or more headline parameters (CTmax, z, or any other confint.profile_tls() target, including grouped names). It is a HORIZONTAL forest display: each parameter (and group level) is a row, the parameter value runs along the x-axis, and the confidence interval is a short, wide pale lens with a hollow point estimate. The shallow horizontal lens reads as a confidence *interval*, never a posterior density – freqTLS intervals are likelihood confidence intervals, so the wording is "confidence", never "posterior". The layout follows the gllvmTMB / drmTMB Confidence-Eye contract.

Usage

```
plot_confidence_eye(
  fit,
  parm = c("CTmax", "z"),
  method = c("profile", "wald", "bootstrap"),
  level = 0.95,
  style = c("eye", "line"),
  raw_data = TRUE,
  fallback = TRUE,
  nboot = 1000L,
  boot_seed = NULL,
  cores = 1L,
  ...
)
```

Arguments

fit	A profile_tls fit from fit_tls().
parm	Character vector of target parameter names. Defaults to c("CTmax", "z") (the headline quantities). Grouped names (e.g. "CTmax:larva") are accepted.
method	One of "profile" (default), "wald", or "bootstrap"; forwarded to confint.profile_tls().
level	Confidence level (default 0.95).
style	One of "eye" (default; pale horizontal lens + hollow point) or "line" (a confidence-interval bar with caps + hollow point, no lens).
raw_data	Logical; overlay observed assay temperatures as a rug on temperature-scale rows (default TRUE).
fallback, nboot, boot_seed, cores	Forwarded to confint.profile_tls(): control the parametric-bootstrap fallback for non-closing profiles (so the eye draws an honest bootstrap lens instead of only a hollow point), make it reproducible, and parallelise the refits. Defaults TRUE, 1000, NULL, and 1.
...	Reserved; must be empty.

Details

Honest fallback for open profiles:

When a profile does not close (`conf.status` is `"open_lower"`, `"open_upper"`, `"open_both"`, or a bound is `NA`), no lens is drawn for that row: a hollow point marks the estimate and the subtitle flags the open interval. The eye is never fabricated from an open profile. A `"wald_fallback"` interval (e.g. `up`) still gets a lens, with the source noted in the caption.

Raw data:

With `raw_data = TRUE` (default), the observed assay temperatures are drawn as a rug beneath any temperature-scale row (`CTmax`), showing the data support and flagging extrapolation when `CTmax` sits outside the assayed range.

Parameters on different scales (temperature for `CTmax`, a positive multiplier for `z`) are stacked in separate panels with a free x-axis.

Value

A `ggplot` object.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
plot_confidence_eye(fit, parm = c("CTmax", "z"))
```

`plot_heat_injury` *Plot a heat-injury survival trajectory with a bootstrap confidence band*

Description

`plot_heat_injury()` draws the point-estimate survival trajectory from `predict_heat_injury()` inside the pointwise parametric-bootstrap confidence band from `heat_injury_envelope()`. The band is prior-free – a confidence band, never a posterior / credible band (the project's honest-uncertainty contract).

Usage

```
plot_heat_injury(
  object,
  trace,
  group = NULL,
  target_surv = NULL,
  t_c = NULL,
  repair = NULL,
  irreversible = TRUE,
  nboot = 1000L,
```

```
  conf.level = 0.95,
  seed = NULL,
  time_div = 1,
  xlab = "Time",
  ylab = "Survival"
)
```

Arguments

object	A `profile_tls` fit from `fit_tls()`, or a `freq_tls` workflow from `fit_4pl()`.
trace	A data frame with numeric columns `time` (strictly increasing, at least two rows) and `temp` (degrees C).
group	Optional single group level (grouped fits only; required when the fit is grouped).
target_surv	Optional absolute survival threshold defining one lethal dose: a single probability strictly between the fitted lower and upper asymptotes. `NULL` (default) uses the project-default relative threshold (the curve midpoint `(low + up) / 2`), matching `derive_ctmax()` and `derive_lt()`.
t_c	Optional damage-cutoff temperature (degrees C): at or below it the damage rate is zero. `NULL` (default) applies no cutoff.
repair	Optional named list of Sharpe-Schoolfield repair parameters (see Details); `NULL` (default) means no repair.
irreversible	Logical; if `TRUE` (default) survival is monotone non-increasing (mortality does not reverse even if dose is repaired).
nboot	Number of bootstrap replicates (default `1000`).
conf.level	Width of the pointwise confidence band (default `0.95`).
seed	Optional integer seed; when supplied the bootstrap is reproducible without disturbing the caller's random stream.
time_div	Optional positive divisor applied to `time` on the x-axis (for example `24` to show days when the trace is in hours); default `1`.
xlab, ylab	Axis labels.

Value

A `ggplot` object.

See Also

`heat_injury_envelope()` for the band data, `predict_heat_injury()` for the point trajectory.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
trace <- data.frame(time = seq(0, 2, by = 0.1),
                    temp = 34 + 6 * sin(seq(0, 2, by = 0.1)))
plot_heat_injury(fit, trace, nboot = 50, seed = 1)
```

plot_survival_curves *Plot fitted survival curves against duration*

Description

plot_survival_curves() draws the fitted survival probability as a function of exposure duration (on a log10 x-axis), one curve per temperature, with the observed survival proportions overlaid as points. For a grouped fit the curves are faceted by group.

Usage

```
plot_survival_curves(fit, temps = NULL, times = NULL, ...)
```

Arguments

fit	A profile_tls fit from fit_tls().
temps	Numeric vector of temperatures to draw curves for. Defaults to the distinct observed temperatures (capped at a readable number).
times	Numeric vector of durations to evaluate the smooth curve over. Defaults to a log-spaced sequence over the observed duration range.
...	Reserved; must be empty.

Value

A ggplot object.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
plot_survival_curves(fit)
```

plot_survival_surface *Plot the fitted survival surface over temperature and duration*

Description

plot_survival_surface() draws the fitted survival probability as a filled heatmap over a temperature-by-duration grid, with contour lines, using predict_survival_surface(). Duration is on a log10 axis. For a grouped fit the surface is faceted by group.

Usage

```
plot_survival_surface(fit, temps = NULL, times = NULL, contour = TRUE, ...)
```

Arguments

fit	A profile_tls fit from fit_tls().
temps, times	Numeric grids passed to predict_survival_surface(). Defaults span the observed ranges.
contour	Logical; overlay contour lines (default TRUE).
...	Reserved; must be empty.

Value

A ggplot object.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
plot_survival_surface(fit)
```

---

plot_tdt_curve *Plot the thermal death-time (TDT) curve: survival-threshold time vs temperature*

Description

plot_tdt_curve() draws the duration at which survival crosses a target probability p (default the relative midpoint, p = 0.5) against temperature – the classic thermal-death-time line, here read directly off the fitted 4PL via derive_lt(). Time is shown on a log10 axis. For a grouped fit a line is drawn per group.

Usage

```
plot_tdt_curve(fit, p = 0.5, temps = NULL, ...)
```

Arguments

fit	A profile_tls fit from fit_tls().
p	Target survival probability for the threshold (default 0.5).
temps	Numeric vector of temperatures. Defaults to a sequence over the observed temperature range.
...	Reserved; must be empty.

Value

A ggplot object.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
plot_tdt_curve(fit)
```

`predict.profile_tls` *Predict survival, link, or midpoint from a fitted freqTLS model*

Description

`predict()` evaluates the fitted four-parameter logistic (4PL) thermal-load- sensitivity model at new temperature-by-duration cells, using exactly the same forward map as the TMB engine in `src/profile_tls.cpp`:

$$mid = \log_{10}(t_{ref}) - (temp - CTmax_g)/z_g$$

$$p = low + (up - low)\,\mathrm{plogis}(-k(\log_{10}(duration) - mid)).$$

Usage

```
## S3 method for class 'profile_tls'
predict(
  object,
  newdata,
  type = c("survival", "link", "midpoint"),
  re.form = c("population", "conditional"),
  ...
)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`newdata`	A data frame with numeric columns `temp` and `duration`, plus every predictor used in the fitted `CTmax`, `log_z`, `low`, `up`, and `log_k` fixed designs. Include `group` for a grouped column-interface fit. Conditional random-effect predictions additionally require every fitted grouping column. `duration` must be strictly positive (it is `log10`-transformed). For `type = "midpoint"`, `duration` may be omitted.
`type`	One of `"survival"` (default), `"link"`, or `"midpoint"`.
`re.form`	How to handle fitted random intercepts. `"population"` (default) sets them to zero; `"conditional"` adds the fitted BLUP for each random-effect grouping column in `newdata`. When omitted for a random- effects fit, `predict()` warns that it is returning a population prediction.
`...`	Reserved; must be empty.

Details

Three response types are available:

- `"survival"` (default) returns the fitted survival probability in `(0, 1)`.
- `"link"` returns the logit of the survival probability, `qlogis(survival)`.
- `"midpoint"` returns the temperature-dependent 4PL midpoint `mid` on the `log10(duration)` axis (constant within a temperature, so the `duration` column is ignored for this type but a `temp` column is still required).

`newdata` must also contain every predictor used by the fitted fixed-effect designs for `CTmax`, `log_z`, `low`, `up`, or `log_k`. For a grouped column- interface fit, supply `group` with values from the fitted `group_levels`. For a formula fit, a literal `tls_bf()` call preserves the fixed-design formulas needed to rebuild transformed or interacted terms. If the model was instead passed through a formula-object variable, `predict()` can rebuild direct numeric design columns but asks the user to refit with a literal `tls_bf()` call when a transformed or interacted design cannot be recovered safely.

Value

A numeric vector with one element per row of `newdata`. Survival values lie in `(0, 1)`.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
nd <- expand.grid(temp = c(34, 36, 38), duration = c(1, 2, 4))
predict(fit, nd, type = "survival")

# A continuous predictor used by CTmax and log_z must also be in newdata.
d$x <- rep(c(-1, 1), length.out = nrow(d))
fit_x <- fit_tls(
  tls_bf(survived | trials(total) ~ time(duration) + temp(temp),
         CTmax ~ x, log_z ~ x),
  data = d, family = "binomial", tref = 1
)
predict(fit_x, data.frame(temp = 36, duration = 2, x = c(-1, 1)))


# Choose population or fitted-group prediction explicitly for an RE fit.
dre <- simulate_tls(family = "binomial", CTmax = 36, z = 4,
                    re_sd = 1, n_re_groups = 8, seed = 2)
fit_re <- fit_tls(
  tls_bf(survived | trials(total) ~ time(duration) + temp(temp),
         CTmax ~ 1 + (1 | colony)),
  data = dre, family = "binomial", tref = 1
)
colony <- as.character(ranef(fit_re)$group[1])
nd_re <- data.frame(temp = 36, duration = 2, colony = colony)
predict(fit_re, nd_re, re.form = "population")
predict(fit_re, nd_re, re.form = "conditional")
```

predict_heat_injury *Predict cumulative heat injury under a temperature trace*

Description

predict_heat_injury() is the deterministic, maximum-likelihood **prediction** analogue of bayesTLS::predict_heat_inj given a fitted thermal-load- sensitivity curve and a temperature time-series (a "trace"), it accumulates thermal damage as a fraction of the lethal dose and reads survival back off the fitted 4PL. It does **not** fit an injury or repair model – fitting injury / repair dynamics remains a bayesTLS concern (the complementary boundary); predict_heat_injury() only predicts injury from the already-fitted survival curve. For a random-effects fit it uses the population curve and does not add a fitted group BLUP.

Usage

```
predict_heat_injury(
  object,
  trace,
  group = NULL,
  target_surv = NULL,
  t_c = NULL,
  repair = NULL,
  irreversible = TRUE
)
```

Arguments

object	A profile_tls fit from fit_tls(), or a freq_tls workflow from fit_4pl().
trace	A data frame with numeric columns time (strictly increasing, at least two rows) and temp (degrees C).
group	Optional single group level (grouped fits only; required when the fit is grouped).
target_surv	Optional absolute survival threshold defining one lethal dose: a single probability strictly between the fitted lower and upper asymptotes. NULL (default) uses the project-default relative threshold (the curve midpoint (low + up) / 2), matching derive_ctmax() and derive_lt().
t_c	Optional damage-cutoff temperature (degrees C): at or below it the damage rate is zero. NULL (default) applies no cutoff.
repair	Optional named list of Sharpe-Schoolfield repair parameters (see Details); NULL (default) means no repair.
irreversible	Logical; if TRUE (default) survival is monotone non-increasing (mortality does not reverse even if dose is repaired).

Details

Dose-accumulation model:

One **lethal dose** is the thermal load that drives survival to a target set by `target_surv`. With the default `target_surv = NULL` the target is the project-default **relative** threshold – the curve midpoint `(low + up) / 2`. At temperature `T` the lethal time to the target is `LT(T) = tref * 10^((CTmax - T) / z - q / k)`, with `q = qlogis((target_surv - low) / (up - low))` (so `q = 0` at the midpoint, the same quantity as `derive_lt()` at survival `target_surv`). The instantaneous damage rate is `1 / LT(T)` (lethal doses per time unit). Cumulative dose is accumulated by forward Euler over the trace, using the actual per-step time increments:

$$D_j = \max\bigl(0,\ D_{j-1} + (\mathrm{dmg}(T_{j-1}) - \mathrm{rep}_{j-1})\,\Delta t_j\bigr), \quad D_1 = 0,$$

where $\Delta t_j = t_j - t_{j-1}$. Survival is read back from the 4PL by treating the accumulated dose as an equivalent `log10`-time: `survival(D) = low + (up - low) * plogis(-k * log10(D) + q)`, so `D = 1` (one lethal dose) reaches exactly `target_surv` – the relative midpoint `(low + up) / 2` by default, or an **absolute** survival threshold when `target_surv` is supplied.

The trace `time` and the fit's duration / `tref` **must share a time unit** (the damage rate is per that unit). With `irreversible = TRUE` (default) survival is monotone non-increasing. A damage cutoff `t_c` (for example from `derive_tcrit()`) sets the damage rate to zero at or below `t_c`.

This integrator is forward Euler (left-endpoint, per actual step), not the single-`dt` scheme some implementations use; irregular traces are integrated with their real increments.

Repair (optional, not identified by the data):

If `repair` is supplied, a Sharpe-Schoolfield repair rate is subtracted each step (scaled by the current survival fraction, so repair shrinks as the population dies). The repair parameters are a **user-supplied scenario layer**: they are not identified by the survival data the model was fitted to, so `predict_heat_injury()` warns when they are used. `repair` is a named list with `r_ref`, `t_a`, `t_al`, `t_ah`, `t_l`, `t_h`, `t_ref`, with the four reference temperatures in **Kelvin**.

Value

A data frame with columns `time`, `temp`, `dose` (cumulative, as a fraction of the lethal dose), `injury` (`dose * 100`, percent), and `survival`.

See Also

`derive_lt()` for the lethal time, `derive_tcrit()` for a damage cutoff temperature.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
trace <- data.frame(time = seq(0, 2, by = 0.05),
                    temp = 34 + 6 * sin(seq(0, 2, by = 0.05)))
head(predict_heat_injury(fit, trace))
```

predict_survival_curves *Predict the fitted survival surface with bootstrap confidence bands*

Description

The frequentist twin of `bayesTLS::predict_survival_curves()`. Evaluates the fitted 4PL survival probability over a temperature-by-duration grid and adds parametric-bootstrap confidence bands. For random-effects fits the curves are population-level: random intercepts are integrated during bootstrap refits, but no fitted group BLUP is added to the reported curve.

Usage

```
predict_survival_curves(
  object,
  temps = NULL,
  durations = NULL,
  nboot = 500L,
  level = 0.95,
  seed = NULL,
  by = NULL
)
```

Arguments

`object`	A `freq_tls` fit from `fit_4pl()` (or a `profile_tls` fit).
`temps`	Temperatures to predict at (default: the observed assay temps).
`durations`	Exposure durations (default: 100 points log-spaced over the observed range, in the data's duration unit).
`nboot`	Number of bootstrap replicates for the bands (default 500).
`level`	Confidence level (default 0.95).
`seed`	Optional RNG seed.
`by`	Optional name for the grouping column.

Value

A `freq_surv_curves` object: `$summary` (a tibble of `[<group>,] temp, duration, survival_lower, survival_median,` and `$meta`.

See Also

`fit_4pl()`, `predict_survival_surface()`, `tls()`

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(dat, family = "binomial", t_ref = 1, quiet = TRUE)
curves <- predict_survival_curves(
  fit, temps = c(34, 36), durations = c(1, 4), nboot = 10, seed = 1
)
curves$summary
```

---

predict_survival_surface

Predict a survival surface over a temperature-by-duration grid

---

Description

`predict_survival_surface()` evaluates the fitted survival probability on a factorial grid of temperatures by durations, returning a long data frame suitable for a heatmap or contour plot (see `plot_survival_surface()`). For random-effects fits this helper returns population-level predictions (random intercepts set to zero); use `predict(..., re.form = "conditional")` for known-group conditional predictions. General continuous fixed designs require `predict()` with their covariate columns supplied in `newdata`.

Usage

```
predict_survival_surface(object, temps = NULL, times = NULL, group = NULL)
```

Arguments

`object`	A `profile_tls` fit from `fit_tls()`.
`temps`	Numeric vector of temperatures. Defaults to a length-60 sequence spanning the fit's observed temperature range.
`times`	Numeric vector of durations (strictly positive). Defaults to a length-60 log-spaced sequence spanning the fit's observed duration range.
`group`	Optional single group level (grouped fits only). When `NULL` (default) the surface is built for every group level and the result carries a `group` column.

Value

A long `data.frame` with columns `temp`, `duration`, `survival` (and `group` when the fit is grouped).

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
head(predict_survival_surface(fit, temps = c(34, 36, 38), times = c(1, 2, 4)))
```

`profile.profile_tls` *Profile-likelihood curves for a fitted thermal-load-sensitivity model*

Description

`profile()` computes the profile-likelihood deviance curve for one scalar target of a `fit_tls()` model. For the target it fixes the corresponding internal (unconstrained) coordinate on a grid, re-optimises the remaining coordinates at each grid point, and returns the deviance `D = 2 * (logLik_hat - logLik_profile)` together with the profile-t cutoff and the profile-likelihood confidence interval. Because the profile is taken on the unconstrained coordinate and the endpoints are then transformed by a monotone function, the interval is exactly equivariant: the `z` interval equals `exp()` of the internal `log_z` interval (the headline equivariance check).

Usage

```
## S3 method for class 'profile_tls'
profile(fitted, parm, level = 0.95, npoints = 30L, trace = FALSE, ...)

## S3 method for class 'profile_tls_profile'
print(x, digits = 4, ...)
```

Arguments

`fitted`	A `profile_tls` fit from `fit_tls()`.
`parm`	A single target name (see Targets).
`level`	Confidence level for the interval and the cutoff line (default `0.95`).
`npoints`	Number of grid points for the deviance curve (default `30`).
`trace`	Logical; print inner-optimisation progress.
`...`	Reserved; must be empty.
`x`	A `"profile_tls_profile"` object.
`digits`	Number of significant digits for the printed summary (default `4`).

Details

The algorithm is a map-refit profile: the target coordinate is fixed with TMB's `map` mechanism and the rest re-optimised, mirroring the bracket-then-`stats::uniroot()` endpoint solver in `drmTMB::R/profile.R:2314-2373`. See `vignette("profile-likelihood")`.

Value

An object of class "profile_tls_profile": a list with parm, profile_value (grid on the natural scale), deviance, estimate, conf.low, conf.high, conf.status, cutoff, level, scale, and transformation.

Functions

- print(profile_tls_profile): Print a compact summary of the profile.

Targets

Target	Profiled coordinate	Endpoint transform
CTmax, CTmax:<grp>	beta_CT[g]	identity
z, z:<grp>	beta_logz[g]	exp
log_z, log_z:<grp>	beta_logz[g]	identity
low	beta_low	plogis
k	beta_logk	exp
phi	log_phi	exp
up	(Wald/delta fallback)	–
dCTmax:<a>-<b>, dlog_z:<a>-<b>	contrast recoding	identity

Under the disjoint-bounds parameterisation up = up_min + up_w * plogis(beta_up) has its own coordinate beta_up, but freqTLS does not yet profile it (the profile path is wired for low but not up — symmetric work, simply not implemented). freqTLS falls back to the delta-method Wald interval for up and says so. Group contrasts (dCTmax, dlog_z) are profiled directly by recoding the design so the contrast is itself a coordinate.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
pc <- profile(fit, "CTmax")
pc$conf.low
pc$conf.high
```

---

ranef *Random-effect BLUPs (conditional modes) for a freqTLS fit*

Description

ranef() returns the predicted random intercepts (the conditional modes / BLUPs) with their conditional standard errors, for a fit with a random intercept on any of CTmax, log_z, low, or log_k (<param> ~ <fixed> + (1 | group)). It errors for a fixed-effects-only fit. Each BLUP is a deviation on its coordinate's internal scale: CTmax in degrees C, log_z on log(z), low on logit(low), log_k on log(k). When several REs are present the rows are stacked in CTmax, log_z, low, log_k order.

Usage

```
ranef(object, ...)

## S3 method for class 'profile_tls'
ranef(object, ...)
```

Arguments

object	A profile_tls fit from fit_tls() with a random intercept.
...	Reserved; must be empty.

Value

A tibble with one row per group level (per RE term): group, term ("CTmax", "log_z", "low", or "log_k"), estimate (the BLUP), and std.error (the conditional SE).

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4,
                  re_sd = 1.5, n_re_groups = 12, seed = 42)
fit <- fit_tls(
  tls_bf(survived | trials(total) ~ time(duration) + temp(temp),
         CTmax ~ 1 + (1 | colony)),
  data = d, family = "binomial", tref = 1)
ranef(fit)
```

---

shrimp_lethal *Brown shrimp lethal thermal-death-time data*

Description

Replicate lethal-TDT trials for brown shrimp (*Crangon crangon*). Each row is one tank of individuals exposed to a fixed assay temperature for a fixed duration; the response is the proportion that died. The model-ready frame for Case Study 1 (lethal endpoint).

Usage

```
shrimp_lethal
```

Format

A data frame with 148 rows and 6 variables:

Date Experiment date (use as a grouping factor).

Tank Holding-tank identifier (use as a grouping factor).

Temperature_assay Assay temperature (degrees C).

Duration_exposure_hours Exposure duration (hours).

N_individuals_after_trial Number of individuals in the trial.

Mortality_after_trial Proportion that died during the trial (deaths / `N_individuals_after_trial`), in the unit interval. Consumed by `standardize_data(mortality = "Mortality_after_trial")`.

Source

Brown shrimp lethal-TDT assay (Case Study 1), obtained from the **bayesTLS** package distribution by Noble, Arnold, and Pottier (2026), licensed CC BY 4.0. freqTLS retains the mortality proportion and documents its count reconstruction above. Raw file: `system.file("extdata", "data_lethal_TDT_brown_shrimp.csv", package = "freqTLS")`.

Examples

```
std <- standardize_data(shrimp_lethal, temp = "Temperature_assay",
                        duration = "Duration_exposure_hours",
                        n_total = "N_individuals_after_trial",
                        mortality = "Mortality_after_trial",
                        random_effects = c("Date", "Tank"),
                        duration_unit = "hours")
```

`shrimp_sublethal` *Brown shrimp sublethal time-to-knockdown data*

Description

Sublethal TDT trials for brown shrimp (*Crangon crangon*): each cup of individuals contributes the elapsed time to loss of response to touch (knockdown) at a fixed assay temperature. Cleaned from the raw clock-time records (excluded rows dropped; start/stop times parsed to elapsed minutes).

Usage

```
shrimp_sublethal
```

Format

A data frame with 299 rows and 5 variables:

assay_temp Assay temperature (degrees C).

time_to_event Time to knockdown (minutes).

date_experiment Experiment date (grouping factor).

tank_ID Holding-tank identifier (grouping factor).

cup_ID Cup identifier, `Trial_ID_Sample` (grouping factor).

Source

Brown shrimp sublethal time-to-knockdown assay (Case Study 1, sublethal endpoint), obtained from the **bayesTLS** package distribution by Noble, Arnold, and Pottier (2026), licensed CC BY 4.0. freqTLS dropped excluded rows and converted the clock times to elapsed minutes. Raw file: `system.file("extdata", "data_sublethal_TDT_brown_shrimp.csv", package = "freqTLS")`.

`simulate_tls` *Simulate survival-count data from the 4PL thermal-load-sensitivity model*

Description

`simulate_tls()` draws survival counts from the locked data-generating process used throughout the freqTLS test suite and benchmarks. It builds a factorial grid of temperatures by durations by replicates, computes the true survival probability under the direct-`CTmax`/`z` 4PL (the same forward map as the TMB engine in `src/profile_tls.cpp`), and draws binomial or beta-binomial counts. The simulating truth is attached as `attr(, "truth")`.

Usage

```
simulate_tls(
  temps = seq(30, 42, by = 2),
  times = c(0.5, 1, 2, 4, 8),
  reps = 3,
  n = 20,
  low = 0.02,
  up = 0.98,
  k = 5,
  CTmax = 36,
  z = 3,
  phi = NULL,
  family = c("binomial", "beta_binomial", "beta"),
  group = NULL,
  re_sd = NULL,
  re_sd_z = NULL,
  re_sd_low = NULL,
  re_sd_logk = NULL,
  n_re_groups = NULL,
  re_group_name = "colony",
  tref = 1,
  seed = NULL
)
```

Arguments

`temps`	Numeric vector of assay temperatures (degrees C).
`times`	Numeric vector of exposure durations (native unit, e.g. hours).

`reps` Number of replicate observations per temperature-by-duration cell (per group).

`n` Number of individuals per observation (binomial size).

`low`, `up` Lower and upper survival asymptotes (`0 < low < up < 1`). A scalar (shared) or, for a grouped simulation, one value per group.

`k` Steepness of the logistic on the `log10(duration)` scale (`k > 0`). A scalar (shared) or, for a grouped simulation, one value per group.

`CTmax` Critical thermal maximum at `tref`. A scalar (ungrouped) or, for a grouped simulation, a vector with one value per group.

`z` Thermal sensitivity (`z > 0`). A scalar or, when grouped, a vector with one value per group.

`phi` Dispersion (sum of Beta shapes); `NULL` for the binomial family. Required when `family = "beta_binomial"` or `family = "beta"`.

`family` One of `"binomial"`, `"beta_binomial"`, or `"beta"` (a continuous proportion in `(0, 1)`, returned in a `prop` column).

`group` Optional **atomic** vector of group labels (character, factor, or numeric). When supplied, `CTmax` and `z` must each be either a single scalar (shared across groups) or a vector with one value per *distinct* group level, in the order the levels first appear. Passing a list (for example `group = list(A = list(CTmax = 34))`) is an error: use the parallel vector API instead, e.g. `simulate_tls(group = c("A", "B"), CTmax = c(34, 38), z = c(3, 5))`.

`re_sd` Optional standard deviation of a **random intercept on** `CTmax`. When supplied (with `n_re_groups`), `n_re_groups` group-level deviations `b_g ~ N(0, re_sd)` are drawn and each group's data is generated with `CTmax_g = CTmax + b_g`. This is the data-generating analogue of the `CTmax ~ 1 + (1 | group)` engine; `CTmax` and `z` must be scalars and it cannot be combined with a fixed `group`. The realised deviations are returned in `attr(, "truth")$b`.

`re_sd_z` Optional standard deviation of a **random intercept on** `log(z)`. When supplied (with `n_re_groups`), `n_re_groups` group-level deviations `c_g ~ N(0, re_sd_z)` are drawn on the log scale and each group's data is generated with `z_g = exp(log(z) + c_g)` (a multiplicative spread on `z`). This is the data-generating analogue of the `log_z ~ 1 + (1 | group)` engine. It may be combined with `re_sd` (both intercepts share the one `re_group_name` grouping); the realised log-z deviations are returned in `attr(, "truth")$b_logz`.

`re_sd_low`, `re_sd_logk` Optional standard deviations of **random intercepts on the lower asymptote** `low` **(logit scale)** and on the steepness `log(k)` — the shape-coordinate analogues of `re_sd` / `re_sd_z` (`low_g = plogis(qlogis(low) + d_g)`, `k_g = exp(log(k) + e_g)`; `up` tracks `low` by a fixed head-room fraction). The simulator can generate these deviations alongside `re_sd` / `re_sd_z`; realised deviations are in `attr(, "truth")$b_low` / `$b_logk`.

`n_re_groups` Number of random-effect groups (required with any `re_sd*`).

`re_group_name` Name of the grouping column added to the output for the random-effect mode (default `"colony"`).

`tref` Reference time at which `CTmax` is defined (default `1`).

`seed` Optional integer seed for reproducibility.

Details

The `phi` convention:

For the beta-binomial family, `phi` is the **sum of the Beta shape parameters**: counts are drawn as `prob <- rbeta(a = p * phi, b = (1 - p) * phi)` followed by `rbinom(n, prob)`. The Beta mean is `p` and its variance is `p (1 - p) / (phi + 1)`, so **larger** `phi` **means less overdispersion** and the binomial is recovered as `phi -> Inf`. This matches the engine's parameterisation in `beta_binomial_tls()`.

Value

A base `data.frame` with columns `temp`, `duration`, the true probability `p`, and (when grouped) `group`. The count families (`binomial`, `beta_binomial`) add `total` and `survived`; the `beta` family instead adds a single continuous proportion column `prop`. The data-generating parameters are attached as `attr(, "truth")`.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
head(d)
attr(d, "truth")$CTmax
```

`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`	Boundary 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.), retain the grouping column. Use `fit_4pl(by = "group")` or grouped formulas in `tls_bf()` to estimate category-level effects, or filter first when separate models are scientifically preferable.

Value

A tibble with the standardised columns plus a "tdt_meta" attribute storing temp_mean, duration_unit, random_effects, response_type ("count" or "proportion"), and response_var (the response column name 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")
```

---

tdt-accessors *Accessors for an extract_tdt() result*

Description

Twins of the bayesTLS get_*_summary / get_*_draws accessors. *_summary returns the median + interval tibble; *_draws returns the per-replicate (bootstrap) tibble — the frequentist analogue of posterior draws.

Usage

```
get_z_summary(et)

get_z_draws(et)

get_ctmax_summary(et)

get_ctmax_draws(et)
```

```
get_tcrit_summary(et)

get_tcrit_draws(et)
```

Arguments

et	An `extract_tdt()` result.

Value

A tibble (see `extract_tdt()` for the column contract).

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(dat, family = "binomial", t_ref = 1, quiet = TRUE)
tdt <- extract_tdt(fit, nboot = 10, seed = 1)
get_z_summary(tdt)
get_ctmax_draws(tdt)
```

---

tdt_parameter_table *4PL parameter table (frequentist twin of* `tdt_parameter_table`*)*

Description

Returns the fitted 4PL parameters (`low`, `up`, `k`, `CTmax`, `z`, and `phi` for over-dispersed families) as point estimates with confidence intervals, in bayesTLS's `parameter / [group] / median / lower / upper` shape.

Usage

```
tdt_parameter_table(object, method = NULL, level = 0.95)
```

Arguments

object	A `freq_tls` fit from `fit_4pl()` (or a `profile_tls` fit).
method	Interval method: `"wald"` (default) or `"profile"`.
level	Confidence level (default 0.95).

Value

A tibble with `parameter`, `group`, `median`, `lower`, `upper`.

See Also

tls(), tidy_parameters()

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(dat, family = "binomial", t_ref = 1, quiet = TRUE)
tdt_parameter_table(fit, method = "wald")
```

---

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))
```

tidy_parameters *Tidy the parameters of a fitted freqTLS model*

Description

`tidy_parameters()` returns a broom-style tibble of the natural-scale parameter estimates with optional Wald confidence intervals. The intervals are computed on the internal (unconstrained / link) scale as `estimate +/- z * std.error` and then back-transformed to the natural scale, so they respect each parameter's bounds (for example `z > 0`, `0 < low < up`) and are equivariant under the link. For `CTmax` (identity link) this is the usual symmetric Wald interval.

Usage

```
tidy_parameters(
  fit,
  conf.int = TRUE,
  conf.level = 0.95,
  method = c("wald", "profile")
)
```

Arguments

`fit`	A `profile_tls` fit from `fit_tls()`.
`conf.int`	Logical; include `conf.low` / `conf.high` columns (default `TRUE`).
`conf.level`	Confidence level for the interval (default `0.95`).
`method`	Either `"wald"` (default) or `"profile"`.

Details

With `method = "profile"` the intervals are profile-likelihood confidence intervals (see `confint.profile_tls()`); with `method = "wald"` (default) they are the back-transformed internal-link Wald intervals. The returned shape is identical; only `interval_type` and the interval values differ. A profile that does not close returns `NA` on the open side (never a fabricated bound).

Value

A tibble with one row per natural-scale parameter and the columns `parameter`, `group`, `estimate`, `std.error`, `conf.low`, `conf.high`, `interval_type`, and `scale`. `scale` is the link on which the interval was constructed (`"identity"`, `"log"`, or `"logit"`); `interval_type` is `"wald"` or `"profile"`.

Examples

```
d <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
fit <- fit_tls(d, y = survived, n = total, time = duration, temp = temp,
               family = "binomial", tref = 1)
tidy_parameters(fit)
```

```
tidy_parameters(fit, method = "profile")
```

`tls` *Thermal-load-sensitivity quantities (z, CTmax) with confidence intervals*

Description

The frequentist twin of `bayesTLS::tls()`. Reads a `fit_4pl()` (`freq_tls`) fit and returns the headline thermal-death-time quantities — thermal sensitivity `z` and `CTmax` — as point estimates with confidence intervals, one row per group when the fit is grouped. Uncertainty uses the engine's profile-likelihood intervals by default (or Wald / bootstrap via `method`).

Usage

```
tls(
  object,
  by = NULL,
  params = c("all", "z", "ctmax"),
  target_surv = "relative",
  lethal = FALSE,
  method = NULL,
  level = 0.95,
  nboot = 1000L,
  TC_rate_range = c(0.1, 1),
  seed = NULL
)

tls_z(object, ...)

tls_ctmax(object, ...)

tls_tcrit(object, ...)
```

Arguments

`object`	A `freq_tls` fit from `fit_4pl()` (or a bare `profile_tls` fit from `fit_tls()`).
`by`	Optional name for the grouping column in `$summary`; defaults to the fit's moderator (e.g. the `ctmax`/`z`/`by` factor).
`params`	`"all"` (z and CTmax, the default), `"z"`, or `"ctmax"`.
`target_surv`	Survival threshold for CTmax: `"relative"` (the curve midpoint, the default), `"absolute"` (50% survival), or a number in `(0, 1)` for an LTx. Non-relative thresholds and `lethal` are derived per bootstrap replicate via `extract_tdt()`.
`lethal`	If `TRUE`, also report `T_crit` (the damage-rate-floor critical temperature); uses the bootstrap path.

method	Interval method for the relative path: "profile" (default, from the fit's stored default), "wald", or "bootstrap". Absolute / lethal always use bootstrap.
level	Confidence level (default 0.95).
nboot, TC_rate_range, seed	Passed to extract_tdt() for the bootstrap path (absolute / LTx / lethal).
...	Passed from tls_z() / tls_ctmax() / tls_tcrit() to tls().

Value

A tls object: a list with $summary (a tibble of [<group>,] quantity, median, lower, upper) and $meta.

See Also

fit_4pl(), tls_z(), tls_ctmax(), confint.profile_tls()

Examples

```
raw <- simulate_tls(family = "binomial", CTmax = 36, z = 4, seed = 1)
dat <- standardize_data(
  raw, temp = "temp", duration = "duration",
  n_total = "total", n_surv = "survived"
)
fit <- fit_4pl(
  dat, family = "binomial", t_ref = 1, method = "wald", quiet = TRUE
)
tls(fit)
tls_z(fit)
tls_ctmax(fit)
```

---

tls_bf *Build a freqTLS formula object (brms/drmTMB-style)*

Description

tls_bf() captures the per-sub-parameter formulas that define a freqTLS model and returns them, unevaluated, as a tls_formula object. It is the formula complement to the column interface of fit_tls(): instead of passing bare column names, you write one response formula plus a formula per model sub-parameter.

Usage

```
tls_bf(...)
```

Arguments

...	The response-and-axes formula first (unnamed, with a left-hand side), then sub-parameter formulas keyed by their left-hand side. See the grammar above.

Details

The **first** argument must be the unnamed response-and-axes formula. Its left-hand side names the survival counts, in either the brms idiom `successes | trials(total)` or the `glm` idiom `cbind(successes, failures)`. Its right-hand side names the two thermal-load-sensitivity axes with the tagged markers `time(<duration>)` and `temp(<temperature>)` (order does not matter).

The remaining arguments are sub-parameter formulas keyed by their left-hand side, one of `low`, `up`, `log_k`, `CTmax`, or `log_z`. Any sub-parameter you omit defaults to `~ 1`. Each of `low`, `up`, and `log_k` may carry its **own** design independently — a grouping factor (`low ~ group`), a continuous covariate (`log_k ~ body_size`), or an intercept — and need not share one factor or match the headline-parameter grouping. `CTmax` and `log_z` accept fixed-effect formulas but must produce the same model-matrix columns (for example, use `CTmax ~ group, log_z ~ group`); their supported random- intercept groupings may differ. A single random intercept, `<param> ~ <fixed> + (1 | group)`, is accepted on `CTmax`, `log_z`, `low`, and `log_k` (one grouping factor each, intercept only) – but not on the upper asymptote `up`, for which the compiled objective has no random-intercept term. Putting the same grouping factor on two or more of them fits independent variances (no correlation term) and warns.

Value

A `tls_formula` object: a list with the captured response formula, the named sub-parameter formulas, and the calling environment.

Parser provenance

The shape of the parser (variadic capture via `substitute()`, a per-entry formula walk, and random-bar detection) is adapted from drmTMB's `drm_formula()` / `parse_drm_formula_entry()` (GPL-3); see `inst/COPYRIGHTS`. freqTLS writes its own grammar (the `time()` / `temp()` axis markers, the five fixed sub-parameter handles, and the package's supported random-effect grammar).

See Also

`fit_tls()`, which accepts either a `tls_formula` or the column interface.

Examples

```
tls_bf(
  survived | trials(total) ~ time(duration) + temp(temp),
  CTmax ~ life_stage,
  log_z ~ life_stage
)
# cbind() response idiom, ungrouped:
tls_bf(cbind(survived, died) ~ time(duration) + temp(temp))
```

`tls_family` *Response families for thermal-load-sensitivity models*

Description

`binomial_tls()` and `beta_binomial_tls()` describe the count response distribution for `fit_tls()`, and `beta_tls()` the continuous-proportion response in `(0, 1)` (e.g. PSII operating efficiency or relative chlorophyll fluorescence). All three model survival as a four-parameter logistic function of log10 duration; the beta-binomial and beta families add a dispersion parameter `phi`.

Usage

```
binomial_tls()

beta_binomial_tls()

beta_tls()
```

Details

The `phi` convention for the beta-binomial family is the **sum of the Beta shape parameters**: for fitted survival probability `p`, counts are Beta-Binomial with shapes `a = p * phi` and `b = (1 - p) * phi`. Larger `phi` means *less* overdispersion (the binomial is recovered as `phi` grows). This matches the simulation convention in `simulate_tls()` and differs from the precision/size parameterisations used by some other packages. The `beta` family uses the **same shapes** for the continuous proportion, `y ~ Beta(p * phi, (1 - p) * phi)`, so `phi` carries the identical meaning and a larger `phi` again means a tighter response around the fitted curve.

Value

A `tls_family` object: a list with `family`, `family_code` (0 binomial, 1 beta-binomial, 2 beta), and `links` for the natural-scale parameters.

Examples

```
binomial_tls()
beta_binomial_tls()
beta_tls()
```

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`. 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 + 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.
`time_multiplier`	Multiplier from the Stage-1 duration unit to minutes (e.g. 60 if durations are in hours). Default 1.
`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).

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`.

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_lethal *Zebrafish lethal thermal-death-time data across life stages*

Description

Lethal-TDT trials for zebrafish (*Danio rerio*) at three life stages. Built from the raw daily survival sheet by summing the per-day morning/afternoon mortality counts into one death count per trial and dropping excluded rows. One row per assay trial. The model-ready frame for Case Study 2.

Usage

```
zebrafish_lethal
```

Format

A data frame with 323 rows and 7 variables:

assay_temp Assay temperature (degrees C).

duration_h Exposure duration (hours).

n_total Number of individuals in the trial.

n_surv Number that survived.

n_dead Number that died (`n_total - n_surv`).

life_stage Life stage, a factor with levels `young_embryos`, `old_embryos`, `larvae`.

Date_experiment Experiment date (grouping factor).

Source

Zebrafish lethal-TDT assay across life stages (Case Study 2), obtained from the **bayesTLS** package distribution by Noble, Arnold, and Pottier (2026), licensed CC BY 4.0. freqTLS removed excluded trials, aggregated daily mortality counts, and derived survivors as documented above. Raw file: `system.file("extdata", "data_lethal_TDT_zebrafish.csv", package = "freqTLS")`.

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 profile-likelihood confidence intervals on those direct parameters.

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 = "freqTLS")`.

Examples

```
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
```

Index