

Package: rTFT (via r-universe)

October 16, 2026

Title Hazard Models for Thermal Failure Time Data

Version 0.1.1

Description Fits discrete-time survival models to the times at which individual organisms fail under thermal exposure (e.g., loss of equilibrium, knockdown or death), using Bayesian inference in 'Stan' through 'brms'. The hazard is the product of a failure rate that depends on temperature, following Kingsolver and Umbanhowar (2018) <[doi:10.1242/jeb.167858](https://doi.org/10.1242/jeb.167858)>, and a baseline that may depend on the time under exposure. Failure times may be known-time, interval-censored or both, and may come from static assays, ramping assays or both. The failure rate may depend on covariates, group-level effects and other stressors such as dissolved oxygen. Static and dynamic critical thermal maxima, thermal death time curves and thermal tolerance landscapes are derived from the fitted model with their posterior uncertainty. Models can be compared by cross-validation that leaves out whole individuals or trials, averaged, and used to predict survival along a record of temperature and of any co-stressors recorded with it.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

URL <https://eogmartins.github.io/rTFT/>,
<https://github.com/eogmartins/rTFT>,
<https://github.com/eogmartins/rTFT/discussions>

BugReports <https://github.com/eogmartins/rTFT/issues>

Depends R (>= 4.1.0)

Imports brms (>= 2.20.0), cli, dplyr, ggplot2 (>= 3.5.2), grDevices,
loo, purrr, stats, tibble, utils

Suggests knitr, plotly, posterior, quarto, testthat (>= 3.0.0)

VignetteBuilder quarto

SystemRequirements Quarto command line tool
(<<https://github.com/quarto-dev/quarto-cli>>).

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libicu-dev

Repository <https://eogmartins.r-universe.dev>

Date/Publication 2026-10-16 16:43:02 UTC

RemoteUrl <https://github.com/eogmartins/rTFT>

RemoteRef v0.1.1

RemoteSha e659624c8cd6f7554a95fae3a992575369f62663

Contents

assess_ramp_linearity	3
average_dctmax	4
average_survival	6
average_tdt_ttl	8
brms_log_h0	10
brms_log_M	10
build_hazard_data	11
build_hazard_prior	14
build_hazard_spec	17
check_survival	19
cloglog_bern_family	21
costressor	21
derive_dctmax	25
derive_tdt_ttl	27
fit_hazard	29
h0_catalogue	31
H0_of	33
H0inv_of	34
hazard_mixed_family	35
hazard_residuals	35
hazard_weights	37
kfold_hazard	38
loo_hazard	40
M_catalogue	41
M_draws	43
plot_check	44
plot_residuals	45
plot_risk	46
plot_survival	47

<i>assess_ramp_linearity</i>	3
------------------------------	---

plot_tdt	49
plot_ttl	50
plot_ttl_3d	52
predict_survival	53
ramp_assay	57
ramp_path	58
static_assay	59
stream_record	60
Index	62

<i>assess_ramp_linearity</i>	<i>Assess how linear a logged temperature trace is</i>
------------------------------	--

Description

A pre-fitting diagnostic with two uses. As quality control on the ramp, it fits $\text{temp} \sim \text{time per path}$ and reports the realised start and rate, the fit ($r_squared$), and, given nominal, the $rate_bias$ against the setpoint, so you can see whether the controller held a linear ramp at the intended rate. And as a check on whether a dense or noisy log can be simplified for fitting: a finely sampled temperature series turns every reading into an at-risk-interval boundary in [build_hazard_data](#), so a linear one can be replaced by a clean two-point ramp from [ramp_path](#) (exact under the linear interpolation, free of sensor jitter, and far fewer rows). A faithful series is otherwise best fitted directly; this only judges that faithfulness.

Usage

```
assess_ramp_linearity(path, nominal = NULL)
```

Arguments

path	A data frame with path_id, time and temp – the logged temperature trace(s).
nominal	An optional named numeric of setpoint rates (names matching path_id). When given, adds rate_nominal and rate_bias (realised minus nominal), the controller’s rate error.

Details

As a rule of thumb, a series is linear enough to simplify when its deviations (and especially max_dev_hot , the worst deviation in the upper temperature quartile, where the failure rate $M(T)$ is steepest) are small relative to $\log(10) / b$, the number of degrees for a tenfold change in an exponential $M(T)$: half that distance already multiplies $M(T)$ by $\sqrt{10}$, about threefold.

That distance is not always the z that [derive_tdt_ttl\(\)](#) reports. z is the number of degrees for a tenfold change in the time to failure, $k \log(10) / b$, with k the shape of a Weibull baseline. The two are the same under the memoryless baseline, where $k = 1$, and differ by the factor k under a Weibull one. Judge the deviations against $\log(10) / b$, not against a reported z .

Value

A tibble, one row per path_id, with n (the number of readings), the realised T_start and rate, r_squared, rmse, max_abs_dev, and max_dev_hot (the worst absolute deviation in the hot quartile); plus rate_nominal and rate_bias when nominal is supplied.

Examples

```
# A logged temperature series close to a 0.3 degree/min ramp, with a small
# mid-run wobble.
path <- data.frame(path_id = "r1", time = 0:20,
                   temp = 20 + 0.3 * (0:20) + c(rep(0, 8), 0.4, 0.5, 0.3, rep(0, 10)))
assess_ramp_linearity(path)

# With the setpoint rate, to see the controller's rate bias.
assess_ramp_linearity(path, nominal = c(r1 = 0.3))
```

average_dctmax	<i>Dynamic critical thermal maximum averaged over several fitted models</i>
----------------	---

Description

Derives the dynamic critical thermal maximum dCTmax under a ramp as `derive_dctmax()` does, averaged over several models fitted to the same data. Each model's posterior draws of dCTmax are pooled in proportion to its weight, and the pooled draws are summarised, so the median and credible interval reflect both the uncertainty within each model and the uncertainty about which model predicts best.

Usage

```
average_dctmax(
  fits,
  weights,
  rate,
  T_start,
  newdata = NULL,
  p = 0.5,
  levels = NULL,
  prob = 0.95,
  n_draws = 1000,
  T_max = NULL
)
```

Arguments

<code>fits</code>	A named list of brmsfits, models fitted with <code>fit_hazard()</code> to the same data.
<code>weights</code>	The weights from <code>hazard_weights()</code> , whose model names are names of fits. Only the models it kept are averaged.

rate	A numeric vector of ramp rates, in degrees per unit of time of the data the model was fitted to (e.g. degrees C per minute), all positive.
T_start	A numeric vector of starting temperatures, one per rate or a single one for all.
newdata	A data frame of covariate values, one row per setting at which to derive. It must hold every covariate in the model's parameter formulas. May be NULL for a model without covariates.
p	A numeric vector of the fractions of individuals failing at which sCTmax and the thermal death time curves are derived, each strictly between 0 and 1.
levels	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. c(DO = 6). Required when the model has co-stressors.
prob	A single number strictly between 0 and 1, the coverage of the credible intervals.
n_draws	A single number, the number of posterior draws in the pool. Each model contributes its share of them, evenly spaced through its posterior. Where a model holds fewer draws than its share, the pool is made smaller so that the shares stay in proportion to the weights, and a message says so.
T_max	A single number, the temperature up to which the ramp is followed. NULL (the default) takes 15 degrees above the warmest temperature the models were fitted to.

Details

The models usually agree closely on dCTmax under a ramp like those they were fitted to, and less under ramps much slower or faster, where the failure-rate and baseline functions are extrapolated.

Value

A list of two tibbles: `dctmax`, in the layout of `derive_dctmax()`'s result, and `weights`, the table of weights with the number of draws each model contributed.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
  trial = rep(c("a", "b"), each = 10),
  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
  checks = checks)
fit_mem <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
fit_weibull <- fit_hazard(long, "exp", baseline = "weibull",
  dpar_formulas = list(loga = ~ 1 + mass))
fits <- list(memoryless = fit_mem, weibull = fit_weibull)

w <- hazard_weights(lapply(fits, loo_hazard))
average_dctmax(fits, w, rate = c(0.1, 0.3), T_start = 14,
  newdata = data.frame(mass = 10))$dctmax
```

```
## End(Not run)
```

average_survival	<i>Survival along a record of temperature, averaged over several fitted models</i>
------------------	--

Description

Predicts survival along a record as `predict_survival()` does, averaged over several models fitted to the same data. Each model's posterior draws are pooled in proportion to its weight and the pool is summarised, so the credible intervals reflect both the uncertainty within each model and the uncertainty about which model holds. This matters most where the record lies outside the temperatures of the assay, where failure-rate functions that fit the data equally well can predict very differently (see `predict_survival()`).

Usage

```
average_survival(
  fits,
  weights,
  record,
  newdata = NULL,
  levels = NULL,
  by = NULL,
  prob = 0.95,
  n_draws = 1000,
  grid_dT = 0.1
)
```

Arguments

<code>fits</code>	A named list of <code>brmsfits</code> , models fitted with <code>fit_hazard()</code> to the same data.
<code>weights</code>	The weights from <code>hazard_weights()</code> , whose model names are names of <code>fits</code> . Only the models it kept are averaged.
<code>record</code>	A data frame with the columns <code>time</code> and <code>temp</code> , the record of temperature through time, in the time unit and temperature scale of the data the model was fitted to, with times increasing and no gaps. It may also hold the column of a co-stressor and the column named in <code>by</code> .
<code>newdata</code>	A data frame of covariate values, one row per setting at which to derive. It must hold every covariate in the model's parameter formulas. May be <code>NULL</code> for a model without covariates.
<code>levels</code>	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. <code>c(D0 = 6)</code> . Required when the model has co-stressors.

by	A character string, the name of a column of record giving the period each reading belongs to (e.g., a day). When given, the result has each period's share of the hazard accrued over the record. An interval between two readings belongs to the period of the first.
prob	A single number strictly between 0 and 1, the coverage of the credible intervals.
n_draws	A single number, the number of posterior draws in the pool. Each model contributes its share of them, evenly spaced through its posterior. Where a model holds fewer draws than its share, the pool is made smaller so that the shares stay in proportion to the weights, and a message says so.
grid_dT	A single positive number, the largest change in temperature within a step along the record, in degrees. Between two readings the temperature is taken to change linearly, and the hazard is accrued in steps over which it changes by at most this much, since the failure rate at the midpoint of a step stands for the whole of it (see <i>The time grid</i> in build_hazard_data). Defaults to 0.1; Inf takes one step between consecutive readings.

Details

The weights say how well each model predicts failures within the assay. They do not say which model extrapolates better below it, which the data cannot tell.

Value

The list [predict_survival\(\)](#) returns, with weights, the table of weights with the number of draws each model contributed. In coverage, the shares of the risk are those of the pooled draws.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                        checks = checks)
fit_mem <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
fit_weibull <- fit_hazard(long, "exp", baseline = "weibull",
                        dpar_formulas = list(loga = ~ 1 + mass))
fits <- list(memoryless = fit_mem, weibull = fit_weibull)
# a two-day record read every hour, with a daily cycle
hours <- 0:48
record <- data.frame(time = hours * 60, day = pmin(hours %% 24, 1),
                    temp = 24 + 3 * sin(2 * pi * (hours - 9) / 24))

w <- hazard_weights(lapply(fits, loo_hazard))
average_survival(fits, w, record, newdata = data.frame(mass = 10))$mortality

## End(Not run)
```

average_tdt_ttl	<i>Thermal tolerance averaged over several fitted models</i>
-----------------	--

Description

Derives the static critical thermal maximum sCTmax, the thermal death time curves and the thermal tolerance landscape as `derive_tdt_ttl()` does, averaged over several models fitted to the same data. Each model's posterior draws of a quantity are pooled in proportion to its weight, and the pooled draws are summarised, so the median and credible interval reflect both the uncertainty within each model and the uncertainty about which model holds.

Usage

```
average_tdt_ttl(
  fits,
  weights,
  newdata = NULL,
  p = c(0.1, 0.5, 0.9),
  t_ref,
  Ta = NULL,
  tau = NULL,
  levels = NULL,
  prob = 0.95,
  n_draws = 1000
)
```

Arguments

<code>fits</code>	A named list of brmsfits, models fitted with <code>fit_hazard()</code> to the same data.
<code>weights</code>	The weights from <code>hazard_weights()</code> , whose model names are names of fits. Only the models it kept are averaged.
<code>newdata</code>	A data frame of covariate values, one row per setting at which to derive. It must hold every covariate in the model's parameter formulas. May be NULL for a model without covariates.
<code>p</code>	A numeric vector of the fractions of individuals failing at which sCTmax and the thermal death time curves are derived, each strictly between 0 and 1.
<code>t_ref</code>	A single positive number, the reference exposure time for sCTmax, in the time unit of the data the model was fitted to (for example 60 for one hour when times are in minutes). It has no default, because the fit does not record that unit.
<code>Ta</code>	An optional numeric vector of the constant temperatures for the thermal death time curves and the tolerance landscape. NULL (the default) takes 50 evenly spaced across the temperatures the data cover.
<code>tau</code>	An optional numeric vector of the elapsed exposure times for the tolerance landscape, in the time unit of the data, none negative. NULL (the default) takes 50 spaced evenly on a log scale across the times the data cover. A time of zero is allowed, and gives a survival of 1, so that a survival curve can start there.

levels	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. <code>c(D0 = 6)</code> . Required when the model has co-stressors.
prob	A single number strictly between 0 and 1, the coverage of the credible intervals.
n_draws	A single number, the number of posterior draws in the pool. Each model contributes its share of them, evenly spaced through its posterior. Where a model holds fewer draws than its share, the pool is made smaller so that the shares stay in proportion to the weights, and a message says so.

Details

The parameters are not averaged. The parameters of different failure-rate functions are different quantities under the same names, so a pooled `a` or `b` would mean nothing; they are reported for each model by `derive_tdt_ttl()`.

The result has the layout of `derive_tdt_ttl()`'s, without its `params` table, so `plot_tdt()`, `plot_ttl()` and `plot_ttl_3d()` draw it as they draw a single model's.

Value

A list of four tibbles: `sctmax`, `tdt_curves` and `ttl_surface` as from `derive_tdt_ttl()`, and `weights`, the table of weights with the number of draws each model contributed.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                          checks = checks)
fit_mem <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
fit_weibull <- fit_hazard(long, "exp", baseline = "weibull",
                          dpar_formulas = list(loga = ~ 1 + mass))
fits <- list(memoryless = fit_mem, weibull = fit_weibull)

w <- hazard_weights(lapply(fits, loo_hazard))
avg <- average_tdt_ttl(fits, w, newdata = data.frame(mass = 10), t_ref = 60)
plot_tdt(avg)

## End(Not run)
```

brms_log_h0	<i>log h0(tau) for each baseline function, as a brms expression string</i>
-------------	--

Description

Each entry is the expression string of $\log h_0(\tau)$ for one function of [h0_catalogue](#), which brms uses to build the model formula. The strings are written in terms of the parameter names and the data columns `log_tau` (the log of the row's elapsed exposure time), for `weibull` `tau_ref` (the reference time its scale is fixed at) and, for `gompertz`, `tau` itself (the elapsed exposure time: the midpoint for an interval row, or the time of a known-time failure). `mem` is the empty string, because a memoryless baseline contributes nothing: $h_0(\tau) = 1$ and the hazard is $M(T)$ alone.

Usage

```
brms_log_h0
```

Format

A named list, one expression string per baseline function of [h0_catalogue](#).

Details

Each string is the baseline **hazard** on the log scale, not the density. For `llogis` the two differ by one $\log 1p$ term: the hazard has a single $-\log 1p(\dots)$, the density would have two.

brms_log_M	<i>log M(T) for each failure rate function, as a brms expression string</i>
------------	---

Description

Each entry is the expression string of $\log M(T)$ for one function of [M_catalogue](#), which brms uses to build the model formula. The strings are written in terms of the parameter names and the data columns `Ta` (the row's temperature: the interval midpoint, or the temperature at the moment of a known-time failure) and `Tr` (the reference temperature of the model; see [M_catalogue](#)). The threshold functions add a small constant inside a logarithm so it stays finite at the critical temperature. The constant is written 10^{-12} and not $1e-12$, because brms spaces out the minus sign of a number in scientific notation when it writes the Stan program ($1e - 12$), which Stan does not accept. The temperature column is named `Ta` rather than `Kingsolver` and `Umbanhowar's T`, because `T` is an alias for `TRUE` in R and a column of that name would silently shadow it in the data-handling code. The threshold functions also reference `k_sw`, the softening steepness, taken from the `k_sw` field of [M_catalogue](#) when the model formula is built by brms. The strings are the single shared definition that fitting, simulation and reporting all evaluate.

Usage

```
brms_log_M
```

Format

A named list, one expression string per failure rate function of [M_catalogue](#).

build_hazard_data	<i>Build the row-per-interval table a failure-rate model is fitted to</i>
-------------------	---

Description

Reshapes one row per individual, together with a temperature trajectory, into one row per individual and at-risk interval. An interval-censored failure gives the failure its whole censoring interval; a known-time one truncates the final at-risk interval at the recorded time. The cumulative hazard is integrated numerically over the at-risk intervals, with the failure-check times serving as the integration nodes. For a known-time failure, the fit integrates on the checks (its final interval ending exactly at the failure time) and no finer sub-grid is required but can be provided.

Usage

```
build_hazard_data(
  ind,
  path,
  observation = c("interval", "continuous", "mixed"),
  covariates = character(0),
  Tr = NULL,
  tau_ref = NULL,
  checks = NULL,
  grid_dt = NULL,
  grid_dT = 0.1,
  costressors = list()
)
```

Arguments

ind	A data frame with one row per individual. Required: id, path_id (key into path), event (1 = failed, 0 = right-censored), plus the covariate columns referenced in the model. Timing depends on how the failure was observed: interval-censored individuals have t_upper (the check at which failure was found, or a survivor's last observed check); known-time ones have t_event (the known time of failure or censoring). For observation = "mixed" the data also needs an obs_type column per individual ("interval" or "continuous"), and each type supplies its own timing column. Every individual is taken to be at risk from time 0 of its path: the table has no entry time, so an individual that joins an exposure after it began cannot be represented.
path	A data frame giving the temperature trajectory for each path. Required columns: path_id, time, temp. The function interpolates these points linearly in time, so a minimal two-point path (start and end) encodes a linear ramp. Interpolation uses the full series, so a temperature data logger whose records are finer than

the checks still supplies the true temperature at each interval midpoint (never a check-to-check straight line); the checks control only the integration grid, not the temperature. Temperatures outside the recorded time range are held flat at the nearest endpoint rather than returned as NA, because the at-risk grid (the checks, the exact failure time, and the $t = 0$ start) need not sit strictly inside the logged times: a check or failure time can fall just past the last reading, or the first interval can begin before the first reading, and an NA temperature there would propagate into T_a and the linear predictor.

observation	How each failure time was observed. One of "interval" (every failure known only to fall within a censoring interval), "continuous" (every failure observed at a known time), or "mixed" (both kinds in one table, each individual's kind given by its <code>obs_type</code> column). This determines which response family the table is fitted with, and whether the exact flag is added; the response column is failed in every mode.
covariates	A character vector of column names in <code>ind</code> to carry through onto every row of the output table, so they are available in the model formula as fixed-effect predictors or as grouping factors for random effects. A trial identifier is the common case: include it here, then place a trial-level random intercept on a parameter in <code>build_hazard_spec</code> , e.g. <code>dpar_formulas = list(loga = ~ 1 + (1 trial))</code> . The individual identifier <code>id</code> is always kept, so an individual-level grouping term needs no entry here.
Tr	The reference temperature, written to a constant <code>Tr</code> column so the functions can form $T_a - Tr$. It is a single model reference, not the per-trial starting temperature (see M_catalogue); the per-trial starts live in the temperature trajectories (<code>path</code>). For the forms that take a logarithm or fractional power of $T_a - Tr$ (<code>zeroexp</code> , <code>power</code> , <code>zeropower</code>) it is the temperature at which the rate is zero (or, for power, falls to its baseline), so it must be given, at or below the coldest temperature in the data. Otherwise it defaults (NULL) to the median temperature at which failures were observed, which places <code>loga</code> where the data are and keeps it from being tied to the thermal sensitivity <code>b</code> ; a reference far from the failures makes the two hard to tell apart. It must be set to a common value across datasets when comparing <code>loga</code> , whose meaning is anchored at <code>Tr</code> .
tau_ref	A single positive number, the reference time, in the time unit of the data, written to a constant <code>tau_ref</code> column. The scale of the <code>weibull</code> baseline is fixed at it (see h0_catalogue), and the default priors of the parameters that depend on the time unit are scaled by it (see build_hazard_prior). It defaults (NULL) to the median time at which failures were observed, which places <code>loga</code> where the data are, as the default <code>Tr</code> does. A reference time much longer than the times at which individuals fail makes <code>loga</code> an extrapolation in time that depends heavily on the shape of the baseline. Like <code>Tr</code> , set it to a common value across datasets when comparing <code>loga</code> under a <code>weibull</code> baseline, whose meaning is anchored at it.
checks	An optional data frame with <code>path_id</code> , <code>time</code> giving the failure-check schedule per path (i.e., the times at which each individual was inspected for failure). Defaults to the distinct path times (<code>logger reading = check</code>). Time 0, the start of the exposure, is added to every schedule when it is not listed. An interval-censored <code>t_upper</code> must be one of its path's checks: a time between two checks, or past the last one, is refused. As noted above, these times double as the integration nodes

	for the cumulative hazard, so their spacing sets the midpoint-rule resolution. For a known-time failure, a regular grid can be used instead (see <code>grid_dt</code>).
<code>grid_dt</code>	An optional numeric giving a fixed time step (in the units of <code>path\$time</code>). If set, a known-time failure is followed on a regular grid of this step, from $t = 0$ to the failure time, instead of on checks. Defaults to <code>NULL</code> (the checks, or the readings of <code>path</code>). Either way the intervals are divided further where <code>grid_dT</code> requires it.
<code>grid_dT</code>	A single positive number, the largest change in temperature allowed within an interval at risk, in degrees. An interval over which temperature changes by more is divided into equal parts (see <i>The time grid</i>). Defaults to 0.1; <code>Inf</code> keeps the intervals as given.
<code>costressors</code>	An optional list of <code>costressor</code> objects declaring non-temperature stressors (dissolved oxygen and the like). Each adds a standardised column <code>z_<name></code> , read from its own trajectory (interpolated to the interval midpoints) or, for a stressor held fixed within a trial, from a column of <code>ind</code> . Defaults to none.

Value

A long-format data frame, one row per individual and at-risk interval, holding the identifiers `id` and `path_id`; the model columns `Ta`, `Tr`, `tau`, `log_tau`, `tau_ref`, `exposure`, `log_exposure`; the carried covariates; one `z_<name>` column per co-stressor; the 0/1 response column `failed`; and, for mixed observation, the integer exact flag the family branches on.

The time grid

Within each interval at risk the hazard is taken as constant, at its value at the midpoint of the interval. That is accurate while temperature changes little within the interval, because the failure rate rises steeply with temperature. With a thermal sensitivity of $b = 1.7$ per degree, for example, the rate is $\exp(1.7 \times 0.1) = 1.19$ times higher at the end of an interval that spans 0.1 degrees than at its start, and $\exp(1.7 \times 0.5) = 2.3$ times higher across one that spans half a degree; the hazard accrued over the interval, taken at its midpoint, is then understated by about $(b \text{ dT})^2 / 24$, that is by 0.1% over a tenth of a degree and by 3% over half a degree, an error that grows with the square of the change in temperature. So any interval over which temperature changes by more than `grid_dT` is divided into equal parts, each with its own row, and a message says how many rows resulted.

Two cases are treated apart. A failure at a known time needs the hazard at the moment it happened, and the midpoint of the interval it fell in is earlier and, in a ramp, cooler; so that interval is closed with a very short one ending at the failure. This is the larger of the two errors: without the closing interval, a ramp followed in intervals of a minute underestimated $b = 1.7$ by about 4% in a simulation of eight large assays, and with it b was recovered whatever the length of the intervals. A failure known only to the interval between two checks keeps that interval whole, since where in it the individual failed is not known, and its contribution is the probability of failing anywhere in it. A warning is given when temperature changes by more than half a degree within such an interval, so that the estimates can be interpreted with that in mind.

When neither checks nor `grid_dt` is given for failures at known times, the intervals are those between the readings of `path`, divided as above, and into at least 20 per individual, so that a trajectory given by its two ends is still followed closely.

Examples

```
# A single linear ramp encoded as two points: 20 -> 35 degrees over 60 minutes,
# with failure checked for every 10 minutes.
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))

# Interval-censored: one failure found at the 30-minute check, one survivor.
ind <- data.frame(id = 1:2, path_id = "A", event = c(1, 0), t_upper = c(30, 60))
build_hazard_data(ind, path, "interval", checks = checks)

# Known-time: the failure time itself, integrated on the same checks.
ind_c <- data.frame(id = 1, path_id = "A", event = 1, t_event = 42)
build_hazard_data(ind_c, path, "continuous", checks = checks)

# A finer, regular integration grid instead of the checks (known-time only);
# the last interval still closes exactly on the failure time.
build_hazard_data(ind_c, path, "continuous", grid_dt = 5)

# Mixed: each individual carries its kind in obs_type; the unused timing
# column is left NA.
ind_m <- data.frame(id = 1:2, path_id = "A", event = 1,
  obs_type = c("interval", "continuous"),
  t_upper = c(30, NA), t_event = c(NA, 42))
build_hazard_data(ind_m, path, "mixed", checks = checks)

# A covariate carried through to every row, and a shared reference Tr.
ind$mass <- c(12.1, 9.8)
build_hazard_data(ind, path, "interval", covariates = "mass", Tr = 15, checks = checks)
```

build_hazard_prior	<i>Weakly-informative default priors for a failure-rate hazard model</i>
--------------------	--

Description

Supplies a sensible starting prior for the model `build_hazard_spec` assembles, keyed to each parameter's intercept and unaffected by whatever covariates the `dpar_formulas` add. For every parameter the chosen functions need, it sets a weakly-informative prior on the intercept (the parameter's value at the reference temperature and start of exposure), a generic $\text{Normal}(0, 1)$ on any covariate slopes, and an $\text{Exponential}(2)$ on the standard deviation of any random effect. The slope and random-effect priors are added only when the parameter actually has those terms, so `brms` raises no "no parameter" warning. Override the centre of any parameter through `intercept_priors`, or the shared slope and random-effect standard-deviation priors through `slope_prior` and `sd_prior`; or replace the whole set with your own prior when fitting with `fit_hazard()`.

Usage

```
build_hazard_prior(
  m_form,
  baseline = "mem",
```

```

dpar_formulas = list(),
intercept_priors = list(),
slope_prior = "normal(0, 1)",
sd_prior = "exponential(2)",
costressors = list(),
long
)

```

Arguments

m_form	A character string denoting a key of M_catalogue (e.g. "exp" or "exp_threshold").
baseline	A character string denoting a key of h0_catalogue : "mem" (default), "weibull", "llogis" or "gompertz".
dpar_formulas	A named list of one-sided formulas, one per parameter, as in build_hazard_spec . A parameter's fixed and random terms decide which slope and standard-deviation priors are added. A formula without an intercept ($\sim 0 + \text{acclim}$) gives the parameter one coefficient per level in place of the intercept, and each of them takes the intercept's prior.
intercept_priors	A named list of prior strings, one per parameter, that replace the default intercept (centre) for those parameters, e.g. <code>list(Tc = "normal(26, 4)")</code> . A parameter left out keeps its default; a name that is not a parameter of the chosen functions is ignored with a warning.
slope_prior	A single prior string for every covariate slope (a population-level effect beyond the intercept), applied to each parameter that has one. Defaults to "normal(0, 1)".
sd_prior	A single prior string for every random-effect standard deviation, applied to each parameter that has one. Defaults to "exponential(2)".
costressors	An optional list of costressor objects, as in build_hazard_spec . Each adds an intercept prior for its main effect and, when the temperature interaction is kept, its interaction coefficient. Defaults to none.
long	A data frame, the row-per-interval table from build_hazard_data that the prior is for. The defaults read from it the reference time <code>tau_ref</code> , the reference temperature <code>Tr</code> and the temperatures at which failures were observed. It has no default. fit_hazard() passes its own table when it builds the default prior.

Details

The defaults take three things from the data in `long`, so that they say the same thing whatever the organism, the time unit and the length of the exposures: the reference time `tau_ref`, the reference temperature `Tr`, and the median temperature at which failures were observed, `T_f`. They are weakly-informative and meant to be changed to suit your study:

- `loga` – log amplitude. Stated where the failures are: at `T_f`, the cumulative hazard reached over `tau_ref` is centred at $\exp(-1)$, about 0.37, with a standard deviation of 2 on the log scale (about 0.007 to 20). With the memoryless and Gompertz baselines `a` is a rate per unit time, so that centre is $-1 - \log(\text{tau_ref})$; with the Weibull and log-logistic it is -1 . The prior is then carried from `T_f` back to the temperature `a` refers to (see below).

- $\text{logb} \sim \text{normal}(-0.7, 1)$ – log temperature sensitivity
- logc – log baseline term of the power function (see below)
- $T_c \sim \text{normal}(T_f, 3)$ – critical temperature, centred where failures were observed
- $\text{logshape} \sim \text{normal}(0, 1)$ – log Weibull / log-logistic shape
- $\text{logscale} \sim \text{normal}(\log(\text{tau_ref}), 1)$ – log log-logistic scale
- $\text{gam} \sim \text{normal}(0, 3 / \text{tau_ref})$ – Gompertz rate: the baseline hazard changes by a factor of $\exp(\text{gamma tau_ref})$ across the reference time, within about 400-fold either way with 95% probability. It is centred at zero, the memoryless case, so unless the data show a trend the fit stays near a flat baseline.

with $\text{normal}(0, 1)$ on any covariate slope and $\text{exponential}(2)$ on any random-effect standard deviation. Each co-stressor in `costressors` adds a $\text{normal}(0, 1)$ on its main effect `c<name>` and a tighter $\text{normal}(0, 0.5)$ on its temperature interaction `dT<name>`, both overridable through `intercept_priors`.

The constants in these priors (the centre of $\exp(-1)$ and the standard deviation of 2 for the amplitude, the centre of -0.7 for log b, the rate of 2 for the group standard deviations, the 0.5 for an interaction) are judgements of what is plausible for a thermal failure assay, not values estimated from data, and the centres that are read from long make the prior depend on the data in a weak way, as a standard deviation of 2 on the log scale is wide. The prior on a covariate slope is per unit of the covariate, so a covariate measured in large units (body mass in grams, say) should be centred and scaled, or the slope prior widened, before $\text{normal}(0, 1)$ is taken as weak. With few groups, the prior on a group standard deviation can matter for predictions of new groups (which cross-validation by group scores, see `kfold_hazard()`); a sensitivity check against $\text{exponential}(1)$ is cheap insurance.

Value

A `brmsprior`.

The prior on a when the reference is far from the failures

For the rates anchored at the reference temperature (`exp`, `zeroexp`, `zeropower`, `power`), a is the rate at T_r , or its scale there, and T_r may lie a distance $d = T_f - T_r$ from where failures were observed. The log rate at T_f is then loga plus a term in b and d ($b d$ for `exp`), so loga is the log rate at T_f less that term, and it inherits the uncertainty in b across the distance:

- its centre is the centre at T_f less the term at the prior median of b ;
- its variance is 2^2 plus the square of (the change in the term per unit of b , times the prior standard deviation of b).

For `exp` this is $\text{normal}(\text{centre} - 0.5 d, \text{sqrt}(4 + (1.07 d)^2))$: the usual prior when the reference is at the failures ($d = 0$), and a much wider one 14 degrees below them. A prior of standard deviation 2 on loga at a distant reference would in effect be a tight prior on b , and biases it. The threshold functions are anchored at T_c , not T_r , and take the centre at T_f unchanged. The calculation uses the default prior on logb ; if you change that prior and keep a distant reference, set loga to match.

The baseline term c of the power function

In $M(T) = c + a(T - T_r)^b$, c is the rate that remains at the reference temperature: failure that happens whatever the warming. Its default is centred so that about 1% of individuals would fail over τ_{ref} for reasons unrelated to temperature, $\text{normal}(\log(0.01) - \log(\tau_{ref}), 2)$ with the memoryless or Gompertz baseline and $\text{normal}(\log(0.01), 2)$ otherwise, which covers background failure from about 0.02% to 40%. The 1% is a judgement, not a measured value. Where controls were held at the reference temperature, set it from them: if a fraction q of controls failed over a time t , c is about $-\log(1 - q) / t$, so for instance `intercept_priors = list(logc = "normal(log(-log(1 - 0.02) / 120), 1)")` for 1 in 50 failing over 120 minutes.

Examples

```
# A ramp from 20 to 35 degrees over 60 minutes, with two failures.
path <- ramp_path("A", T_start = 20, rate = 0.25, t_max = 60)
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:3, path_id = "A", event = c(1, 1, 0),
                  t_upper = c(30, 40, 60), acclim = c(10, 15, 15))
long <- build_hazard_data(ind, path, "interval", covariates = "acclim",
                         checks = checks)

# Default weakly-informative priors for an exponential-threshold + Weibull model.
build_hazard_prior("exp_threshold", baseline = "weibull", long = long)

# Override a couple of priors (a cooler species, a tighter amplitude) and narrow
# the covariate-slope prior; every other parameter keeps its default.
build_hazard_prior("exp_threshold", baseline = "weibull",
  dpar_formulas = list(loga = ~ 1 + acclim),
  intercept_priors = list(Tc = "normal(26, 4)", loga = "normal(-6, 1.5)"),
  slope_prior = "normal(0, 0.5)", long = long)
```

build_hazard_spec	<i>Assemble the brms formula and family for a failure-rate hazard model</i>
-------------------	---

Description

Builds a nonlinear brms formula whose linear predictor is the log failure rate $\log M(T)$, plus (unless the baseline is memoryless) the log baseline hazard $\log h_0(\tau)$, plus the log exposure of each at-risk interval. The failure rate and baseline are named by their keys in [M_catalogue](#) and [h0_catalogue](#), and their log-scale expression strings ([brms_log_M](#), [brms_log_h0](#)) are pasted straight into the predictor, so the formula fitted here is the same expression [build_hazard_data](#) assumes and that reporting later evaluates.

Usage

```
build_hazard_spec(
  m_form,
  baseline = "mem",
  dpar_formulas = list(),
```

```

observation = c("interval", "continuous", "mixed"),
k_sw = NULL,
costressors = list()
)

```

Arguments

m_form	A character string denoting a key of M_catalogue (e.g. "exp" or "exp_threshold").
baseline	A character string denoting a key of h0_catalogue : "mem" (default), "weibull", "llogis" or "gompertz".
dpar_formulas	A named list of one-sided formulas, one per parameter you want covariates on. A parameter the functions need but you omit defaults to ~ 1 (intercept only); a name not among the parameters is ignored with a warning. For example, <code>list(loga = ~ 1 + acclim + (1 trial), logb = ~ 1 + (1 trial))</code> .
observation	A character string denoting how each failure time was observed, as in build_hazard_data : "interval" (the cloglog_bern_family), "continuous" (a Poisson piecewise-exponential), or "mixed" (the hazard_mixed_family , which reads the per-row exact flag through <code>vint()</code>).
k_sw	An optional positive numeric representing the softening steepness of a threshold function's gate, in inverse temperature units (the transition band is roughly its reciprocal). This is an advanced override: leave NULL (the default) to use the value in M_catalogue (50), which suits most fits. A larger value is a sharper switch over a narrower band, but too large returns the sampler to the hard step geometry, where the critical temperature T_c mixes poorly (low effective sample size, sometimes divergences). This is a sampling-geometry effect, separate from identifiability: T_c is poorly identified whenever the observed temperatures do not bracket it, whatever k_{sw} is. Ignored, with a warning, for a function that has no threshold.
costressors	An optional list of costressor objects. Each appends its terms to $\log M(T)$ (a main effect <code>c<name></code> , and unless switched off a temperature interaction <code>dT<name></code>), so the failure rate becomes a surface $M(T, Z)$. Their coefficients join the parameters getting a sub-formula, so a <code>dpar_formulas</code> entry can put covariates on one. Defaults to none.

Details

Each parameter the chosen functions need ($\log a$, $\log b$, T_c , and so on) gets its own sub-formula, an intercept only (~ 1) by default. Pass a one-sided formula in `dpar_formulas` to put covariates on a parameter; a formula named for a parameter the chosen functions do not use is ignored with a warning.

Value

A named list with the brms formula, the family, and the observation mode.

Examples

```
# An exponential failure rate and memoryless baseline, with interval-censored
# observation (the default).
build_hazard_spec("exp")

# An exponential-threshold failure rate with a Weibull baseline hazard in elapsed
# exposure time.
build_hazard_spec("exp_threshold", baseline = "weibull")

# A different structure per parameter: acclim as a fixed covariate on the
# amplitude loga, and a trial-level random intercept on both loga and the
# temperature sensitivity logb (`acclim` and `trial` are carried through by
# build_hazard_data()).
build_hazard_spec("exp",
  dpar_formulas = list(loga = ~ 1 + acclim + (1 | trial), logb = ~ 1 + (1 | trial)))

# Mixed interval-censored and known-time failures: the response holds the per-row
# `exact` flag through vint().
build_hazard_spec("exp", observation = "mixed")

# A crossed temperature-by-oxygen design: dissolved oxygen enters log M as a main
# effect (cDO) and a temperature interaction (dTDO).
do <- costressor("DO", column = "do", ref = 8.5, scale = 1, direction = -1,
  interaction = TRUE)
build_hazard_spec("exp", costressors = list(do))

# Advanced: a gentler threshold switch than the catalogue default.
build_hazard_spec("exp_threshold", k_sw = 20)
```

check_survival

Check a fitted model against the survival observed in each group

Description

Compares, for each group of individuals, the proportion observed to survive through time with the proportion surviving in datasets simulated from the fitted model: a posterior predictive check. Where the model describes the data, the observed proportions lie within the band of the simulated ones.

Usage

```
check_survival(fit, by = NULL, prob = 0.95, n_draws = 200)
```

Arguments

fit	A brmsfit, a model fitted with <code>fit_hazard()</code> .
by	An optional character string naming a variable of the fitted model with one value per individual (a grouping such as a trial, or a covariate with few values), to split the individuals of each trajectory by.

prob	A single number between 0 and 1, the coverage of the band of simulated proportions. Defaults to 0.95.
n_draws	A single number, the number of posterior draws to simulate from, evenly spaced through the posterior. Defaults to 200.

Details

A group is the set of individuals that followed one temperature trajectory (one `path_id` of the data), or, with `by`, a subset of them that share a value of another variable, such as a trial. Individuals on different trajectories are never pooled, since at the same elapsed time they are at different temperatures.

The observed proportion is the Kaplan-Meier estimate, which allows for individuals no longer observed; a failure known only to an interval is placed at the end of the interval. For the simulated proportion, each posterior draw is used to simulate every individual from the start of its trajectory to the last time any individual on that trajectory was observed, with its own covariates and the group-level effects estimated for its groups. The band therefore includes the variation expected among samples of this size, which is wide for a group of ten. The simulation is random: call `set.seed()` first for the same result again.

Value

A tibble with a row for each group and time: `path_id`, the `by` variable when given, `n` (the number of individuals in the group), `time`, `Ta` (the temperature of the trajectory in the interval ending at that time), `observed` (the Kaplan-Meier estimate of survival), and `median`, `lower` and `upper` of the simulated proportion surviving. Each group starts with survival 1 at time 0.

See Also

`plot_check()` to draw it, `hazard_residuals()` for residuals by individual.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                          checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

chk <- check_survival(fit)
plot_check(chk, time_unit = "min")

# each trial on its own
plot_check(check_survival(fit, by = "trial"))

## End(Not run)
```

cloglog_bern_family	<i>Complementary log-log Bernoulli response, on the identity link</i>
---------------------	---

Description

The response distribution for interval-censored failures. `brms`'s own `bernoulli(link = "cloglog")` forms the failure probability first, and that probability rounds to exactly one once the linear predictor passes about 3.6, after which every survived interval returns minus infinity. This is not a rare edge case in thermal failure data: near CTmax the failure rate grows very large, so the cumulative hazard over an interval (and with it the linear predictor) routinely passes that threshold, and the fit becomes unusable. This family keeps the predictor on the identity link and forms the log density from it directly, which is numerically stable across the whole range.

Usage

```
cloglog_bern_family()
```

Value

A customfamily for use with `fit_hazard()`.

costressor	<i>Declare a co-stressor for a failure-rate hazard model</i>
------------	--

Description

Describes one non-temperature environmental variable (Z) the individual is exposed to and that co-determines the failure rate: dissolved oxygen, salinity, pH, a toxicant concentration, or, for a terrestrial ectotherm, humidity. The failure rate then becomes a surface $M(T, Z)$. The co-stressor enters $\log M(T)$ as its own term and, when asked for with `interaction = TRUE`, an interaction with temperature, so a crossed temperature-by-stressor design (a ramp run at two or more stressor levels) can estimate how the stressor changes the thermal response. Build one for each co-stressor and pass them together, as a list, to `build_hazard_data`, `build_hazard_spec`, `build_hazard_prior` and `fit_hazard`, so all four read the same declaration.

Usage

```
costressor(
  name,
  column,
  ref,
  scale,
  direction,
  interaction = FALSE,
  trajectory = NULL,
  transform = NULL
)
```

Arguments

name	A character string, a short name for the co-stressor, used to name its column (<code>z_<name></code>) and its parameters (<code>c<name></code> , <code>dT<name></code>). Letters and digits only, starting with a letter, because brms allows no dot or underscore in a parameter name. For example "D0" gives the column <code>z_D0</code> and the parameters <code>cD0</code> and <code>dTD0</code> .
column	A character string naming the column that holds the raw measurement <code>X</code> : a column of <code>ind</code> (one value per individual) for a stressor held fixed within a trial, or a column of <code>trajectory</code> for one that changes over the trial.
ref	The benign, no-additional-stress reference level of the raw measurement (air-saturation for dissolved oxygen, zero for a toxicant, the control level for salinity or pH). The failure rate reduces to the pure $M(T)$ here.
scale	A single positive number, the unit a coefficient is expressed per (per mg/L or per kPa for oxygen, per unit concentration for a toxicant).
direction	Either 1 or -1, set so that a positive <code>z</code> always means more stress: -1 for dissolved oxygen (less oxygen is more stress), 1 for a toxicant (more is more stress).
interaction	A single logical value. When FALSE (the default) the co-stressor enters through its main effect alone, changing the failure rate by the same proportion at every temperature. When TRUE it also enters through a temperature interaction <code>dT<name></code> , so it can change the temperature slope of $\log M$. For a multiplicative alternative that rescales thermal sensitivity instead, see 'Two kinds of temperature interaction' below, and for what an interaction implies outside the temperatures of the data, see 'An interaction outside the temperatures of the data'.
trajectory	An optional data frame giving the co-stressor over time, with columns <code>path_id</code> , <code>time</code> and the column named above, in the shape of the <code>path</code> argument of build_hazard_data . Supply it for a stressor that changes during a trial; it is interpolated to the same interval midpoints as temperature. Leave NULL for a stressor held fixed within a trial, which is then read from <code>ind</code> .
transform	An optional function of the standardised deviation $(X - \text{ref}) / \text{scale}$ that returns the column <code>z</code> . Use it for a stressor whose harm is not signed-linear, for example a squared deviation for a U-shaped response to salinity or pH, where a departure either way is harmful. When NULL (the default) the signed-linear deficit direction $*(X - \text{ref}) / \text{scale}$ is used.

Details

The raw measurement `X` becomes a standardised column $z = \text{direction} * (X - \text{ref}) / \text{scale}$, which is zero at the benign reference. There the failure rate reduces to the pure thermal $M(T)$: the amplitude $\log a$ and critical temperature T_c keep their unstressed meaning, and a model with the co-stressor held at its reference is the same as one without it. This follows from centring the column at a chosen reference rather than at the sample mean.

The three pieces the standardisation needs (a reference, a scale, and a direction) are all required, as none has a safe default. A domain-chosen scale (per mg/L of oxygen, per unit concentration) is preferred to a data standard deviation, which is unstable with only two or three treatment levels. As an example, for dissolved oxygen with a reference at air-saturation ($\text{ref} = 8.5 \text{ mg/L}$), a scale of 1 mg/L ($\text{scale} = 1$), and less oxygen meaning more stress ($\text{direction} = -1$), $z = -1 * (X - 8.5) / 1 = 8.5 -$

X, the oxygen deficit is: $z = 0$ at saturation and $z = 2.5$ at 6 mg/L (below saturation). The direction re-orientes every stressor onto a common axis on which a larger z always means more stress, so the main effect `c<name>` has the same sign meaning for every stressor: a positive value is a hazard that rises under stress.

This signed-linear column assumes the stressor harm runs one way. For a stressor that is harmful in either direction (salinity or pH, where a departure above or below the optimum both add stress), supply a transform function so z measures distance from the reference rather than a signed deficit: a squared standardised deviation, `transform = function(u) u^2`, makes $z = ((X - \text{ref}) / \text{scale})^2$, zero at the optimum and rising on both sides. The direction then no longer applies, since z is symmetric.

The co-stressor contributes one or two parameters, named from `name`. The main effect `c<name>` is the change in the log failure rate per scale of stressor beyond the reference, and the interaction `dT<name>` (with `interaction = TRUE`) is how that change varies with temperature. These are the names brms estimates and that `build_hazard_prior` sets priors on. `c<name>` is a continuous dose term: with three or more stressor levels it is a per-scale slope through them, but with only two it cannot be told apart from a per-treatment shift of the amplitude `loga`. A main effect on its own is in fact just that shift: it is equivalent to a covariate on the amplitude, $\text{loga} \sim 1 + z$ in `build_hazard_spec`'s `dpar_formulas`. The additive `dT<name>` term is not: no `dpar_formulas` entry reproduces it, so it is the part `costressor()` supplies. A temperature interaction of a different shape can still be had through `dpar_formulas` (the multiplicative form in 'Two kinds of temperature interaction' below); what is unique to `costressor()` is this particular additive term.

Because `c<name>` and `dT<name>` are ordinary parameters, `dpar_formulas` in `build_hazard_spec` can put covariates on either, so the stressor's effect can itself vary: `cDO = ~ 1 + acclim` lets the oxygen sensitivity depend on acclimation (a stressor-by-covariate interaction), and the same on `dT<name>` lets the temperature interaction vary. These are higher-order terms, so they need a design that varies temperature, the stressor and the covariate together enough to identify them; the formula is built either way, but a thin design shows up as poor mixing rather than a clean fit.

Value

A `costressor` object: a named list holding the declaration, for the `costressors` argument of `build_hazard_data`, `build_hazard_spec`, `build_hazard_prior` and `fit_hazard`.

Categorical stressor treatments

`costressor()` expects a stressor measured on a numeric scale, so `ref`, `scale` and `direction` mean something. Quantitative treatment levels belong here even though they are "treatments": dissolved oxygen dosed at 8.5, 5 and 2 mg/L is numeric, so pass the measured level as `column`. A genuinely unordered categorical stressor, with no scale on which a reference and a per-unit step make sense, does not. Model it through `dpar_formulas` in `build_hazard_spec` instead: a factor on the amplitude `loga` gives a per-treatment level shift, the same as a stressor main effect. To let the treatment effect vary with temperature, put the factor on the parameter that governs the temperature response (`logb` for the exponential forms, `Tc` for the threshold forms). That is a genuine temperature-by-treatment interaction, though a different parameterisation from the co-stressor's `dT<name>` term, as it scales the temperature slope rather than adding to it (see 'Two kinds of temperature interaction' below). A two-level treatment can instead be encoded 0/1 and passed to `costressor()`, which then gives the co-stressor's own main and interaction terms.

An interaction outside the temperatures of the data

The interaction is off by default because of how it behaves away from the temperatures at which individuals failed. It makes the effect of the stressor a straight line in temperature, $c + dT (T_a - T_r)$, which is informed only over those temperatures: a few degrees, in a ramping assay. Extended below them the line reaches zero at $T_r - c / dT$ and changes sign beyond, so a model in which the stressor is harmful where individuals failed may have it protective at cooler temperatures, such as those of a field record. Use the interaction to describe how the stressor's effect changes within the temperatures of the assay, and compare what the models with and without it imply at the temperatures a prediction is needed for (see the vignette *Temperature and a co-stressor*).

Two kinds of temperature interaction

The $dT<name>$ interaction is additive on the temperature slope: it adds $dT<name> * (T_a - T_r) * z$ to $\log M$, so the per-degree effect shifts by a constant amount per unit of stressor. This is the standard temperature-by-stressor interaction, it works the same on every $M(T)$ function, and it lets the slope change sign, so a stressor can in principle flatten or reverse the thermal effect. The sign of $dT<name>$ then reads as the synergy or antagonism of temperature and the stressor (on the proportional-hazards scale): positive steepens the thermal slope, negative flattens it, and a large enough negative value reverses it (an antagonistic crossover where warming becomes protective under severe harm caused by the co-stressor). The alternative is to let the stressor rescale thermal sensitivity by a proportion, which keeps it positive: set `interaction = FALSE` (so the co-stressor adds only its main effect and the $z_{<name>}$ column) and put that column on the sensitivity parameter through `dpar_formulas` instead: $\log b \sim 1 + z_{<name>}$ for the exponential forms (for the threshold forms a factor on T_c shifts the threshold location, a different effect). Choose the additive form for a conventional interaction and the freedom for the slope to change sign (e.g., an antagonistic crossover), the multiplicative form when the biology is a proportional change in sensitivity that should stay positive (it can attenuate the thermal slope but not reverse it).

Examples

```
# Dissolved oxygen held at a fixed level per trial: benign at air-saturation
# (about 8.5 mg/L), expressed per mg/L, with less oxygen meaning more stress. It
# enters through its main effect alone.
costressor("DO", column = "do", ref = 8.5, scale = 1, direction = -1)

# The same with its interaction with temperature.
costressor("DO", column = "do", ref = 8.5, scale = 1, direction = -1,
           interaction = TRUE)

# A toxicant that changes over the trial, supplied as its own trajectory.
tox_path <- data.frame(path_id = "A", time = c(0, 60), conc = c(0, 4))
costressor("Tox", column = "conc", ref = 0, scale = 1, direction = 1,
           trajectory = tox_path)

# Salinity, harmful above or below an optimum: a squared standardised deviation
# makes z measure distance from the optimum (about 15 PSU, per 5 PSU) in either
# direction. With a transform supplied, `direction` no longer applies.
costressor("Sal", column = "sal", ref = 15, scale = 5, direction = 1,
           transform = function(u) u^2)
```

derive_dctmax	<i>Dynamic critical thermal maximum under a ramp, from a fitted model</i>
---------------	---

Description

Derives, from a model fitted with `fit_hazard()`, the dynamic critical thermal maximum dCTmax of Jorgensen et al. (2021): the temperature at which a fraction p of individuals has failed while warmed from T_{start} at a constant rate. It is the critical thermal maximum a ramping trial would record under that ramp, with its posterior median and credible interval, at the covariate values in each row of `newdata`. Random effects are set to zero, so the results are those of a typical group.

Usage

```
derive_dctmax(
  fit,
  rate,
  T_start,
  newdata = NULL,
  p = 0.5,
  levels = NULL,
  prob = 0.95,
  n_draws = 1000,
  T_max = NULL
)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>rate</code>	A numeric vector of ramp rates, in degrees per unit of time of the data the model was fitted to (e.g. degrees C per minute), all positive.
<code>T_start</code>	A numeric vector of starting temperatures, one per rate or a single one for all.
<code>newdata</code>	A data frame with one row per set of covariate values to report at, holding every covariate in the parameters' formulas. <code>NULL</code> (the default) suits a model with no covariates.
<code>p</code>	A numeric vector of fractions of individuals failing, each strictly between 0 and 1. Defaults to 0.5.
<code>levels</code>	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. <code>c(D0 = 6)</code> . Required when the model has co-stressors.
<code>prob</code>	A single number between 0 and 1, the coverage of the credible intervals. Defaults to 0.95.
<code>n_draws</code>	A single number, the number of posterior draws to use, evenly spaced through the posterior. Defaults to 1000.
<code>T_max</code>	A single number, the temperature up to which the ramp is followed. <code>NULL</code> (the default) takes 15 degrees above the warmest temperature the model was fitted to.

Details

Unlike the static critical thermal maximum from `derive_tdt_ttl()`, dCTmax depends on the ramp. Individuals accrue hazard throughout the ramp, so a slower one gives them more time to fail before a given temperature is reached, and its dCTmax is lower. Giving several rates shows how much. The ramp need not be one that was run: the model was fitted to failure times, from a static or a ramping assay, and is applied here to any ramp.

Under the ramp $T(t) = T_{\text{start}} + \text{rate } t$, the cumulative hazard is the integral of $M(T(t)) h_0(t)$ from the start of the ramp, and dCTmax solves, for each posterior draw, cumulative hazard = $-\log(1 - p)$. With a baseline other than the memoryless one, the hazard also depends on the time elapsed since the ramp began.

The ramp is followed up to T_{max} . A posterior draw whose fraction failed has not reached p by then counts as infinite, as in `derive_tdt_ttl()`, and is counted in `n_unreached`. A message is given when a dCTmax lies above the temperatures the model was fitted to, where the data do not constrain the failure-rate function.

Value

A tibble with a row for each row of `newdata`, `ramp` and `p`: the columns of `newdata`, `rate`, `T_start`, `p`, and the posterior median, lower and upper of dCTmax, with `prob`, `n_draws` and `n_unreached`.

References

Jorgensen, L. B., Malte, H., Orsted, M., Klahn, N. A. and Overgaard, J. (2021). A unifying model to estimate thermal tolerance limits in ectotherms across static, dynamic and fluctuating exposures to thermal stress. *Scientific Reports* 11, 12840.

See Also

`derive_tdt_ttl()` for the static critical thermal maximum, and `average_dctmax()` to average dCTmax over several models.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

# the CTmax expected under three ramps from 14 degrees
derive_dctmax(fit, rate = c(0.1, 0.2, 0.3), T_start = 14,
              newdata = data.frame(mass = 15))

## End(Not run)
```

derive_tdt_ttl

*Thermal tolerance derived from a fitted failure-rate model***Description**

Derives, from a model fitted with `fit_hazard()`, the failure-rate parameters on their natural scale, the static critical thermal maximum `sCTmax`, the thermal death time curves and the thermal tolerance landscape, each with its posterior median and equal-tailed credible interval (the interval between the $(1 - \text{prob}) / 2$ and $1 - (1 - \text{prob}) / 2$ quantiles of the draws), at the covariate values in each row of `newdata`. The model's settings (failure-rate and baseline functions, reference temperature, softening steepness, co-stressors) are read from the fit. Random effects are set to zero, so the results are those of a typical group.

Usage

```
derive_tdt_ttl(
  fit,
  newdata = NULL,
  p = c(0.1, 0.5, 0.9),
  t_ref,
  Ta = NULL,
  tau = NULL,
  levels = NULL,
  prob = 0.95,
  n_draws = 1000
)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>newdata</code>	A data frame of covariate values, one row per setting at which to derive. It must hold every covariate in the model's parameter formulas. May be <code>NULL</code> for a model without covariates.
<code>p</code>	A numeric vector of the fractions of individuals failing at which <code>sCTmax</code> and the thermal death time curves are derived, each strictly between 0 and 1.
<code>t_ref</code>	A single positive number, the reference exposure time for <code>sCTmax</code> , in the time unit of the data the model was fitted to (for example 60 for one hour when times are in minutes). It has no default, because the fit does not record that unit.
<code>Ta</code>	An optional numeric vector of the constant temperatures for the thermal death time curves and the tolerance landscape. <code>NULL</code> (the default) takes 50 evenly spaced across the temperatures the data cover.
<code>tau</code>	An optional numeric vector of the elapsed exposure times for the tolerance landscape, in the time unit of the data, none negative. <code>NULL</code> (the default) takes 50 spaced evenly on a log scale across the times the data cover. A time of zero is allowed, and gives a survival of 1, so that a survival curve can start there.

levels	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. <code>c(D0 = 6)</code> . Required when the model has co-stressors.
prob	A single number strictly between 0 and 1, the coverage of the credible intervals.
n_draws	An optional single number, the number of posterior draws to use, evenly spaced through the posterior. Defaults to 1000, which keeps the derivation fast while the limits of a 95% interval still rest on about 25 draws in each tail; NULL uses them all.

Details

sCTmax is the constant temperature at which a fraction p of individuals fail within the reference exposure time t_{ref} , the static critical thermal maximum of Jorgensen et al. (2021), who report it for one hour. Under a constant temperature, survival is $\exp(-M(T) H_0(t_{\text{ref}}))$, so for each posterior draw sCTmax solves $M(T) = -\log(1 - p) / H_0(t_{\text{ref}})$. It is not the critical thermal maximum a ramping experiment records (their dynamic CTmax).

The parameters reported are a , and where the failure-rate function has them b , c and T_c ; the baseline's shape (Weibull, log-logistic), its scale (log-logistic; the Weibull's is fixed at the reference time, see [h0_catalogue](#)) or its rate constant gamma (Gompertz); and each co-stressor's coefficients. With a co-stressor that interacts with temperature, $b_{\text{at_levels}}$ is the thermal sensitivity $b + dT z$ at the chosen levels. The thermal sensitivity z , the temperature change that shortens the time to failure tenfold, is reported only where the thermal death time curve is a straight line in log time against temperature, so that one number describes it: the exponential failure rate with the memoryless or Weibull baseline, where $z = k \log(10) / b$ ($k = 1$ for the memoryless baseline, the Weibull shape otherwise). Elsewhere the curve bends and no single z stands for it.

The thermal death time curve for a fraction p gives, at each constant temperature in T_a , the time until that fraction has failed: $t_p = H_0^{-1}(-\log(1 - p) / M(T))$. The thermal tolerance landscape gives survival $\exp(-M(T) H_0(\tau))$ over the grid of temperatures T_a and elapsed exposure times τ . "Thermal death time" is the literature's name for the curve, whatever the failure endpoint.

Value

A list of four tibbles, each starting with the columns of `newdata` and ending with the posterior summary median, lower, upper, prob, `n_draws` and `n_unreached`:

- `params`: one row per parameter, named in `quantity`;
- `sctmax`: one row per value of p , with p and t_{ref} ;
- `tdt_curves`: one row per value of p and temperature, with p and T_a ; the summary is of the time until that fraction has failed;
- `ttl_surface`: one row per temperature and elapsed time, with T_a and τ ; the summary is of survival.

Draws that never reach the quantity

A posterior draw for which a quantity is never reached (an sCTmax the rate cannot attain within 1000 degrees of the reference temperature, or at all; a time to failure that never comes, under a Gompertz baseline whose cumulative hazard levels off) counts as infinite rather than being dropped, so the summaries describe the whole posterior. `n_unreached` counts these draws, and a message

lists the quantities that have any. With coverage prob, let $u = (1 - \text{prob}) / 2$ (0.025 for a 95% interval):

- fewer than a share u of the draws unreached: both bounds are finite; report the summary as it stands.
- between u and half: the outer bound is infinite; report it as unbounded (for example "above 40 degC", "not reached"), and consider whether the question lies beyond the temperatures or exposure times the data cover.
- half or more: the median is infinite; the quantity is not estimable there. Ask at a lower p or a shorter t_{ref} .

References

Jorgensen, L. B., Malte, H., Orsted, M., Klahn, N. A. and Overgaard, J. (2021). A unifying model to estimate thermal tolerance limits in ectotherms across static, dynamic and fluctuating exposures to thermal stress. *Scientific Reports* 11, 12840.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                        checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

tol <- derive_tdt_ttl(fit, newdata = data.frame(mass = c(8, 12)), p = 0.5,
                    t_ref = 60)

tol$sctmax

## End(Not run)
```

fit_hazard

Fit a failure-rate hazard model with brms

Description

Assembles the formula and family with [build_hazard_spec](#), the default prior with [build_hazard_prior](#) (unless you supply your own), selects the Stan code the custom families need, and hands it all to [brms::brm](#). Anything in ... is forwarded to brm (chains, iterations, cores, backend, initial values, and so on).

Usage

```
fit_hazard(
  long,
  m_form,
  baseline = "mem",
  dpar_formulas = list(),
  observation = c("interval", "continuous", "mixed"),
  prior = NULL,
  intercept_priors = list(),
  slope_prior = NULL,
  sd_prior = NULL,
  costressors = list(),
  k_sw = NULL,
  ...
)
```

Arguments

long	A data frame, the row-per-interval table from build_hazard_data .
m_form, baseline, dpar_formulas, observation	As in build_hazard_spec .
prior	An optional brmsprior for full manual control. When NULL (the default) a weakly-informative prior is built with build_hazard_prior . A supplied prior is used as given, in place of that default rather than merged with it, so any coefficient it leaves out falls back to brms's own default, not the weakly-informative one. To adjust only some priors and keep the rest, use <code>intercept_priors / slope_prior / sd_prior</code> instead of this.
intercept_priors, slope_prior, sd_prior	As in build_hazard_prior , forwarded when the default prior is built (ignored if you supply prior). Each changes only the priors it names and leaves the rest at their defaults, so a user who only calls <code>fit_hazard</code> can retune without building a prior by hand.
costressors	An optional list of costressor objects, as in build_hazard_spec , forwarded to both the formula and the default prior. The same list must have been passed to build_hazard_data , so long carries the <code>z_<name></code> columns the formula reads. Defaults to none.
k_sw	An optional softening steepness for a threshold function's gate, as in build_hazard_spec . An advanced override: leave NULL (the default) to use the value in M_catalogue . A fit made with a custom value must be reported with the same one.
...	Further arguments forwarded to brms::brm . Any argument it accepts (e.g. <code>chains</code> , <code>iter</code> , <code>cores</code> , <code>backend</code> , <code>init</code>), except <code>formula</code> , <code>data</code> , <code>family</code> , <code>prior</code> and <code>stanvars</code> , which <code>fit_hazard</code> sets itself. See brms::brm for the full set.

Details

A missing value in any variable the model uses is an error. brms would drop the rows concerned without saying so, leaving some individuals with only part of their exposure, so remove those individuals or fill in the values first.

Before fitting, the temperature is checked against the reference T_r for the failure rates that take a logarithm or fractional power of $T_a - T_r$ (zeroexp, power, zeropower): these are undefined where the temperature reaches T_r (zeroexp, zeropower) or drops below it (power), so a run that would hit that is stopped with a clear error rather than a cryptic sampler failure.

Value

A `brmsfit`, carrying in `$rtft` the settings it was fitted with: the failure-rate and baseline functions, the observation mode, the reference temperature T_r and reference time `tau_ref`, the softening steepness, the co-stressors, the ranges of temperature and elapsed exposure time the data cover, and, for each row of `long`, the individual and the temperature trajectory it belongs to and the elapsed time at which its interval ends. The reporting functions and the checks read them from there, so they are never re-entered by hand.

Examples

```
## Not run:
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1,
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", checks = checks)
fit <- fit_hazard(long, "exp", chains = 4, iter = 2000, cores = 4)

## End(Not run)
```

h0_catalogue	<i>Baseline hazard functions $h_0(\tau)$</i>
--------------	---

Description

This catalogue contains the four baseline hazard functions $h_0(\tau)$, the factor in elapsed exposure time that multiplies the failure rate $M(T)$. `mem` is the memoryless case, in which the hazard does not depend on how long the individual has already been exposed; `weibull`, `llogis` and `gompertz` allow it to rise or fall with elapsed exposure.

Usage

```
h0_catalogue
```

Format

A named list, one entry per baseline function.

Details

mem is the baseline canonical to Kingsolver and Umbanhowar (2018), and is the default throughout. Their framework has no baseline-hazard-in-time component at all: the only hazard is $M(T)$ evaluated along the temperature trajectory $T(t)$. The other three are extensions of their framework.

gompertz, $h_0(\tau) = \exp(\gamma * \tau)$, is the only one of the four baselines whose hazard increases exponentially with elapsed exposure time, which is also the shape an exponential $M(T)$ exhibits as temperature increases through a ramp. Because the two share that shape, a single ramp rate cannot separate them, as it only sees the combined acceleration, not how much comes from the increasing temperature and how much from time under exposure. Two or more ramp rates can separate them, because the temperature part steepens with the ramp rate while the elapsed time part does not. The same confound occurs between weibull, whose hazard is a power of elapsed time, and a power $M(T)$ (zeropower or zeropower_threshold), since a power of temperature along a ramp is also a power of elapsed time. So the two are confounded under a single ramp and, as for gompertz, are separated by two or more ramp rates. llogis, whose hazard rises then falls, matches none of the failure-rate functions of temperature, which all rise monotonically, so it is not confounded with any of them.

A Gompertz baseline hazard is a different object from the Gompertz distribution of failure times that Kingsolver and Umbanhowar's Eqn 6 derives. The distribution is an outcome of the model: an exponential $M(T)$ traversed along a linear ramp, under a memoryless baseline, produces Gompertz-distributed failure times. The baseline hazard presented here is an assumption about how risk depends on elapsed exposure time, independent of temperature.

Each entry gives the integer the Stan code dispatches on (stan_id) and the parameters of each baseline function (dpars), as in [M_catalogue](#). mem has no parameters. weibull has a shape parameter and llogis shape and scale parameters, on the log scale (logshape, logscale). gompertz has the rate constant gamma (gam) parameter in reciprocal time, left on the natural scale and unconstrained in sign (a positive value is a hazard rising with elapsed exposure time and a negative one a hazard falling). The amplitude of the Gompertz hazard is not included, because it is the same quantity as loga in $M(T)$, and mem omits a parameter for the same reason.

The scale of the weibull baseline is not estimated. It is fixed at the reference time tau_ref set in [build_hazard_data](#), so that $h_0(\tau) = (k / \tau_{\text{ref}}) (\tau / \tau_{\text{ref}})^{(k - 1)}$ and the cumulative baseline hazard at tau_ref is 1. A Weibull scale s and the amplitude a of $M(T)$ enter the hazard only through a / s^k , so the data cannot tell them apart and only one of them can be estimated. With the scale fixed, a is the cumulative hazard reached over the reference time at the temperature where $M(T) = a$ (the reference temperature T_r , for the functions anchored there). The scale of llogis also enters where its hazard turns from rising to falling, so it is estimated; the data inform it only weakly when few failures come after that turn, and its posterior then leans on its prior.

References

Kingsolver, J. G. and Umbanhowar, J. (2018). The analysis and interpretation of critical temperatures. *Journal of Experimental Biology* 221, jeb167858. doi:10.1242/jeb.167858

H0_of

*The cumulative baseline hazard of a given baseline function***Description**

Returns the function $H_0(\tau)$ for the chosen baseline. Under the memoryless baseline it is elapsed exposure itself; the Weibull, log-logistic and Gompertz baselines let the hazard rise or fall with how long the individual has already been exposed. Shape and scale arrive as per-draw vectors, and elapsed time may be a vector or a matrix whose rows are draws.

Usage

```
H0_of(baseline)
```

Arguments

baseline A character string, a key of [h0_catalogue](#): "mem", "weibull", "llogis" or "gompertz".

Details

For the Gompertz baseline the first argument is the rate constant gamma and the second is ignored. A negative gamma is a valid value and means a hazard that falls the longer the individual has been exposed. The cumulative hazard then levels off: it rises towards $1/\text{gamma}$ but never reaches it, instead of growing without limit. Survival therefore never drops to zero, and the model predicts that some individuals never fail (a cure fraction, in survival-analysis terms). `H0inv_of()` gives the time by which a chosen fraction of individuals has failed; when that fraction is never reached it returns NaN for the time, rather than a large number standing in for "never".

Value

A function of (tau, shape, scale) giving the cumulative baseline hazard. What shape and scale mean depends on the baseline:

- "mem": both are ignored.
- "weibull" and "llogis": shape is k and scale is s.
- "gompertz": shape is the rate constant gamma and scale is ignored.

Examples

```
# The memoryless baseline is elapsed exposure itself.
H0_of("mem")(tau = c(10, 30, 60), shape = 1, scale = 1)

# A Weibull baseline with shape 2 and scale 60:  $H_0(\tau) = (\tau / 60)^2$ .
H0_of("weibull")(tau = 30, shape = 2, scale = 60)
```

H0inv_of

*The inverse of the cumulative baseline hazard***Description**

Returns the function that gives the elapsed time at which the cumulative baseline hazard reaches a given value, which is what turns a failure probability into a thermal death time.

Usage

```
H0inv_of(baseline)
```

Arguments

baseline A character string, a key of [h0_catalogue](#): "mem", "weibull", "llogis" or "gompertz".

Details

Under the Gompertz baseline with a negative rate constant the cumulative hazard levels off below $1/\text{gamma}$, so a target above that level is never reached. The time returned there is NaN, rather than a large number standing in for "never". A target exactly at the level is reached only after infinite time, and returns Inf.

Value

A function of (x, shape, scale) giving the elapsed time at which the cumulative baseline hazard equals x. shape and scale mean the same as in [H0_of\(\)](#):

- "mem": both are ignored.
- "weibull" and "llogis": shape is k and scale is s.
- "gompertz": shape is the rate constant gamma and scale is ignored.

Examples

```
# Invert a Weibull baseline (shape 2, scale 60): the time at which H0 reaches 0.25.
H0inv_of("weibull")(x = 0.25, shape = 2, scale = 60)    # 30
```

hazard_mixed_family	<i>Response distribution for a table mixing interval-censored and known-time failures</i>
---------------------	---

Description

Some failures are known only to have fallen within an interval, others at a known time. This family fits both kinds in one model. A row where the individual survived its interval contributes the same amount either way, so only the row recording the failure differs: an interval-censored failure contributes the probability of failing somewhere inside the interval, whereas a failure seen at a known time contributes the density at that recorded time. A per-row flag, passed to brms through `vint()`, tells the model which of the two to use, so a single fit can contain individuals whose failures were recorded as either kind.

Usage

```
hazard_mixed_family()
```

Value

A customfamily for use with `fit_hazard()`.

hazard_residuals	<i>Residuals of a fitted model, one per individual</i>
------------------	--

Description

Computes a randomised residual for each individual, in the manner of the simulation-based residuals of the DHARMa package: the position of what was observed within what the fitted model predicts for that individual. When the model describes the data, the residuals are uniform between 0 and 1.

Usage

```
hazard_residuals(fit, n_draws = 400)
```

Arguments

<code>fit</code>	A brmsfit, a model fitted with <code>fit_hazard()</code> .
<code>n_draws</code>	A single number, the number of posterior draws to average the predicted survival over, evenly spaced through the posterior. Defaults to 400.

Details

The residual is the fraction of individuals that the model predicts to have failed by the time this one did, $1 - S(t)$, with the individual's own covariates and the group-level effects estimated for its groups, averaged over the posterior draws. A residual near 0 is an individual that failed earlier than the model predicts, and one near 1 an individual that lasted longer. What is observed is not always a single time, so the residual is drawn at random within what was observed:

- a failure at a known time has the residual $1 - S(t)$;
- a failure known only to an interval has a residual drawn uniformly between that fraction at the start of the interval and at its end;
- a survivor, known only to have outlasted its last observation, has a residual drawn uniformly between that fraction at its last observation and 1.

The draws are random: call `set.seed()` first for the same residuals again. One residual per individual is used, and not one per interval, because the intervals of an individual are slices of one exposure and are not separate observations.

The residuals are to be examined against what was set before the outcome was known: a covariate, a treatment, a trial. They should not be examined against the time of failure, or the temperature at failure in a ramp. An individual that failed late has, by that fact alone, a high predicted fraction failed at its time of failure, so the residuals rise with the time of failure under any model, right or wrong.

Value

A tibble with one row per individual: `id`; `path_id`; the variables of the model that have one value per individual (covariates and groupings); `time`, the time at which it failed or was last observed; `Ta`, the temperature at that time; `event`, 1 for a failure and 0 for a survivor; and `residual`.

References

Hartig, F. (2024). DHARMA: Residual diagnostics for hierarchical (multi-level / mixed) regression models. R package version 0.4.7. [doi:10.32614/CRAN.package.DHARMA](https://doi.org/10.32614/CRAN.package.DHARMA)

See Also

`plot_residuals()` to draw them, `check_survival()` for a check by group.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                        checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
```

```

set.seed(1)
res <- hazard_residuals(fit)
plot_residuals(res)           # against the uniform distribution
plot_residuals(res, against = "trial") # by trial

## End(Not run)

```

hazard_weights	<i>Weights for averaging failure-rate models</i>
----------------	--

Description

Weights several models fitted to the same data by how well each predicts data it was not fitted to, from their cross-validation results. The models can be any set: any failure-rate functions with any baselines. Which models are averaged is chosen by which results are given.

Usage

```
hazard_weights(cv, method = c("stacking", "pseudobma"), min_weight = 1)
```

Arguments

cv	A named list of cross-validation results, one per model, from loo_hazard() or kfold_hazard() . All must have the same unit and the same units left out. The names identify the models.
method	A character string, "stacking" (the default) or "pseudobma".
min_weight	A single number above 0 and at most 1, the cumulative weight the models kept must reach. The default, 1, keeps every model.

Details

With method = "stacking" the weights are those of the combination of models that predicts best. It is the method Yao et al. (2018) recommend, and the default here. Stacking can give a model a weight of zero. This happens when two models make nearly the same predictions: including both adds nothing to the combination, so one takes the weight and the other gets none, even though it would predict almost as well on its own. With method = "pseudobma" each model is weighted by its own predictive density, allowing for its uncertainty over the units by the Bayesian bootstrap; these weights are smoother, and steadier when the units are few. The Bayesian bootstrap is random: call [set.seed\(\)](#) first for the same weights again.

With min_weight below 1, only the highest-weighted models are kept, as many as it takes for their weights to add up to at least min_weight, and the weights of those kept are then computed again among themselves. Recomputing matters for stacking, whose weights are found jointly for a set of models and are not simply rescaled when a model leaves the set.

The weights rest on one value per unit left out, so they are less certain the fewer units there are, as when the unit is one of a handful of trials. A message says how many units there were.

Results from `loo_hazard()` and `kfold_hazard()` can be mixed, as when the leave-one-out approximation is unreliable for one model and that model alone is cross-validated by refitting. The two agree in what they estimate when each fold leaves out a single unit. With fewer folds than units, each refit sees less of the data than a leave-one-out fit does, so its predictions score slightly lower and its model is at a small disadvantage. A message names the models this applies to.

Value

A tibble with one row per model, highest weight first: `model`; `elpd`, its summed log predictive density; `weight_all`, its weight among all the models, and `cumulative`, the running total; `kept`, whether it is kept; and `weight`, its weight among the models kept, which is the weight to average with (0 for a model not kept). The unit, the number of units and the method are attached as attributes.

References

Yao, Y., Vehtari, A., Simpson, D. and Gelman, A. (2018). Using stacking to average Bayesian predictive distributions. *Bayesian Analysis* 13, 917-1007.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit_mem <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
fit_weibull <- fit_hazard(long, "exp", baseline = "weibull",
                         dpar_formulas = list(loga = ~ 1 + mass))
fits <- list(memoryless = fit_mem, weibull = fit_weibull)

cv <- lapply(fits, loo_hazard)
hazard_weights(cv)
hazard_weights(cv, min_weight = 0.95)

## End(Not run)
```

kfold_hazard

Cross-validation by refitting, leaving out whole individuals or groups

Description

Estimates how well a model fitted with `fit_hazard()` predicts data it was not fitted to, exactly, by refitting it with each unit left out in turn and predicting the unit from the refit. It answers the same question as `loo_hazard()`, with the same choice of unit, but without its approximation: use

it where `loo_hazard()` warns that PSIS cannot be trusted, which is typical when whole trials with a strong trial effect are left out.

Usage

```
kfold_hazard(fit, unit = NULL, K = NULL, ...)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>unit</code>	An optional character string, the unit to leave out, as in <code>loo_hazard()</code> .
<code>K</code>	An optional single whole number, the number of folds, from 2 to the number of units. <code>NULL</code> (the default) gives one fold per unit, leaving out each unit on its own.
<code>...</code>	Further arguments for the refits, passed to <code>brms::brm()</code> through <code>stats::update()</code> , such as <code>chains</code> , <code>iter</code> or <code>cores</code> .

Details

The units are split into `K` folds, and the model is refitted once per fold without that fold's units. A unit's rows are summed within each draw of the refit, so its value is its joint log predictive density, averaged over the draws. Where the unit's level of a group-level term was left out entirely (a whole trial), its effect is drawn from the fitted population distribution, so the trial is predicted as a new trial would be.

The refits take time: one model fit per fold. With `K` smaller than the number of units, the units are assigned to folds at random; call `set.seed()` first to get the same folds again.

Value

A `kfold` object in the layout of `loo::kfold()`, with one pointwise value per unit, which `loo::loo_compare()` accepts, and extra elements `unit` (its name), `K`, and `folds` (the fold of each unit).

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)

fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + (1 | trial)))
cv <- kfold_hazard(fit, cores = 4)      # one refit per trial

## End(Not run)
```

loo_hazard	<i>Leave-one-out cross-validation that leaves out whole individuals or groups</i>
------------	---

Description

Estimates how well a model fitted with `fit_hazard()` predicts data it was not fitted to, by Pareto-smoothed importance sampling (PSIS-LOO), leaving out one unit at a time: an individual, or a group of individuals.

Usage

```
loo_hazard(fit, unit = NULL)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>unit</code>	An optional character string, the unit to leave out: "id" for the individual, or one of the groupings of the model's group-level terms, such as "trial", or "trial:tank" for the tanks of (1 trial/tank). NULL (the default) chooses as described above.

Details

The model is fitted one row per individual and interval at risk, so `brms::loo()` would leave out one interval, and ask how well the model predicts the rest of an individual's exposure from its beginning, which is nearly always easy. Here the pointwise log-likelihood is summed within each unit, which given the parameters is that unit's log-likelihood exactly, and the unit is left out whole.

Where individuals share a group-level effect in the model, such as a random intercept per trial, leaving out one individual still leaves its trial's effect, estimated from the other individuals of the trial, in the fit, and the prediction looks better than it would for a new trial. So by default the unit is the coarsest grouping variable whose levels individuals share, with a message saying so, and otherwise the individual. Nested groupings, such as tanks within trials, are handled by this: leaving out a trial takes its tanks with it. Asking for a finer unit gives a warning that the coarser grouping's effect stays in the fit, which is appropriate when the data to be predicted come from the same levels of it, and optimistic when they come from new ones.

Crossed groupings, such as families spread across trials, have no default: no single unit removes the effects of both, and leaving out trials answers how well the model predicts new trials of known families, while leaving out families answers how well it predicts new families. The unit must then be given.

Leaving out a large unit can change the posterior too much for PSIS to stand in for refitting. Where the Pareto k diagnostic of any unit is above the threshold for the number of draws, a warning says so, and `kfold_hazard()` gives an exact estimate by refitting.

Models compared with `loo::loo_compare()` must be cross-validated with the same unit.

Value

A loo object (see `loo::loo()`) with one pointwise value per unit, and an extra element unit naming the unit.

References

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

Vehtari, A., Simpson, D., Gelman, A., Yao, Y. and Gabry, J. (2024). Pareto smoothed importance sampling. *Journal of Machine Learning Research* 25(72), 1-58.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                        checks = checks)

fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + (1 | trial)))
loo_hazard(fit) # leaves out one trial at a time

## End(Not run)
```

M_catalogue	Failure rate functions $M(T)$
-------------	-------------------------------

Description

This catalogue contains every function the failure rate may take: the nine failure rate functions $M(T)$ of Kingsolver and Umbanhowar (2018, Table 1), each catalogue key named for its model there. Without a threshold: const (a constant rate), exp (exponential), zeroexp (exponential minus its value at the reference temperature T_r , so the rate is zero there), power (a power of temperature above a baseline rate) and zeropower (the same power with no baseline, zero at T_r). The threshold variants const_threshold, exp_threshold, zeroexp_threshold and zeropower_threshold hold the rate at zero below a fitted critical temperature T_c ; above T_c it is constant for const_threshold and accelerates for the other three. The exponential functions give a straight thermal death time curve, whereas the power functions give a curved one.

Usage

M_catalogue

Format

A named list, one entry per failure rate function $M(T)$.

Details

T_r (T_0 in Kingsolver and Umbanhowar's notation) is a single reference temperature set for the whole model, not the per-trial starting temperature. The exponential is anchored at T_r and the zero forms vanish there, but because the failure rate must depend only on the current temperature, T_r cannot be each trial's own starting temperature. Trials that begin at different temperatures (acclimation groups, field conditions) enter through their temperature trajectories, not through T_r .

To keep fitting, simulation and reporting from diverging, all three evaluate the same expression strings, which are held in `brms_log_M`.

Each entry gives the integer the Stan code dispatches on (`stan_id`), the parameters the function needs (`dpars`), and the steepness of the threshold switch (`k_sw`). The rate parameters a , b and c are represented on the log scale (`loga`, `logb`, `logc`), so the linear predictor is unconstrained and their strictly positive values are recovered by exponentiation; the critical temperature T_c is left on the natural scale, since it is not sign-constrained. `k_sw` is roughly the inverse of the width of the transition in temperature, so a larger value is a sharper switch over a narrower band. The switch is smoothed rather than hard because Stan fits the model with Hamiltonian Monte Carlo, which explores the posterior by following the gradient of the log-density with respect to each parameter; the gradient of a hard step with respect to the critical temperature is zero on either side of it and undefined at the step itself, so the sampler gets no slope to move that temperature along, whereas a smooth switch supplies one.

The units of $M(T)$

The hazard is $h_0(\tau) M(T)$, and which of the two factors carries the unit of reciprocal time depends on the baseline (see [h0_catalogue](#)). With the memoryless and Gompertz baselines $h_0(\tau)$ has no unit, so $M(T)$ is a rate per unit time, and a is the failure rate at the temperature where $M(T) = a$. With the Weibull and log-logistic baselines $h_0(\tau)$ is itself per unit time, so $M(T)$ is a multiplier with no unit; with the Weibull, a is the cumulative hazard reached over the reference time at that temperature. The value of a therefore cannot be compared between models with different baselines, nor between datasets timed in different units when the baseline is memoryless or Gompertz.

Whatever the baseline, survival at a constant temperature is $\exp(-M(T) H_0(\tau))$, with H_0 the cumulative baseline hazard. `derive_tdt_ttl()` derives the critical thermal maximum, the thermal death time curves and the thermal tolerance landscape from that expression, and those can be compared between models.

References

Kingsolver, J. G. and Umbanhowar, J. (2018). The analysis and interpretation of critical temperatures. *Journal of Experimental Biology* 221, jeb167858. doi:10.1242/jeb.167858

M_draws

*The fitted failure rate over temperature, for each posterior draw***Description**

Evaluates the failure rate $M(T)$ of a model fitted with `fit_hazard()` at each temperature in `Ta`, for every posterior draw, at the covariate values in each row of `newdata`. The rate is computed from the same expression the model was fitted with, and the failure-rate function, reference temperature, softening steepness and co-stressors are read from the fit, so none of them is entered by hand. Random effects are set to zero, so the rate is that of a typical group.

Usage

```
M_draws(fit, Ta, newdata = NULL, levels = NULL, n_draws = NULL)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>Ta</code>	A numeric vector of temperatures at which to evaluate the rate.
<code>newdata</code>	A data frame of covariate values, one row per setting at which to evaluate the rate. It must hold every covariate in the model's parameter formulas; other columns the model needs are filled in. May be <code>NULL</code> for a model without covariates.
<code>levels</code>	A named numeric vector giving the level of each co-stressor, on its measurement scale, e.g. <code>c(DO = 6)</code> for 6 mg/L of dissolved oxygen. Required when the model has co-stressors.
<code>n_draws</code>	An optional single number, the number of posterior draws to keep, evenly spaced through the posterior. <code>NULL</code> (the default) keeps them all.

Details

The draws are returned as they are, not summarised, for plotting the rate with its uncertainty or for further calculation.

Value

A tibble with one row per draw, temperature and row of `newdata`: the columns of `newdata`, `.draw` (the draw's number in the fit, so draws from different calls can be matched), `Ta` and the failure rate `M`.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
```

```

ind    <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                     trial = rep(c("a", "b"), each = 10),
                     t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long   <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                           checks = checks)
fit    <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

rate <- M_draws(fit, Ta = seq(25, 40, by = 0.5),
               newdata = data.frame(mass = c(8, 12)))

## End(Not run)

```

plot_check

Plot the check of a fitted model against observed survival

Description

Draws the output of `check_survival()`: for each group, the proportion of individuals surviving in datasets simulated from the fitted model, as its median and band, with the proportion observed to survive as a stepped line over it. Where the model describes the data, the stepped line stays within the band. The result is a ggplot object, which can be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```
plot_check(x, time_unit = NULL)
```

Arguments

<code>x</code>	A tibble, the output of <code>check_survival()</code> .
<code>time_unit</code>	An optional character string giving the time unit for the axis label, e.g. "min". The fit does not record the unit of its times, so none is shown unless given.

Value

A ggplot object, with a panel per group.

Examples

```

## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path  <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind    <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                     trial = rep(c("a", "b"), each = 10),
                     t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long   <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                           checks = checks)

```

```
fit    <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

plot_check(check_survival(fit), time_unit = "min")

## End(Not run)
```

plot_residuals

Plot the residuals of a fitted model

Description

Draws the output of `hazard_residuals()`. With `against = NULL`, the residuals are set against the values expected of a uniform distribution, which is what they follow when the model describes the data: the points then lie along the diagonal. With `against` naming a column, the residuals are drawn against it, to look for a pattern the model has missed: points for a numeric variable, such as body mass, and boxes for a grouping, such as the trial or the treatment. A horizontal line marks 0.5, the middle of the residuals under the model; residuals below it are individuals that failed earlier than the model predicts. The time of failure is not a variable to draw them against (see `hazard_residuals()`). The result is a ggplot object, which can be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```
plot_residuals(x, against = NULL)
```

Arguments

<code>x</code>	A tibble, the output of <code>hazard_residuals()</code> .
<code>against</code>	An optional character string naming a column of <code>x</code> to draw the residuals against. <code>NULL</code> (the default) compares them with the uniform distribution.

Value

A ggplot object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path  <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind    <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                     trial = rep(c("a", "b"), each = 10),
                     t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long   <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                           checks = checks)
fit     <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
```

```

res <- hazard_residuals(fit)
plot_residuals(res)
plot_residuals(res, against = "mass")
plot_residuals(res, against = "trial")

## End(Not run)

```

plot_risk

Plot each period's share of the risk along a record

Description

Draws the risk table of `predict_survival()` or `average_survival()`: the share of the hazard accrued over the record that falls in each period, as a bar at the posterior median with its credible interval. It shows where along the record the risk is concentrated. The result is a ggplot object, which can be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```
plot_risk(x, facet = TRUE, interval = TRUE)
```

Arguments

x	A list, the output of <code>predict_survival()</code> or <code>average_survival()</code> called with <code>by</code> .
facet	A single logical value or the character string "all". TRUE (the default) gives a panel per setting of newdata, cut down to quartile values when there are many; "all" a panel for every setting; FALSE no panels, with the settings side by side in each period.
interval	A single logical value, whether to draw the credible interval. Defaults to TRUE.

Details

The shares are of whatever risk there is, so they say where it falls and not how large it is; read them with the mortality table. A period has no bar where no posterior draw has any hazard along the record.

By default each setting of newdata has its own panel, following the same rule as in `plot_tdt()`. With `facet = FALSE` the settings are drawn side by side within each period, each in its own colour, from teal to deep purple in the order of newdata; this needs eight settings or fewer.

Value

A ggplot object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
# a two-day record read every hour, with a daily cycle
hours <- 0:48
record <- data.frame(time = hours * 60, day = pmin(hours %% 24, 1),
                    temp = 24 + 3 * sin(2 * pi * (hours - 9) / 24))

pred <- predict_survival(fit, record, newdata = data.frame(mass = c(8, 12)),
                        by = "day")

plot_risk(pred)
plot_risk(pred, facet = FALSE)

## End(Not run)
```

plot_survival

Plot survival along a record of temperature

Description

Draws the survival predicted by [predict_survival\(\)](#) or [average_survival\(\)](#): the probability of surviving to each time of the record, as the posterior median with its credible interval. The result is a ggplot object, to be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```
plot_survival(
  x,
  temperature = FALSE,
  facet = TRUE,
  colour_by = NULL,
  interval = is.null(colour_by),
  time_unit = NULL
)
```

Arguments

x A list, the output of [predict_survival\(\)](#) or [average_survival\(\)](#).

temperature	A single logical value, whether to draw the record of temperature, and of any co-stressor that changes along it, in panels above the survival curve. Defaults to FALSE.
facet	A single logical value or the character string "all". TRUE (the default) gives a panel per setting of newdata, cut down to quartile values when there are many; "all" a panel for every setting; FALSE no panels, with a colour per setting.
colour_by	An optional character string naming a numeric covariate of newdata, to colour the curves by.
interval	A single logical value, whether to draw the credible interval. Defaults to TRUE, and to FALSE with colour_by.
time_unit	An optional character string giving the time unit for the axis label, e.g. "min". The fit does not record the unit of its times, so none is shown unless given.

Details

By default each setting of newdata has its own panel, following the same rule as in `plot_tdt()`. There are two ways to draw the settings together in one panel. With `facet = FALSE` each setting has its own colour, from teal to deep purple in the order of newdata, and an entry in the legend, which suits a few settings or a covariate that is not numeric; with more than eight settings the colours are too close to tell apart, and a message says to use panels or `colour_by`. With `colour_by`, the curves of every value of one numeric covariate are coloured on a continuous scale from teal for its lowest value to deep purple for its highest, and the intervals are left out by default, since they overlap.

With `temperature = TRUE` the record of temperature is drawn above the survival curve on the same time axis, so that a drop in survival can be read against the temperatures that caused it. A co-stressor the record follows through time is drawn in a panel of its own between the two, named by its column, when it changes along the record; one that stays constant is left out.

Value

A ggplot object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))
# a two-day record read every hour, with a daily cycle
hours <- 0:48
record <- data.frame(time = hours * 60, day = pmin(hours %% 24, 1),
                     temp = 24 + 3 * sin(2 * pi * (hours - 9) / 24))
```

```

pred <- predict_survival(fit, record, newdata = data.frame(mass = c(8, 12)))
plot_survival(pred, time_unit = "min")
plot_survival(pred, temperature = TRUE, time_unit = "min")

# both masses in one panel, the record drawn once
plot_survival(pred, temperature = TRUE, facet = FALSE, time_unit = "min")

## End(Not run)

```

plot_tdt

*Plot the thermal death time curves***Description**

Draws the thermal death time curves from `derive_tdt_ttl()`: the time until a fraction p of individuals has failed, against constant temperature, as the posterior median with its credible interval. The result is a ggplot object, which can be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```

plot_tdt(
  x,
  facet = TRUE,
  colour_by = NULL,
  interval = is.null(colour_by),
  time_unit = NULL
)

```

Arguments

<code>x</code>	A list, the output of <code>derive_tdt_ttl()</code> .
<code>facet</code>	A single logical value or the character string "all". TRUE (the default) gives a panel per setting of newdata, cut down to quartile values when there are many; "all" a panel for every setting; FALSE no panels.
<code>colour_by</code>	An optional character string naming a numeric covariate of newdata, to colour the curves by instead of by p .
<code>interval</code>	A single logical value, whether to draw the credible interval. Defaults to TRUE, and to FALSE with <code>colour_by</code> .
<code>time_unit</code>	An optional character string giving the time unit for the axis label, e.g. "min". The fit does not record the unit of its times, so none is shown unless given.

Details

By default each fraction p has its own colour, from blue for the fewest failed to red for the most, and each setting of newdata its own panel. Each setting has its own curve, so settings are shown or left out, never averaged: with more than six, a numeric covariate with more than five values is cut down to the values nearest its minimum, quartiles and maximum, and a message says which were kept.

With colour_by, the curves of every value of one numeric covariate are drawn together, coloured on a continuous scale from teal for its lowest value to deep purple for its highest, with a panel per fraction p . This shows how the curve shifts with the covariate. The intervals are then left out by default, since they overlap. Where the covariate is itself a temperature, such as an acclimation temperature, a cool-to-warm scale may read better, and can be added to the result: `+ ggplot2::scale_colour_gradient(low = "#2C7BB6", high = "#D7191C")`.

Time is on a log scale, on which an exponential failure rate gives straight lines. Where a share of the posterior never reaches the fraction, the interval runs to the top of the panel, and where the median itself is never reached the line stops.

Value

A ggplot object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

tol <- derive_tdt_ttl(fit, newdata = data.frame(mass = c(8, 12)), t_ref = 60)
plot_tdt(tol, time_unit = "min")

# how the curves shift with body mass
tol <- derive_tdt_ttl(fit, newdata = data.frame(mass = seq(5, 15, by = 0.5)),
                    t_ref = 60)
plot_tdt(tol, colour_by = "mass", time_unit = "min")

## End(Not run)
```

Description

Draws the thermal tolerance landscape from `derive_tdt_ttl()`: survival over constant temperature and elapsed exposure time. The result is a ggplot object, to be refined by the user with further ggplot2 layers, scales, labels and themes.

Usage

```
plot_ttl(
  x,
  type = c("surface", "curves"),
  facet = TRUE,
  Ta = NULL,
  survival = c(0.1, 0.5, 0.9),
  labels = TRUE,
  time_unit = NULL
)
```

Arguments

<code>x</code>	A list, the output of <code>derive_tdt_ttl()</code> .
<code>type</code>	A character string, either "surface" (the default) for the landscape as a map, or "curves" for survival curves at a few temperatures.
<code>facet</code>	A single logical value or the character string "all". TRUE (the default) gives a panel per setting of newdata, cut down to quartile values when there are many; "all" a panel for every setting; FALSE no panels.
<code>Ta</code>	An optional numeric vector of the temperatures to draw for <code>type = "curves"</code> , each replaced by the nearest temperature of the landscape's grid. NULL (the default) takes the grid temperatures nearest its minimum, quartiles and maximum.
<code>survival</code>	A numeric vector of survival levels between 0 and 1, at which to draw the contour lines for <code>type = "surface"</code> .
<code>labels</code>	A single logical value, whether to label each contour line of <code>type = "surface"</code> with its survival level.
<code>time_unit</code>	An optional character string giving the time unit for the axis label, e.g. "min". The fit does not record the unit of its times, so none is shown unless given.

Details

With `type = "surface"` the whole landscape is drawn as a map, elapsed time across and temperature up, coloured by the posterior median survival from red (most have failed) through pale yellow (a survival of one half) to blue (most survive), with contour lines at the survival levels in `survival`, each labelled with its level. The credible interval is not shown on it. With `type = "curves"` survival is drawn against elapsed time at a few temperatures, each as the posterior median with its credible interval.

Elapsed time is on a log scale, unless the times include zero. The panels follow the same rule as in `plot_tdt()`, except that the map needs a panel for each setting of newdata, so `facet = FALSE` is refused for it when there are several.

Value

A ggplot object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                        checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

tol <- derive_tdt_ttl(fit, newdata = data.frame(mass = c(8, 12)), t_ref = 60)
plot_ttl(tol, time_unit = "min")
plot_ttl(tol, type = "curves", Ta = c(22, 25, 28), time_unit = "min")

## End(Not run)
```

plot_ttl_3d

Plot the thermal tolerance landscape as a surface to rotate

Description

Draws the thermal tolerance landscape from `derive_tdt_ttl()` in three dimensions: elapsed exposure time and constant temperature across the floor, and the posterior median survival as the height. The figure is interactive, so it can be rotated, zoomed and read by hovering, in a viewer, a browser or an HTML document.

Usage

```
plot_ttl_3d(x, row = NULL, interval = FALSE, grid = FALSE, time_unit = NULL)
```

Arguments

x	A list, the output of <code>derive_tdt_ttl()</code> .
row	An optional single whole number, the setting of newdata to draw, as its row number among the settings. NULL (the default) draws the first, with a message when there are several.
interval	A single logical value, whether to add the lower and upper bounds of the credible interval as two faint surfaces around the median.

grid	A single logical value, or one or two whole numbers. TRUE draws a grid on the median surface, ten lines at constant temperature and ten at constant elapsed time. A number sets how many lines each way, and two numbers the lines at constant temperature and at constant time, e.g. c(5, 15). FALSE (the default) draws none.
time_unit	An optional character string giving the time unit for the axis label, e.g. "min".

Details

It needs the plotly package, and returns a plotly object rather than a ggplot, so it is refined with plotly's own functions (for example `plotly::layout()`), not with ggplot2 layers. A three-dimensional surface hides part of itself in a static figure; for print, `plot_ttl()` shows the same landscape as a map.

One setting of newdata is drawn at a time. Elapsed time is on a log scale, unless the times include zero.

Value

A plotly object.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

tol <- derive_tdt_ttl(fit, newdata = data.frame(mass = c(8, 12)), t_ref = 60)
plot_ttl_3d(tol, row = 2, time_unit = "min")

## End(Not run)
```

predict_survival	<i>Survival along a record of temperature, from a fitted failure-rate model</i>
------------------	---

Description

Predicts, from a model fitted with `fit_hazard()`, the survival of an individual exposed to a measured record of temperature: the probability of surviving to each time of the record and of failing by its end, each with its posterior median and credible interval, at the covariate values in each row of newdata. Random effects are set to zero, so the prediction is for a typical group.

Usage

```
predict_survival(
  fit,
  record,
  newdata = NULL,
  levels = NULL,
  by = NULL,
  prob = 0.95,
  n_draws = 1000,
  grid_dT = 0.1
)
```

Arguments

<code>fit</code>	A <code>brmsfit</code> , a model fitted with <code>fit_hazard()</code> .
<code>record</code>	A data frame with the columns <code>time</code> and <code>temp</code> , the record of temperature through time, in the time unit and temperature scale of the data the model was fitted to, with times increasing and no gaps. It may also hold the column of a co-stressor and the column named in <code>by</code> .
<code>newdata</code>	A data frame with one row per set of covariate values to predict at, holding every covariate in the parameters' formulas. <code>NULL</code> (the default) suits a model with no covariates.
<code>levels</code>	A named numeric vector giving the level of each co-stressor that <code>record</code> does not follow, on its measurement scale, e.g. <code>c(D0 = 6)</code> .
<code>by</code>	A character string, the name of a column of <code>record</code> giving the period each reading belongs to (e.g., a day). When given, the result has each period's share of the hazard accrued over the record. An interval between two readings belongs to the period of the first.
<code>prob</code>	A single number between 0 and 1, the coverage of the credible intervals. Defaults to 0.95.
<code>n_draws</code>	A single number, the number of posterior draws to use, evenly spaced through the posterior. Defaults to 1000. The draws are held for every step along the record, so a long record with many draws needs memory in proportion to their product.
<code>grid_dT</code>	A single positive number, the largest change in temperature within a step along the record, in degrees. Between two readings the temperature is taken to change linearly, and the hazard is accrued in steps over which it changes by at most this much, since the failure rate at the midpoint of a step stands for the whole of it (see <i>The time grid</i> in <code>build_hazard_data</code>). Defaults to 0.1; <code>Inf</code> takes one step between consecutive readings.

Details

Along the record the cumulative hazard grows, over each step, by $M(T) (H_0(\text{tau}_1) - H_0(\text{tau}_0))$: the failure rate at the temperature in the middle of the step, times the growth of the cumulative baseline hazard over it. A step is the interval between two readings, or a part of it where the

temperature changes by more than `grid_dT` between them. Survival to a time is $\exp(-\text{cumulative hazard up to it})$. With the memoryless baseline the growth of H_0 is the length of the interval, so the cumulative hazard is the failure rate summed over the record, and it does not matter where the record begins. With the other baselines the hazard depends on elapsed exposure, which is counted from the first time in the record.

Value

A list of tibbles, each with the columns of `newdata`:

- `survival`: for each time of the record, the posterior mean, median, lower and upper of the probability of surviving to it (1 at the first time).
- `mortality`: the same for the probability of failing by the end of the record.
- `risk` (when `by` is given): each period's share of the hazard accrued over the record. The shares of a posterior draw add to one; `n_draws` is the number of draws with any hazard, which the summary rests on.
- `coverage`: the lowest and highest temperatures at which failures were observed in the data (`Ta_failed_min`, `Ta_failed_max`); the share of the record's duration below, within and above them; the share of the hazard accrued along the record at those temperatures (`risk_below`, `risk_within`, `risk_above`); the longest exposure in the data (`tau_max`); and the share of the record's duration that comes after it (beyond) with the share of the hazard accrued there (`risk_beyond`). Each share of the hazard is the median over the posterior draws with any hazard.
- `record`: the times and temperatures of the record, with any co-stressor it follows, kept so that `plot_survival()` can draw them above the survival they led to.

Outside the temperatures and exposures of the assay

An assay is run at temperatures that cause failure within its duration, and a field record may lie below them. At the temperatures at which failures were observed the data hold the failure-rate function in place, so functions that fit equally well predict alike. Below them nothing does, and the prediction depends on which function was chosen: functions that fit the assay equally well can differ by orders of magnitude there. The temperatures that count are those at which individuals failed, and not all those they were exposed to. In a ramping assay every individual starts at a temperature at which none fails, and the time spent there shows only that failures were rare over that time.

The coverage table gives the share of the record below, within and above the temperatures at which failures were observed, by duration and by the risk accrued. The second is the one to read: a record may spend most of its time at cool temperatures where little hazard accrues, and take nearly all of its risk from a few warm hours that the assay did cover.

A record may also last longer than any exposure in the assay. With a baseline other than the memoryless one, the prediction then rests on how the baseline extends beyond the exposures the data hold, and with the memoryless one on the assumption that the hazard does not change with the time under exposure, which the data support only over those exposures. The coverage table also gives the longest exposure in the data and the share of the record, by duration and by the risk accrued, that comes after it.

A message is given when any of the record lies outside the temperatures or after the longest exposure. Comparing the predictions of several failure-rate functions and baselines, or averaging them with `average_survival()`, shows how much the prediction depends on those choices.

The zero-type failure rates (zeroexp, zeropower) are zero at the reference temperature T_r and are taken as zero below it. The power function is not defined below T_r , and a record that goes below it is refused.

The mean and the median

Each posterior draw gives a probability of surviving, and the tables summarise those probabilities by their mean and by their median. The mean is the probability of surviving with the uncertainty in the parameters taken into account (and, from `average_survival()`, the uncertainty about the model, as far as the weights measure it): it is the fraction of individuals expected to survive in a typical group (random effects at zero). The median is the value that half of the posterior lies below. The two are close when the posterior is concentrated, and differ when it is skewed or has two separate groups of values, as it can when models that disagree are averaged; the credible interval should then be reported with either.

No recovery

The hazard accrued along the record is only ever added to, so the prediction assumes that an individual does not recover when the temperature returns to a range in which repair is expected to exceed thermal damage. Along a record that moves in and out of stressful temperatures, such as one with a daily cycle, the predicted mortality is then too high if individuals recover in between, and the more so the more cycles there are.

Co-stressors

For a model with co-stressors, a co-stressor whose column is in record is followed through time, taking in each interval the value halfway between the two readings. One that is not in record is held at the level given in levels.

Examples

```
## Not run:
# a model fitted to a small ramping trial: 20 individuals warmed from 20 to 35
# degrees over 60 minutes and checked every 10, in two trials (see fit_hazard())
path <- data.frame(path_id = "A", time = c(0, 60), temp = c(20, 35))
checks <- data.frame(path_id = "A", time = seq(0, 60, by = 10))
ind <- data.frame(id = 1:20, path_id = "A", event = 1, mass = runif(20, 8, 12),
                  trial = rep(c("a", "b"), each = 10),
                  t_upper = sample(seq(10, 60, 10), 20, replace = TRUE))
long <- build_hazard_data(ind, path, "interval", covariates = c("mass", "trial"),
                         checks = checks)
fit <- fit_hazard(long, "exp", dpar_formulas = list(loga = ~ 1 + mass))

pred <- predict_survival(fit, record, newdata = data.frame(mass = 10), by = "day")
pred$mortality
pred$risk

## End(Not run)
```

ramp_assay	<i>A simulated ramping assay of thermal tolerance, with two levels of oxygen</i>
------------	--

Description

Times to loss of equilibrium of 320 juvenile salmonids warmed from 14 degrees C at a constant rate until they failed or the water reached 29.5 degrees C, simulated from a known model. The fish were run in 32 trials of ten. Four ramp rates (0.025, 0.1, 0.2 and 0.3 degrees C per minute) are crossed with two levels of dissolved oxygen (10 and 5 mg/L), four trials of each combination. The fish were watched throughout, so each failure was observed at a known time.

Usage

```
ramp_assay
```

Format

A tibble with 320 rows, one per fish, and 8 columns:

id Integer, the fish.

trial Character, the trial the fish was in ("R01" to "R32").

path_id Character, the ramp ("r0.025", "r0.1", "r0.2", "r0.3"), the key to its temperature trajectory.

rate Numeric, the ramp rate, degrees C per minute.

do Numeric, dissolved oxygen in the trial, mg/L (10 or 5).

mass Numeric, body mass, g.

event Integer, 1 if the fish lost equilibrium, 0 if it survived to the end of the ramp.

t_event Numeric, the time of failure, or of the end of the ramp for a survivor, minutes from its start.

Details

The slow rate allows the same data to show what the choice of ramp rates lets a model estimate: the three faster rates span 3-fold, and the rates of 0.025, 0.1 and 0.3 span 12-fold (see the vignette *Ramping assays*).

The data are simulated, not measured. The values are plausible for a juvenile rainbow trout acclimated to 14 degrees C, and stand for no particular study.

The model the data were simulated from

The hazard of losing equilibrium is $h_0(\tau) M(T)$, with an exponential failure rate $M(T) = a \exp(b(T - 26))$ and a Weibull baseline $h_0(\tau) = (k / s) (\tau / s)^{(k - 1)}$, with times in minutes. The values, stored as `attr(ramp_assay, "truth")`, are $b = 1.7$ per degree C, $k = 1.5$, $s = 60$ minutes, and $\log a = -2.13$ at 26 degrees C for a 15 g fish in normoxia. $\log a$ varies among trials with a standard deviation of 0.3 and rises by 0.03 per gram of body mass. At a constant temperature half the fish

then fail in about 1 hour at 27 degrees C, 10 hours at 25 and 4 days at 23. [static_assay](#) was simulated from the same model.

Low dissolved oxygen raises the failure rate, and the more so the warmer the water: each mg/L of oxygen below 10 adds $0.3 + 0.04 (T - 26)$ to $\log M(T)$. The effect is 0.3 per mg/L at 26 degrees C and changes by 0.04 for each degree (0.38 at 28 degrees C, 0.22 at 24), which is the form that the interaction of a co-stressor with temperature fits (see [costressor\(\)](#)). The two values are stored as `b_do` and `d_do`.

Source

Simulated by `data-raw/make-example-data.R` in the package's repository.

See Also

[static_assay](#), [stream_record](#)

Examples

```
head(ramp_assay)

# The four ramps, from 14 to 29.5 degrees C.
rates <- c(0.025, 0.1, 0.2, 0.3)
path <- ramp_path(c("r0.025", "r0.1", "r0.2", "r0.3"), T_start = 14, rate = rates,
                  t_max = 15.5 / rates)
# No grid is given: the intervals follow the ramp in steps of 0.1 degrees C.
long <- build_hazard_data(ramp_assay, path, "continuous",
                        covariates = c("mass", "do", "trial"))
head(long)
```

ramp_path

Build a linear ramp temperature trajectory

Description

A convenience function for the `path` argument of [build_hazard_data](#), useful when there is no logged temperature series: when temperature follows a controlled linear ramp, $\text{temp}(t) = T_{\text{start}} + \text{rate} * t$, the nominal ramp stands in for a logged trace. It is vectorised over its arguments: give several ramps at once and it stacks them into one table keyed by `path_id`.

Usage

```
ramp_path(path_id, T_start, rate, t_max, by = NULL)
```

Arguments

path_id, T_start, rate, t_max

The ramp specification, one value per ramp: path_id the identifier, T_start the start temperature, rate in degrees per time unit, and t_max the duration. The four are recycled together, so equal-length vectors describe several ramps and a length-one argument is reused across all of them.

by

An optional numeric giving the sampling step in time. When NULL (the default) the path is just its two endpoints, which the linear interpolation in [build_hazard_data](#) reproduces exactly. A value such as by = 5 creates a row every by time units, which then serves as the check schedule in interval mode or a ready-made integration grid in continuous mode.

Value

A data frame with path_id, time and temp, ready to pass to [build_hazard_data](#) as its path.

Examples

```
# One ramp: 20 -> 35 degrees over 60 minutes, as a two-point path.
ramp_path("A", T_start = 20, rate = 0.25, t_max = 60)

# Two ramps at different rates, each sampled every 10 minutes (T_start recycled).
ramp_path(c("slow", "fast"), T_start = 20, rate = c(0.2, 0.4),
          t_max = c(75, 40), by = 10)
```

static_assay

A simulated static assay of thermal tolerance

Description

Times to loss of equilibrium of 120 juvenile salmonids held at three constant temperatures, simulated from a known model so that what a fitted model estimates can be set against the values the data were made from. Twelve trials of ten fish, four at each temperature (23, 25 and 27 degrees C), were checked every hour for five days, so each failure is known to have happened between two checks. Fish that had not failed by the end of the fifth day are survivors. The coolest temperature is one a summer heat wave can reach, so that the assay informs predictions for the field (see [stream_record](#)).

Usage

```
static_assay
```

Format

A tibble with 120 rows, one per fish, and 7 columns:

id Integer, the fish.

trial Character, the trial the fish was in ("S01" to "S12").

path_id Character, the temperature group ("T23", "T25", "T27"), the key to its temperature trajectory.

temp Numeric, the constant temperature of the trial, degrees C.

mass Numeric, body mass, g.

event Integer, 1 if the fish lost equilibrium, 0 if it survived to the end of the trial.

t_upper Numeric, the time of the check at which the fish was found to have failed, or of the last check for a survivor, minutes.

Details

The data are simulated, not measured. The values are plausible for a juvenile rainbow trout acclimated to 14 degrees C, and stand for no particular study. The model they come from is described under [ramp_assay](#), which shares it, and is stored with the data as `attr(static_assay, "truth")`.

Source

Simulated by `data-raw/make-example-data.R` in the package's repository.

See Also

[ramp_assay](#), [stream_record](#)

Examples

```
head(static_assay)

# The temperature trajectory of each group, and the checks.
path <- data.frame(path_id = rep(c("T23", "T25", "T27"), each = 2),
                  time = rep(c(0, 7200), 3),
                  temp = rep(c(23, 25, 27), each = 2))
checks <- expand.grid(path_id = c("T23", "T25", "T27"),
                    time = seq(0, 7200, by = 60))
long <- build_hazard_data(static_assay, path, "interval",
                        covariates = c("mass", "trial"), checks = checks)
head(long)
```

stream_record

A simulated record of stream temperature

Description

Thirty days of hourly water temperature, with a daily cycle and a heat wave in the third week that takes the afternoon maximum to about 24 degrees C. It is simulated, to give `predict_survival()` a record that lies below the temperatures of the example assays for most of its length and reaches the coolest of them, 23 degrees C, on the afternoons of the heat wave.

Usage

`stream_record`

Format

A tibble with 721 rows, one per hourly reading, and 3 columns:

time Numeric, minutes from the first reading.

day Numeric, the day of the record, 1 to 30.

temp Numeric, water temperature, degrees C.

Source

Simulated by `data-raw/make-example-data.R` in the package's repository.

See Also

[static_assay](#), [ramp_assay](#)

Examples

```
range(stream_record$temp)
tapply(stream_record$temp, stream_record$day, max)[15:21]
```

Index

- * **datasets**
 - ramp_assay, [57](#)
 - static_assay, [59](#)
 - stream_record, [60](#)
- assess_ramp_linearity, [3](#)
- average_dctmax, [4](#)
- average_dctmax(), [26](#)
- average_survival, [6](#)
- average_survival(), [46](#), [47](#), [55](#), [56](#)
- average_tdt_ttl, [8](#)
- brms::brm, [29](#), [30](#)
- brms::brm(), [39](#)
- brms_log_h0, [10](#), [17](#)
- brms_log_M, [10](#), [17](#), [42](#)
- build_hazard_data, [3](#), [7](#), [11](#), [15](#), [17](#), [18](#),
[21–23](#), [30](#), [32](#), [54](#), [58](#), [59](#)
- build_hazard_prior, [12](#), [14](#), [21](#), [23](#), [29](#), [30](#)
- build_hazard_spec, [12](#), [14](#), [15](#), [17](#), [21](#), [23](#),
[29](#), [30](#)
- check_survival, [19](#)
- check_survival(), [36](#), [44](#)
- cloglog_bern_family, [18](#), [21](#)
- costressor, [13](#), [15](#), [18](#), [21](#), [30](#)
- costressor(), [58](#)
- derive_dctmax, [25](#)
- derive_dctmax(), [4](#), [5](#)
- derive_tdt_ttl, [27](#)
- derive_tdt_ttl(), [3](#), [8](#), [9](#), [26](#), [42](#), [49](#), [51](#), [52](#)
- fit_hazard, [21](#), [23](#), [29](#)
- fit_hazard(), [4](#), [6](#), [8](#), [14](#), [15](#), [19](#), [21](#), [25](#), [27](#),
[35](#), [38–40](#), [43](#), [53](#), [54](#)
- h0_catalogue, [10](#), [12](#), [15](#), [17](#), [18](#), [28](#), [31](#), [33](#),
[34](#), [42](#)
- H0_of, [33](#)
- H0_of(), [34](#)
- H0inv_of, [34](#)
- H0inv_of(), [33](#)
- hazard_mixed_family, [18](#), [35](#)
- hazard_residuals, [35](#)
- hazard_residuals(), [20](#), [45](#)
- hazard_weights, [37](#)
- hazard_weights(), [4](#), [6](#), [8](#)
- kfold_hazard, [38](#)
- kfold_hazard(), [16](#), [37](#), [38](#), [40](#)
- loo::kfold(), [39](#)
- loo::loo(), [41](#)
- loo::loo_compare(), [39](#), [40](#)
- loo_hazard, [40](#)
- loo_hazard(), [37–39](#)
- M_catalogue, [10–12](#), [15](#), [17](#), [18](#), [30](#), [32](#), [41](#)
- M_draws, [43](#)
- plot_check, [44](#)
- plot_check(), [20](#)
- plot_residuals, [45](#)
- plot_residuals(), [36](#)
- plot_risk, [46](#)
- plot_survival, [47](#)
- plot_survival(), [55](#)
- plot_tdt, [49](#)
- plot_tdt(), [9](#), [46](#), [48](#), [51](#)
- plot_ttl, [50](#)
- plot_ttl(), [9](#), [53](#)
- plot_ttl_3d, [52](#)
- plot_ttl_3d(), [9](#)
- plotly::layout(), [53](#)
- predict_survival, [53](#)
- predict_survival(), [6](#), [7](#), [46](#), [47](#), [60](#)
- ramp_assay, [57](#), [60](#), [61](#)
- ramp_path, [3](#), [58](#)
- set.seed(), [20](#), [36](#), [37](#), [39](#)

`static_assay`, [58](#), [59](#), [61](#)
`stats::update()`, [39](#)
`stream_record`, [58](#), [59](#), [60](#), [60](#)