Package: stagelife (via r-universe)

November 3, 2026

Type Package

Title Random Number Generation and Inferences for Two-Stage Life-Testing Data

Version 0.1.0

Description Provides random number generation, estimation, and hypothesis testing for two-stage life-testing experiments in which units begin on a baseline stress and, at a fixed or random change point, some or all surviving units move to an elevated stress (` `stage life testing"), following the cumulative-exposure model of Laumen and Cramer (2019), ` `Stage life testing", <doi:10.1002/nav.21874>. Twelve baseline lifetime distributions are supplied (Weibull, Power Lindley, Exponentiated Lindley, Exponentiated Exponential, Marshall-Olkin Exponential, Lomax, Exponentiated Teissier, Log-Logistic, Chen, Log-normal, Gamma, and Power Muth/Power Teissier) together with three parameter-sharing cases (common, shape1, shape11). The package implements data simulation under both the fixed-proportion (typeP) and fixed-count (typeM) rules and the simple-step-stress special case, maximum likelihood estimation using a hybrid Nelder-Mead and BFGS optimisation routine, asymptotic (Wald) and non-parametric/parametric bootstrap confidence intervals (including log-scale variants that guarantee a positive interval), a stage-1-observation planning probability, Kolmogorov-Smirnov goodness-of-fit testing with a parametric-bootstrap correction for estimated parameters, likelihood-ratio and information criteria for nested/non-nested model comparison, and diagnostic plots comparing fitted and empirical distributions, including the cumulative distribution function (CDF), survival function, probability-probability (P-P), and quantile-quantile (Q-Q) plots.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports compiler, ggplot2, graphics, lamW, parallel, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/vksharma-bhu/stagelife

BugReports https://github.com/vksharma-bhu/stagelife/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Vikas Kumar Sharma [aut, cre]

Maintainer Vikas Kumar Sharma <vksharma@bhu.ac.in>

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

Date/Publication 2026-11-03 15:23:11 UTC

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

RemoteRef HEAD

RemoteSha bd2831de9174160ed7304b31f26438e2cac18fba

Contents

coef.slt_fit . . . 3
confint.slt_fit . . . 3
fit_slt . . . 4
logLik.slt_fit . . . 5
loglik_slt . . . 6
plot.slt_fit . . . 7
plot_cdf . . . 7
plot_pp . . . 8
plot_qq . . . 9
plot_survival . . . 9
print.slt_fit . . . 10
print.summary.slt_fit . . . 10
pslt . . . 11
rslt . . . 12
slt_bobotas_kateri_2015 . . . 14
slt_bootstrap_ci . . . 14
slt_cases . . . 16
slt_dist . . . 17
slt_equiv_age . . . 21
slt_families . . . 22
slt_gof_ks . . . 23
slt_han_kundu_2014 . . . 24
slt_lrt . . . 25

slt_model_comparison 25
slt_prob_stage1 26
slt_reduce_params 27
slt_tau 28
slt_wang_fei_2003 29
slt_zhu_2010 29
summary.slt_fit 30
vcov.slt_fit 31

Index **32**

`coef.slt_fit` *Extract coefficients from a fitted two-stage life testing model*

Description

Extract coefficients from a fitted two-stage life testing model

Usage

```
## S3 method for class 'slt_fit'
coef(object, ...)
```

Arguments

`object`	An `"slt_fit"` object from `fit_slt()`.
`...`	Currently unused.

Value

The named numeric vector `object$par` (natural, positive scale; names and length depend on `object$case`, see `slt_cases()`).

`confint.slt_fit` *Asymptotic (Wald) confidence intervals for a fitted two-stage life testing model*

Description

Two Wald-type intervals are available: `"normal"` and `"lognormal"`.

Usage

```
## S3 method for class 'slt_fit'
confint(object, parm, level = 0.95, type = c("lognormal", "normal"), ...)
```

Arguments

object	An "slt_fit" object from fit_slt().
parm	Optional character vector of parameter names to include (default: all).
level	Confidence level.
type	"lognormal" (default) or "normal".
...	Currently unused.

Details

"lognormal" (the default) builds the interval on the log(theta) scale (where the MLE is asymptotically normal by construction) and exponentiates, which guarantees a positive interval and is generally preferred for these positive shape/scale parameters. "normal" applies the delta-method standard error directly on the natural scale and can (rarely) produce a negative lower limit for small samples.

Value

A numeric matrix with one row per parameter and two columns (lower, upper).

Examples

```
set.seed(1)
d <- rslt(200, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
confint(fit)
```

---

fit_slt *Fit a two-stage life testing model by maximum likelihood*

Description

Fit the stage life testing model for given data using the cumulative-exposure method for one of 12 baseline distributions and one of three parameter cases.

Usage

```
fit_slt(
  data,
  family = "weibull",
  case = c("common", "shape1", "shape11"),
  tau1,
  init = NULL,
  n_starts = 4L,
  bootstrap = FALSE,
  control = list()
)
```

Arguments

data	A list with numeric vectors Y (stage-0 observed failure times) and Z (stage-1 observed failure times), as produced by rslt() (or assembled directly from real data).
family	One of slt_families().
case	One of "common", "shape1", "shape11" (see slt_cases()).
tau1	The stage-change time, used to shift Z back to the stage-1 origin.
init	Optional numeric vector of starting values **on the natural scale**, in the order given by case_info(case)$par_names (see slt_cases() for the parameter order for each case). If NULL a data-driven default is used.
n_starts	Number of (perturbed) starting values to try; the converged fit with the highest log-likelihood is returned.
bootstrap	If TRUE, use looser/faster optimiser control settings appropriate for repeated bootstrap refits.
control	Optional list overriding stats::optim() control parameters for both the Nelder-Mead and BFGS stages.

Details

Estimation uses a Nelder-Mead warm start followed by BFGS (with a numerical Hessian for inference), repeated from several perturbed starting values to guard against local optima; the converged fit with the highest log-likelihood is returned.

Value

An object of class "slt_fit", a list with elements including converged, family, case, tau1, par (natural-scale estimates), se, theta0, theta1, vcov_log, loglik, aic, bic. If no starting value converges, converged is FALSE and most other elements are absent.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, family = "weibull", case = "shape1", tau1 = 5)
summary(fit)
```

---

logLik.slt_fit *Extract the log-likelihood of a fitted two-stage life testing model*

Description

Extract the log-likelihood of a fitted two-stage life testing model

Usage

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

Arguments

object	An "slt_fit" object from fit_slt().
...	Currently unused.

Value

An object of class "logLik" (the fitted log-likelihood, NA_real_ if object did not converge, with "df"/"nobs" attributes set for use by stats::AIC()/stats::BIC()).

---

loglik_slt *Two-stage cumulative-exposure log-likelihood*

Description

Two-stage cumulative-exposure log-likelihood

Usage

```
loglik_slt(data, family, theta0, theta1, tau1)
```

Arguments

data	A list with numeric vectors Y (stage-0 observed failure times) and Z (stage-1 observed failure times), as returned by rslt().
family	A family key (see slt_families()) or family object.
theta0	Numeric c(lambda0, alpha0), baseline-stage parameters.
theta1	Numeric c(lambda1, alpha1), elevated-stage parameters.
tau1	The stage-change time used to shift Z back to the stage-1 time origin.

Value

The log-likelihood (a scalar), or -Inf for parameter values outside the support of the model.

Examples

```
set.seed(1)
d <- rslt(80, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
loglik_slt(d, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5)
```

plot.slt_fit *Diagnostic plots for a fitted two-stage life-testing model*

Description

See plot_cdf() for an important caveat: these assume the classical simple-step-stress design (prop_transfer = 1 in rslt()).

Usage

```
## S3 method for class 'slt_fit'
plot(x, which = c("cdf", "survival", "pp", "qq", "all"), ...)
```

Arguments

x	A converged "slt_fit" object.
which	One of "cdf", "survival", "pp", "qq", or "all" (returns a named list of all four ggplot objects).
...	Currently unused.

Value

A ggplot object, or (if which = "all") a named list of four ggplot objects.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
plot(fit, which = "cdf")
```

plot_cdf *Fitted vs empirical CDF plot*

Description

Plots the fitted population CDF (pslt()) against the empirical CDF of the observed data.

Usage

```
plot_cdf(fit)
```

Arguments

fit	A converged "slt_fit" object, fit to data from a simple-step-stress (rslt() with prop_transfer = 1) design.

Details

Assumes every survivor was transferred at `tau1` (the classical simple-step-stress design, `prop_transfer = 1` in `rslt()`) – see the note in `slt_gof_ks()` for why: the pooled `Y/Z` sample only has `pslt()` as its correct marginal CDF in that case. This applies to all four diagnostic plots in this file.

Value

A `ggplot` object.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
plot_cdf(fit)
```

`plot_pp` *P-P (probability-probability) plot*

Description

P-P (probability-probability) plot

Usage

```
plot_pp(fit)
```

Arguments

`fit`	A converged `"slt_fit"` object, fit to data from a simple-step-stress (`rslt()` with `prop_transfer = 1`) design.

Value

A `ggplot` object.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
plot_pp(fit)
```

plot_qq *Q-Q (quantile-quantile) plot*

Description

Q-Q (quantile-quantile) plot

Usage

```
plot_qq(fit)
```

Arguments

`fit`	A converged `"slt_fit"` object, fit to data from a simple-step-stress (`rslt()` with `prop_transfer = 1`) design.

Value

A `ggplot` object.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
plot_qq(fit)
```

plot_survival *Fitted vs empirical survival curve*

Description

Fitted vs empirical survival curve

Usage

```
plot_survival(fit)
```

Arguments

`fit`	A converged `"slt_fit"` object, fit to data from a simple-step-stress (`rslt()` with `prop_transfer = 1`) design.

Value

A `ggplot` object.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
plot_survival(fit)
```

---

print.slt_fit *Print a fitted two-stage life testing model*

Description

Print a fitted two-stage life testing model

Usage

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

Arguments

x	An "slt_fit" object from fit_slt().
digits	Number of digits to round to.
...	Currently unused.

Value

x, invisibly.

See Also

summary.slt_fit() for a more detailed summary with confidence limits.

---

print.summary.slt_fit *Print a summary of a fitted two-stage life testing model*

Description

Print a summary of a fitted two-stage life testing model

Usage

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

Arguments

x	A "summary.slt_fit" object from summary.slt_fit().
digits	Number of digits to round to.
...	Currently unused.

Value

x, invisibly.

---

pslt *Population CDF/PDF/quantile function of a two-stage step-stress model*

Description

pslt(), dslt() and qslt() implement the full stage life testing design, in which every unit switches from the baseline to the elevated stress at a common change-point tau1. Combined with the cumulative-exposure equivalent age slt_equiv_age(), the two branches (pre-/post-tau1) are pieced together into a single, continuous population CDF.

Usage

```
pslt(t, family, theta0, theta1, tau1)

dslt(x, family, theta0, theta1, tau1)

qslt(p, family, theta0, theta1, tau1)
```

Arguments

t, x	Numeric vector of times at which to evaluate the CDF/PDF.
family	A family key (see slt_families()) or family object.
theta0	Numeric c(lambda0, alpha0), baseline-stage parameters.
theta1	Numeric c(lambda1, alpha1), elevated-stage parameters.
tau1	Scalar change-point time (same argument as tau in slt_equiv_age()).
p	Numeric vector of probabilities at which to evaluate the quantile function.

Details

For a single unit switching from stage 0 to stage 1 at the fixed time tau1 (every unit switches – the classical simple-step-stress design, matching prop_transfer = 1 in rslt()), the CEM construction (see slt_equiv_age()) gives a population cdf

$$F(t;\theta_0,\theta_1,\tau_1) = \begin{cases} F_0(t;\theta_0), & 0 < t \le \tau_1, \\ F_1(t-\tau_1+\nu_1;\ \theta_1), & t > \tau_1, \end{cases}$$

where $\nu_1 = \nu(\tau_1; \theta_0, \theta_1)$ is the stage-1 equivalent age (`slt_equiv_age()`) at the moment of the switch. Differentiating – a constant *shift* of the argument by $\tau_1 - \nu_1$, not a general change of variables, so no extra Jacobian factor arises – gives the matching population pdf

$$f(t; \theta_0, \theta_1, \tau_1) = \begin{cases} f_0(t; \theta_0), & 0 < t \le \tau_1, \\ f_1(t - \tau_1 + \nu_1;\ \theta_1), & t > \tau_1. \end{cases}$$

F is continuous at τ_1 by construction of ν_1 ($F_1(\tau_1 - \tau_1 + \nu_1; \theta_1) = F_1(\nu_1; \theta_1) = F_0(\tau_1; \theta_0)$, verified as an automatic consequence of the equal-reliability definition of ν_1, not a separate assumption) – but F is generally *not* differentiable at τ_1 (the hazard rate jumps there), an intrinsic, well-known feature of the cumulative exposure model, not an artefact of this implementation. `qslt()` inverts F directly (analytically, not numerically): for $p \le F_0(\tau_1; \theta_0)$ it returns $F_0^{-1}(p; \theta_0)$; otherwise it returns $\tau_1 - \nu_1 + F_1^{-1}(p; \theta_1)$, using each stage's own quantile function (itself closed-form for every family in `slt_families()`).

Value

A numeric vector the same length as t/x/p.

See Also

`slt_equiv_age()` for nu1; `loglik_slt()` for the estimation log-likelihood; `rslt()` for simulating data under the actual random-withdrawal design; `fit_slt()` for maximum likelihood estimation.

Examples

```
pslt(c(1, 6, 10), "weibull", c(0.1, 1.5), c(0.3, 1.5), tau1 = 5)
```

---

rslt *Simulate data from the two-stage life-testing model*

Description

Simulates data from the two-stage life-testing model of Laumen & Cramer (2019, Procedure 2.3).

Usage

```
rslt(
  n,
  family,
  theta0,
  theta1,
  tau1,
  design = c("typeP", "typeM"),
  prop_transfer = 0.5,
  R0_1 = NULL
)
```

Arguments

n	Number of units placed on test.
family	A family key (see slt_families()) or family object.
theta0	Numeric c(lambda0, alpha0), baseline-stage parameters.
theta1	Numeric c(lambda1, alpha1), elevated-stage parameters.
tau1	Time at which the transfer decision is made.
design	"typeP" (default, fixed target proportion) or "typeM" (fixed target count); see Details.
prop_transfer	Proportion (in [0, 1]) of survivors at tau1 transferred to the elevated stage, for design = "typeP". Set to 1 for the classical simple-step-stress design.
R0_1	Target number of survivors transferred (capped by the number actually available), for design = "typeM".

Details

At the stage-change time tau1, R*_1 of the units that have *not yet failed* are randomly selected and moved to the elevated stage (cumulative-exposure shifted); the remaining survivors continue failing under the original stage-0 conditions and are never observed under stage 1.

Two rules for generating R*_1 are supported, following the paper:

"typeP" A fixed target *proportion* pi1 of the survivors is transferred: R*_1 = floor(pi1 * (n - D1)), where D1 is the (random) number of stage-0 failures observed before tau1. pi1 = 1 transfers *every* survivor, which is exactly the classical "simple step-stress" design (Remark 2.2(4) of the paper) – no separate function is needed for that special case.

"typeM" A fixed target *count* R0_1 is transferred, capped by the number of available survivors: R*_1 = min(n - D1, R0_1).

Value

A list with Y (all stage-0 failure times: early failures before tau1 plus the non-transferred survivors, who keep failing under stage-0), Z (stage-1 observed failure times for the transferred units), n_early (D1, number failing before tau1) and n_transferred (R*_1).

Examples

```
set.seed(1)
d <- rslt(100, "weibull", theta0 = c(0.05, 1.5), theta1 = c(0.2, 1.5),
          tau1 = 5, design = "typeP", prop_transfer = 0.5)
str(d)

d_m <- rslt(100, "weibull", theta0 = c(0.05, 1.5), theta1 = c(0.2, 1.5),
            tau1 = 5, design = "typeM", R0_1 = 30)
str(d_m)

# The classical simple-step-stress design (every survivor transferred):
d_full <- rslt(100, "weibull", theta0 = c(0.05, 1.5), theta1 = c(0.2, 1.5),
               tau1 = 5, design = "typeP", prop_transfer = 1)
str(d_full)
```

slt_bobotas_kateri_2015 *Bobotas & Kateri (2015) simple step-stress data*

Description

A simple step-stress life-testing data set on the charge detrapping and dielectric breakdown of nanocrystalline embedded high-k devices, with 15 units observed under the baseline stress and 23 under the elevated stress after a common change-point at `tau1 = 600` (seconds).

Usage

```
slt_bobotas_kateri_2015
```

Format

A list with elements `y0`, `y1`, `tau1`, `description`, `source` (see slt_han_kundu_2014 for the structure).

Source

Bobotas, P. and Kateri, M. (2015). The step-stress tampered failure rate model under interval monitoring. *Statistical Methodology*, 27, 100-122.

See Also

slt_han_kundu_2014, slt_wang_fei_2003, slt_zhu_2010.

Examples

```
data(slt_bobotas_kateri_2015)
str(slt_bobotas_kateri_2015)
```

slt_bootstrap_ci *Bootstrap confidence intervals for a fitted two-stage life testing model*

Description

Computes percentile, bootstrap-t, and log-scale bootstrap-t confidence intervals via either nonparametric or a parametric bootstrap.

Usage

```
slt_bootstrap_ci(
  fit,
  B = 1000,
  level = 0.95,
  method = c("nonparametric", "parametric"),
  design = c("typeP", "typeM"),
  prop_transfer = 1,
  R0_1 = NULL,
  n_cores = 1L
)
```

Arguments

`fit`	A converged `"slt_fit"` object.
`B`	Number of bootstrap replicates.
`level`	Confidence level.
`method`	`"nonparametric"` (default) or `"parametric"`.
`design`	For `method = "parametric"`, the design used to simulate new data via `rslt()`: `"typeP"` (fixed target proportion, the default) or `"typeM"` (fixed target count).
`prop_transfer`	Proportion transferred, for `design = "typeP"`. Defaults to `1` (every survivor transferred, the classical simple-step-stress design, matching a typical published step-stress data set); set to the actual value used in your experiment if it was a genuine partial-withdrawal design.
`R0_1`	Target number transferred, for `design = "typeM"`.
`n_cores`	Number of cores for `parallel::mclapply()` (ignored on Windows, where replicates run sequentially).

Details

Non-parametric bootstrap resamples (case resampling) the observed `Y/Z` failure times; **parametric** bootstrap simulates fresh data from the fitted model at the MLE. The parametric bootstrap is generally more efficient when the fitted model is (approximately) correctly specified – see `slt_gof_ks()` – but is more sensitive to misspecification; the non-parametric bootstrap is robust to misspecification but, for small samples, can only reproduce combinations of already-observed values.

The plain (natural-scale) bootstrap-t interval, like the asymptotic Wald interval from `confint(..., type = "normal")`, is an ordinary linear pivot (`estimate -/+ quantile * se`) and can occasionally produce a negative lower bound for a positive parameter, especially at small `n` or for a skewed bootstrap distribution. The log-scale bootstrap-t interval applies the same pivot construction on `log(theta)` and exponentiates the result, which guarantees a positive interval by construction – directly analogous to why `confint(..., type = "lognormal")` (the default) is generally preferred over `type = "normal"` for the asymptotic interval. The percentile interval is always positive already (it is built from quantiles of the always-positive bootstrap estimates themselves), so only bootstrap-t needed a log-scale counterpart.

Value

An object of class "slt_boot_ci": a list with percentile, boot_t, and log_boot_t data.frames (columns lower, upper, one row per parameter), B, B_converged, method, design, level, and the raw boot_estimates matrix.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
slt_bootstrap_ci(fit, B = 100)
```

---

slt_cases *Identifiable parameter-sharing cases for two-stage estimation*

Description

Describes the three parameter-sharing structures between the baseline-stage parameters theta0 = (lambda0, alpha0) and the elevated-stage parameters theta1 = (lambda1, alpha1) supported for estimation. A fourth, fully unrestricted structure (alpha0 and alpha1 both free) is deliberately not supported; see Details.

Usage

```
slt_cases()
```

Details

Four parameter-sharing structures are logically possible for the pair $\theta_0 = (\lambda_0, \alpha_0)$, $\theta_1 = (\lambda_1, \alpha_1)$:

full :	$\theta_0 = (\lambda_0, \alpha_0)$,	$\theta_1 = (\lambda_1, \alpha_1)$	(4 free parameters)
common :	$\theta_0 = (\lambda_0, \alpha)$,	$\theta_1 = (\lambda_1, \alpha)$	(3 free parameters, shared shape)
shape1 :	$\theta_0 = (\lambda_0, \alpha_0)$,	$\theta_1 = (\lambda_1, 1)$	(3 free parameters)
shape11 :	$\theta_0 = (\lambda_0, 1)$,	$\theta_1 = (\lambda_1, 1)$	(2 free parameters)

This package supports the last three; full is deliberately **not** offered (see "Why full is excluded" below).

shape1/shape11 fix the elevated-stage shape to the *numeric value* 1 – this is a structural constraint, not a claim that the elevated stage becomes Exponential. The two only coincide for the four families whose alpha = 1 boundary happens to be Exponential (weibull, expexponential, moexponential, gamma; see ?slt_dist); for the other eight families, alpha1 = 1 is simply that family's own (otherwise unnamed) member at shape 1 – e.g. Chen with its shape fixed at 1, not an Exponential elevated stage. Fixing the shape is still meaningful regardless: it reduces the free parameter count the same way for every family (see "Why full is excluded" below), and "is the elevated-stage shape exactly 1?" is a concrete, testable nested hypothesis via slt_lrt() whether or not that constraint has a textbook name.

Each case defines a bijection between its free-parameter vector `par` (on the natural, positive scale, in the order given by `case_info(case)$par_names`) and `(theta0, theta1)`, implemented by `slt_expand_params()` and inverted by `slt_reduce_params()`:

$$\texttt{common}: \ (\lambda_0, \lambda_1, \alpha) \mapsto \big((\lambda_0, \alpha),\ (\lambda_1, \alpha)\big),$$

$$\texttt{shape1}: \ (\lambda_0, \alpha_0, \lambda_1) \mapsto \big((\lambda_0, \alpha_0),\ (\lambda_1, 1)\big),$$

$$\texttt{shape11}: \ (\lambda_0, \lambda_1) \mapsto \big((\lambda_0, 1),\ (\lambda_1, 1)\big).$$

`fit_slt()` optimises the log-likelihood (`loglik_slt()`, via `neg_loglik_slt()`) over `log(par)` in this order, then reports the fitted `par` under exactly these names – so, for instance, `case = "shape1"`'s second free parameter is always labelled `alpha0` and is always the baseline-stage shape, never accidentally the elevated-stage rate or any other quantity. This parameter-index bookkeeping (`case_info()$par_names`, `slt_expand_params()`, `slt_reduce_params()`, and `fit_slt()`'s internal default-initial-value heuristic all sharing one consistent order) is exactly the kind of place a silent mislabelling bug could hide, so it has been checked directly: the round trip `slt_expand_params(slt_reduce_params(theta1, case), case)` reproduces `(theta0, theta1)` exactly for all three cases, and a full simulate-then-refit check (large `n`, known `theta0`/`theta1`) recovers the true values under the correct parameter names for all three cases (see the package's test suite, `test-fit.R`).

`full` (both shapes free) is excluded because, with a single change-point and no replicated stress levels, its extra free parameter is empirically non-identifiable (a flat likelihood ridge, frequent non-convergence); `shape1`/`shape11` remove this by fixing one or both shapes to 1, `common` by tying them together instead. `shape11` is nested in both `common` and `shape1` (the natural null model for `slt_lrt()`); `shape1` and `common` are not nested in each other, so compare those two with `slt_model_comparison()` instead.

Value

A data.frame with columns `case`, `n_par`, `parameters`, `description`.

See Also

`slt_expand_params()` and `slt_reduce_params()` for the parameter mapping implementing each case; `fit_slt()` for estimation; `slt_lrt()` for the nested likelihood-ratio test; `slt_model_comparison()` for comparing non-nested cases.

Examples

```
slt_cases()
```

---

slt_dist *Baseline density, distribution and quantile functions for a distribution*

Description

Returns the single-stage d/p/q functions, each of `(x_or_p, lambda, alpha)`, for a baseline lifetime distribution. Useful for exploring a distribution in isolation from the two-stage step-stress structure. In every distribution, `lambda > 0` is a rate/scale-type parameter and `alpha > 0` is a shape/power parameter; see Details for the pdf/cdf of each and its literature source.

Usage

```
slt_dist(family)
```

Arguments

`family`	A family key from `slt_families()`.

Details

Below, $F(x;\lambda,\alpha)$ and $f(x;\lambda,\alpha) = F'(x;\lambda,\alpha)$ denote the cdf and pdf for $x > 0$ (both are 0 for $x \le 0$). Φ/ϕ are the standard normal cdf/pdf, and $\Gamma(\alpha, x) = \int_x^\infty t^{\alpha-1}e^{-t}\,dt$ is the upper incomplete gamma function.

Weibull (Weibull, 1951):

$$F(x;\lambda,\alpha) = 1 - \exp(-\lambda x^\alpha), \qquad f(x;\lambda,\alpha) = \lambda\alpha x^{\alpha-1}\exp(-\lambda x^\alpha).$$

Reduces to Exponential(rate = λ) at $\alpha = 1$.

Power Lindley (Ghitany, Al-Mutairi, Balakrishnan & Al-Enezi, 2013):

$$F(x;\lambda,\alpha) = 1 - \frac{1+\lambda+\lambda x^\alpha}{1+\lambda}\exp(-\lambda x^\alpha), \qquad f(x;\lambda,\alpha) = \frac{\alpha\lambda^2}{1+\lambda}(1+x^\alpha)\,x^{\alpha-1}\exp(-\lambda x^\alpha).$$

Reduces to (ordinary) Lindley(λ) at $\alpha = 1$ (not Exponential). The quantile function inverts a Lindley-type equation via the W_{-1} branch of the Lambert W function (`lamW::lambertWm1()`): the survival function's numerator-times- exponential (product) structure reduces to the $we^w = z$ form W solves.

Exponentiated Lindley (Nadarajah, Bakouch & Tahmasbi, 2011): with baseline Lindley cdf/pdf $G_0(x;\lambda) = 1 - \frac{1+\lambda+\lambda x}{1+\lambda}\exp(-\lambda x)$ and $g_0(x;\lambda) = \frac{\lambda^2}{1+\lambda}(1+x)\exp(-\lambda x)$ (Power Lindley's $\alpha = 1$ case above),

$$F(x;\lambda,\alpha) = G_0(x;\lambda)^\alpha, \qquad f(x;\lambda,\alpha) = \alpha\, g_0(x;\lambda)\, G_0(x;\lambda)^{\alpha-1}.$$

Reduces to (ordinary) Lindley(λ) at $\alpha = 1$ (not Exponential) – the same limit as **Power Lindley** above, of which this is the "exponentiate the cdf" sibling (exactly as Exponentiated Teissier and Power Muth are the two siblings of the Teissier baseline). Its quantile function reuses the same Lambert-W baseline-Lindley inverter as Power Lindley, just inverting a different target probability.

Exponentiated Exponential (Gupta & Kundu, 1999):

$$F(x;\lambda,\alpha) = \left(1-\exp(-\lambda x)\right)^\alpha, \qquad f(x;\lambda,\alpha) = \alpha\lambda\exp(-\lambda x)\left(1-\exp(-\lambda x)\right)^{\alpha-1}.$$

Reduces to Exponential(λ) at $\alpha = 1$.

Marshall-Olkin Exponential (Marshall & Olkin, 1997):

$$F(x;\lambda,\alpha) = 1 - \frac{\alpha\, e^{-\lambda x}}{1-(1-\alpha)e^{-\lambda x}}, \qquad f(x;\lambda,\alpha) = \frac{\lambda\alpha\, e^{-\lambda x}}{\left(1-(1-\alpha)e^{-\lambda x}\right)^2}.$$

Reduces to Exponential(λ) at $\alpha = 1$. Constructed by a different mechanism than the exponentiated/power-transform families here – Marshall & Olkin's "tilt parameter" construction, applicable to any baseline survival function – with α as the tilt parameter (an odds-ratio-type parameter, interpretable via the family's proportional-odds property).

Lomax (Lomax, 1954):

$$F(x;\lambda,\alpha) = 1-(1+\lambda x)^{-\alpha}, \qquad f(x;\lambda,\alpha) = \alpha\lambda(1+\lambda x)^{-\alpha-1}.$$

At $\alpha = 1$ this is simply the boundary member of the Lomax family itself (not a different named distribution).

Exponentiated Teissier (Sharma, Singh & Shekhawat, 2022): with baseline Teissier (1934) cdf/pdf $G_0(x;\lambda) = 1-\exp\{-(e^{\lambda x}-1-\lambda x)\}$ and $g_0(x;\lambda) = \lambda(e^{\lambda x}-1)\exp\{-(e^{\lambda x}-1-\lambda x)\}$,

$$F(x;\lambda,\alpha) = G_0(x;\lambda)^{\alpha}, \qquad f(x;\lambda,\alpha) = \alpha\, g_0(x;\lambda)\, G_0(x;\lambda)^{\alpha-1},$$

with quantile function (Sharma, Singh & Shekhawat, 2022, Proposition 4.2)

$$Q(p;\lambda,\alpha) = \frac{1}{\lambda}\log\left(-W_{-1}\left(\frac{p^{1/\alpha}-1}{e}\right)\right),$$

where W_{-1} is the lower (W_{-1}) branch of the Lambert W function (`lamW::lambertWm1()`). Reduces to baseline Teissier(λ) at $\alpha = 1$ (the same limit as **Power Muth** below, of which this is the "exponentiate the cdf" sibling). The Teissier distribution is also known as the Muth distribution; see Power Muth below, both verified against the cited paper's own closed-form quantile.

Log-Logistic / Fisk (Fisk, 1961):

$$F(x;\lambda,\alpha) = \frac{(\lambda x)^{\alpha}}{1+(\lambda x)^{\alpha}}, \qquad f(x;\lambda,\alpha) = \frac{\alpha\lambda(\lambda x)^{\alpha-1}}{(1+(\lambda x)^{\alpha})^2}.$$

The only family here with a non-monotonic (rise-then-fall) hazard rate for $\alpha > 1$; at $\alpha = 1$ it is simply Log-Logistic with shape 1 (not a different named distribution).

Chen (Chen, 2000):

$$F(x;\lambda,\alpha) = 1-\exp\left(\lambda\left(1-e^{x^{\alpha}}\right)\right), \qquad f(x;\lambda,\alpha) = \lambda\alpha x^{\alpha-1}e^{x^{\alpha}}\exp\left(\lambda\left(1-e^{x^{\alpha}}\right)\right).$$

Hazard rate is increasing for $\alpha \geq 1$ and bathtub-shaped for $\alpha < 1$.

Log-normal: parameterised here as

$$F(x;\lambda,\alpha) = \Phi\left(\frac{\log(\lambda x)}{\alpha}\right), \qquad f(x;\lambda,\alpha) = \frac{1}{\alpha x}\phi\left(\frac{\log(\lambda x)}{\alpha}\right),$$

i.e. $\log X \sim N(-\log\lambda,\ \alpha^2)$: $1/\lambda$ is the **median** (not the mean) and α is the log-scale standard deviation, keeping "λ a rate/scale parameter, α a shape parameter" consistent with every other family here. Built directly on `stats::plnorm()`-equivalent expressions (`stats::pnorm()`/`stats::dnorm()`/`stats::qnorm()`) rather than a hand-derived formula.

Gamma: standard shape-rate parameterisation,

$$F(x;\lambda,\alpha) = 1-\frac{\Gamma(\alpha,\lambda x)}{\Gamma(\alpha)}, \qquad f(x;\lambda,\alpha) = \frac{\lambda^{\alpha}}{\Gamma(\alpha)}x^{\alpha-1}e^{-\lambda x}.$$

Reduces to Exponential(λ) at $\alpha = 1$. Built directly on `stats::pgamma()`/`stats::dgamma()`/`stats::qgamma()` rather than a hand-derived formula.

Power Muth (also known as **Power Teissier**; Jodra, Gomez, Jimenez-Gamero & Alba-Fernandez, 2017, generalising the baseline Muth/Teissier distribution of Muth, 1977, and Teissier, 1934, and itself studied via the Lambert W function by Jodra, Jimenez-Gamero & Alba-Fernandez, 2015):

$$F(x;\lambda,\alpha) = 1-\exp\left(-\left(e^{\lambda x^\alpha}-1-\lambda x^\alpha\right)\right), \qquad f(x;\lambda,\alpha) = \lambda\alpha x^{\alpha-1}\left(e^{\lambda x^\alpha}-1\right)\exp\left(-\left(e^{\lambda x^\alpha}-1-\lambda x^\alpha\right)\right),$$

with quantile function

$$Q(p;\lambda,\alpha) = \left(\frac{1}{\lambda}\log\left(-W_{-1}\left(\frac{p-1}{e}\right)\right)\right)^{1/\alpha},$$

using the same Lambert-W identity as Exponentiated Teissier above (p in place of $p^{1/\alpha}$ there, since the power transform is applied to x here rather than to the cdf). Reduces to baseline Teissier/Muth(λ) at $\alpha = 1$ – the same limit as **Exponentiated Teissier** above, of which this is the "power the argument" sibling: both generalise the same baseline distribution, via $x \mapsto x^\alpha$ here versus $F_0 \mapsto F_0^\alpha$ there (exactly as Power Lindley and Exponentiated Exponential are, respectively, the "power the argument" and "exponentiate the cdf" generalisations of their own baselines).

Value

A list with functions `d`, `p`, `q`.

References

Chen, Z. (2000). A new two-parameter lifetime distribution with bathtub shape or increasing failure rate function. *Statistics & Probability Letters*, 49(2), 155-161.

Fisk, P. R. (1961). The graduation of income distributions. *Econometrica*, 29(2), 171-185.

Ghitany, M. E., Al-Mutairi, D. K., Balakrishnan, N., and Al-Enezi, L. J. (2013). Power Lindley distribution and associated inference. *Computational Statistics & Data Analysis*, 64, 20-33.

Gupta, R. D. and Kundu, D. (1999). Generalized exponential distributions. *Australian & New Zealand Journal of Statistics*, 41(2), 173-188.

Jodra, P., Gomez, H. W., Jimenez-Gamero, M. D., and Alba-Fernandez, M. V. (2017). The power Muth distribution. *Mathematical Modelling and Analysis*, 22(2), 186-201.

Jodra, P., Jimenez-Gamero, M. D., and Alba-Fernandez, M. V. (2015). On the Muth distribution. *Mathematical Modelling and Analysis*, 20(3), 291-310.

Lomax, K. S. (1954). Business failures: Another example of the analysis of failure data. *Journal of the American Statistical Association*, 49(268), 847-852.

Marshall, A. W. and Olkin, I. (1997). A new method for adding a parameter to a family of distributions with application to the exponential and Weibull families. *Biometrika*, 84(3), 641-652.

Nadarajah, S., Bakouch, H. S., and Tahmasbi, R. (2011). A generalized Lindley distribution. *Sankhya B*, 73(2), 331-359.

Muth, E. J. (1977). Reliability models with positive memory derived from the mean residual life function. In *The Theory and Applications of Reliability*, Vol. 2 (C. P. Tsokos and I. N. Shimi, eds.), 401-435. Academic Press, New York.

Sharma, V. K., Singh, S. V., and Shekhawat, K. (2022). Exponentiated Teissier distribution with increasing, decreasing and bathtub hazard functions. *Journal of Applied Statistics*, 49(2), 371-393. doi:10.1080/02664763.2020.1813694

Teissier, G. (1934). Recherches sur le vieillissement et sur les lois de la mortalite. *Annales de Physiologie et de Physicochimie Biologique*, 10, 237-284.

Weibull, W. (1951). A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18(3), 293-297.

See Also

`slt_families()` for the list of family keys; `dslt()`, `pslt()`, `qslt()`, `rslt()` for the two-stage (not baseline) population functions built from these; `slt_equiv_age()` for how a baseline distribution's cdf/quantile combine across the stage change.

Examples

```
fam <- slt_dist("weibull")
fam$p(1, lambda = 1, alpha = 2)
fam$q(0.5, lambda = 1, alpha = 2)
```

---

slt_equiv_age *Equivalent age under the elevated stress*

Description

Under the cumulative exposure model (CEM), a unit that has accumulated damage equivalent to surviving to time `tau` under stage-0 conditions has, under stage-1 conditions, already "used up" an equivalent age `nu`. This function computes `nu`.

Usage

```
slt_equiv_age(family, tau, theta0, theta1)
```

Arguments

`family`	A family key (see `slt_families()`) or family object.
`tau`	Non-negative scalar change-point time.
`theta0`	Numeric `c(lambda0, alpha0)`, baseline-stage parameters.
`theta1`	Numeric `c(lambda1, alpha1)`, elevated-stage parameters.

Details

The CEM's defining assumption is that only the *total accumulated damage* determines a unit's remaining life, not the stress history that produced it. Formalising "equal accumulated damage" as "equal reliability", a unit surviving to τ under the stage-0 distribution F_0 is equivalent to a unit that has survived to some age ν under the stage-1 distribution F_1, where ν solves

$$S_1(\nu;\theta_1) = S_0(\tau;\theta_0) \quad \Longleftrightarrow \quad F_1(\nu;\theta_1) = F_0(\tau;\theta_0),$$

i.e.

$$\nu = \nu(\tau;\theta_0,\theta_1) = Q_1(F_0(\tau;\theta_0);\ \theta_1)\,,$$

where $Q_1 = F_1^{-1}$ is the stage-1 quantile function. This is exactly `fam$quantile(fam$cdf(tau, theta0), theta1)` below: plug the stage-0 cdf at `tau` into the stage-1 quantile function. ν is the building block for the population functions `pslt()`/`dslt()`/ `qslt()` and for the log-likelihood `loglik_slt()` – see their documentation for how it is used.

Value

The equivalent age (a non-negative scalar).

See Also

`pslt()` for how `nu` combines with `tau1` into the population cdf; `loglik_slt()` for the resulting log-likelihood; `slt_tau()` for choosing `tau` as a stage-0 quantile.

Examples

```
slt_equiv_age("weibull", tau = 5, theta0 = c(0.1, 1.5), theta1 = c(0.3, 1.5))
```

---

slt_families *Available baseline lifetime distributions*

Description

Lists the two-parameter baseline lifetime distributions, theta = (lambda, alpha), that can be used as the `family` argument throughout the package. See `slt_dist()` for the pdf/cdf and literature source of each.

Usage

```
slt_families()
```

Value

A character vector of family keys.

See Also

`slt_dist()` for the mathematical definition and reference of each distribution; `slt_cases()` for the parameter-sharing cases these distributions are combined under.

Examples

```
slt_families()
```

slt_gof_ks *Kolmogorov-Smirnov goodness-of-fit test with parametric-bootstrap correction for a fitted two-stage life testing model*

Description

Performs the Kolmogorov-Smirnov goodness-of-fit test and reports both the naive p-value (for reference) and the standard parametric-bootstrap-corrected p-value.

Usage

```
slt_gof_ks(fit, B = 1000, n_cores = 1L)
```

Arguments

fit	A converged "slt_fit" object, fit to data from a simple-step-stress (rslt() with prop_transfer = 1) design.
B	Number of parametric bootstrap replicates.
n_cores	Number of cores for parallel::mclapply().

Details

The classical KS null distribution assumes a fully specified (not estimated) reference distribution; using it when the parameters were estimated from the same data makes the naive p-value anti-conservative (too small), a problem first noted by Lilliefors. The parametric bootstrap correction simulates B fresh data sets of the same size from the fitted model via rslt() (at the MLE), refits each with the same family/case, and takes the corrected p-value to be the proportion of bootstrap KS statistics at least as extreme as the one observed on the real data.

This test assumes every survivor was transferred at tau1 (the classical simple-step-stress design, prop_transfer = 1 in rslt()), because the single pooled reference distribution it uses (pslt()) is only the correct marginal CDF of the *combined* Y/Z sample in that case. Under genuine partial withdrawal, units that survive tau1 but are *not* transferred keep failing on stage 0 unmodified, which pslt() does not represent – pooling them into one KS statistic against pslt() would be a biased test. If fit$data came from a partial-withdrawal rslt() call (prop_transfer < 1 or design = "typeM"), this function is not appropriate; a Y-only and a Z-only KS test against their own (design-invariant) marginals would be the correct generalisation, but is not currently implemented.

Value

An object of class "slt_gof" with elements D (KS statistic), p_naive, p_bootstrap, B, B_converged.

Examples

```
set.seed(1)
d <- rslt(150, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fit <- fit_slt(d, "weibull", "shape1", tau1 = 5)
slt_gof_ks(fit, B = 100)
```

`slt_han_kundu_2014` *Han & Kundu (2014) simple step-stress data*

Description

A simple step-stress life-testing data set on the reliability of a solar lighting device, with 16 units observed under the baseline stress (normal operating temperature, 293K) and 15 under the elevated stress (353K) after a common change-point at `tau1 = 500` (hours).

Usage

```
slt_han_kundu_2014
```

Format

A list with elements:

y0 Numeric vector of 16 stage-0 (baseline stress) failure times.

y1 Numeric vector of 15 stage-1 (elevated stress) failure times.

tau1 The change-point time.

description, source Character strings.

Details

The original experiment placed `n = 35` units on test, but only these 31 failure times are reported in the source paper – the remaining 4 units did not fail before the experiment ended. This package's model has no right-censoring mechanism (see `?stagelife-package`), so those 4 units are not, and cannot be, represented here.

Source

Han, D. and Kundu, D. (2014). Inference for a step-stress model with competing risks for failure from the generalized exponential distribution under Type-I censoring. *IEEE Transactions on Reliability*, 64(1), 31-43.

See Also

slt_bobotas_kateri_2015, slt_wang_fei_2003, slt_zhu_2010 for the other three bundled data sets; `fit_slt()` for fitting a model to data in this format.

Examples

```
data(slt_han_kundu_2014)
fit <- fit_slt(list(Y = slt_han_kundu_2014$y0, Z = slt_han_kundu_2014$y1),
                  "weibull", "shape1", tau1 = slt_han_kundu_2014$tau1)
summary(fit)
```

slt_lrt *Likelihood-ratio test between nested parameter-sharing cases*

Description

Tests whether a reduced (nested) parameter-sharing case is adequate against a less-restricted alternative, via the likelihood-ratio statistic.

Usage

```
slt_lrt(fit_full, fit_reduced)
```

Arguments

`fit_full`, `fit_reduced`
: Converged `"slt_fit"` objects for the same `family`, `tau1` and `data`, with `fit_reduced` nested in `fit_full`.

Details

The only nested pairs among the three supported cases are `("shape1", "shape11")` and `("common", "shape11")` (both fix the remaining free shape parameter(s) to 1); see `slt_cases()`. This function tests `H0`: the reduced (`"shape11"`) model is adequate, against the corresponding less-restricted alternative.

Value

An object of class `"slt_lrt"` with `statistic`, `df`, `p.value`.

Examples

```
set.seed(1)
d <- rslt(200, "weibull", c(0.05, 1), c(0.2, 1), tau1 = 5, prop_transfer = 1)
f1 <- fit_slt(d, "weibull", "shape1", tau1 = 5)
f0 <- fit_slt(d, "weibull", "shape11", tau1 = 5)
slt_lrt(f1, f0)
```

slt_model_comparison *Compare several fitted models by information criteria*

Description

Builds a ranked comparison table across any number of fitted models, by information criteria.

Usage

```
slt_model_comparison(fits, sort_by = c("aic", "bic", "loglik"))
```

Arguments

fits	A (preferably named) list of "slt_fit" objects.
sort_by	One of "aic" (default), "bic", "loglik".

Details

By default sorted by AIC, across any number of "slt_fit" objects – typically different family/case combinations fit to the same data set – for exploratory model selection on real data. This is a lightweight diagnostic table for a single data set, not a Monte Carlo simulation study.

Value

A data.frame with columns model, family, case, n_par, logLik, AIC, BIC, converged.

Examples

```
set.seed(1)
d <- rslt(200, "weibull", c(0.05, 1.5), c(0.2, 1.5), tau1 = 5, prop_transfer = 1)
fits <- list(
  "Weibull / shape1"  = fit_slt(d, "weibull", "shape1", tau1 = 5),
  "Weibull / shape11" = fit_slt(d, "weibull", "shape11", tau1 = 5),
  "Weibull / common"  = fit_slt(d, "weibull", "common", tau1 = 5)
)
slt_model_comparison(fits)
```

---

slt_prob_stage1 *Probability of observing at least one stage-1 failure (Laumen & Cramer, Lemma 4.1)*

Description

Computes the probability of observing at least one stage-1 failure, P(R*_1 > 0), implementing Lemma 4.1 of Laumen & Cramer (2019).

Usage

```
slt_prob_stage1(
  n,
  tau1,
  theta0,
  family = "weibull",
  design = c("typeP", "typeM"),
  pi1 = NULL,
  R0_1 = NULL
)
```

Arguments

n	Number of units placed on test.
tau1	Stage-change time.
theta0	Numeric c(lambda0, alpha0), baseline-stage parameters.
family	A family key (see slt_families()) or family object.
design	"typeP" (fixed target proportion pi1) or "typeM" (fixed target count R0_1), matching rslt().
pi1	Target proportion transferred, for design = "typeP".
R0_1	Target number transferred, for design = "typeM".

Details

Because the maximum likelihood estimate of the elevated-stage parameter only exists when at least one unit is actually transferred to stage 1 and fails there (R*_1 > 0), this is a useful quantity for planning a stage life test: choosing n, tau1, and pi1/R0_1 so that slt_prob_stage1() is comfortably close to 1.
R*_1 = rho(D1) > 0 iff D1 <= n*, where n* = floor(n - 1/pi1) for the "typeP" design and n* = n - 1 for the "typeM" design (with R0_1 >= 1), and D1 ~ Binomial(n, F0(tau1)).

Value

A list with n_star (the threshold on D1) and prob (P(R*_1 > 0)).

Examples

```
# Reproduces Table 5 of Laumen & Cramer (2019) exactly (n = 12,
# theta0 = 1 i.e. lambda0 = 1, tau1 = 0.5):
slt_prob_stage1(12, 0.5, c(1, 1), design = "typeP", pi1 = 0.25)  # n*=8,  prob=0.9864756680
slt_prob_stage1(12, 0.5, c(1, 1), design = "typeM", R0_1 = 6)    # n*=11, prob=0.9999862301
```

---

slt_reduce_params *Reduce full* (theta0, theta1) *parameters to a case's free parameters*

Description

The inverse of slt_expand_params(); mostly useful for extracting the "true" parameter vector matching a given case in simulation studies. For "common", theta0[2] (rather than an average) is used as the shared shape and a warning is issued if theta0[2] != theta1[2].

Usage

```
slt_reduce_params(theta0, theta1, case = c("common", "shape1", "shape11"))
```

Arguments

theta0, theta1	Numeric c(lambda, alpha) vectors.
case	One of "common", "shape1", "shape11".

Value

A named numeric vector matching case_info(case)$par_names.

Examples

```
slt_reduce_params(c(0.2, 1.5), c(0.4, 1.5), case = "common")
```

---

slt_tau *p-th quantile of the baseline-stage distribution*

Description

Convenience wrapper typically used to set the change-point at a chosen quantile of the stage-0 lifetime distribution, e.g. tau <- slt_tau(family, theta0, p = 0.5) places the change-point at the stage-0 median.

Usage

```
slt_tau(family, theta0, p = 0.5)
```

Arguments

family	A family key (see slt_families()) or family object.
theta0	Numeric c(lambda0, alpha0), baseline-stage parameters.
p	Probability in (0, 1).

Value

The p-th quantile of the stage-0 distribution.

Examples

```
slt_tau("weibull", theta0 = c(0.1, 1.5), p = 0.5)
```

slt_wang_fei_2003 *Wang & Fei (2003) simple step-stress data*

Description

A simple step-stress life-testing data set on the reliability of electronic components under temperature stress, with 30 units observed under the baseline stress and 20 under the elevated stress after a common change-point at `tau1 = 910` (hours).

Usage

```
slt_wang_fei_2003
```

Format

A list with elements `y0`, `y1`, `tau1`, `description`, `source` (see slt_han_kundu_2014 for the structure).

Source

Wang, R. and Fei, H. (2003). Uniqueness of the maximum likelihood estimate of the Weibull distribution tampered failure rate model. *Communications in Statistics - Theory and Methods*, 32(12), 2321-2338.

See Also

slt_han_kundu_2014, slt_bobotas_kateri_2015, slt_zhu_2010.

Examples

```
data(slt_wang_fei_2003)
str(slt_wang_fei_2003)
```

slt_zhu_2010 *Zhu (2010) simple step-stress data*

Description

A simple step-stress life-testing data set on light-bulb-filament fatigue (two sets of 32 bulbs pooled), with 33 units observed under the baseline stress and 20 under the elevated stress after a common change-point at `tau1 = 94`.

Usage

```
slt_zhu_2010
```

Format

A list with elements `y0`, `y1`, `tau1`, `description`, `source` (see slt_han_kundu_2014 for the structure).

Details

The original experiment used two sets of 32 bulbs (`n = 64` total), but only these 53 failure times are bundled here – the remaining 11 units did not fail before their set's test ended. This package's model has no right-censoring mechanism (see `?stagelife-package`), so those 11 units are not, and cannot be, represented here.

The source describes the nominal change-point as 96 hours, but this is inconsistent with the recorded data itself: the smallest stage-1 (post-change) observation is 94.38 (Set-II), which would be impossible if the change-point were 96 (a unit cannot fail on stage 1 before the stage change happens). `tau1 = 94` – the only value consistent with both stages of the recorded data (every stage-0 observation is `<= 91.56`; every stage-1 observation is `>= 94.38`) – is used here instead.

Source

Zhu, Y. (2010). *Optimal Design and Equivalency of Accelerated Life Testing Plans*. PhD dissertation, Rutgers, The State University of New Jersey, School of Graduate Studies.

See Also

slt_han_kundu_2014, slt_bobotas_kateri_2015, slt_wang_fei_2003.

Examples

```
data(slt_zhu_2010)
str(slt_zhu_2010)
```

`summary.slt_fit` *Summarise a fitted two-stage life testing model*

Description

Summarise a fitted two-stage life testing model

Usage

```
## S3 method for class 'slt_fit'
summary(object, level = 0.95, ...)
```

Arguments

`object`	An `"slt_fit"` object from `fit_slt()`.
`level`	Confidence level for the Wald (log-normal) confidence limits reported in the coefficient table.
`...`	Currently unused.

Value

An object of class `"summary.slt_fit"`.

`vcov.slt_fit` *Variance-covariance matrix of a fitted two-stage life testing model's coefficients*

Description

Computes the natural-scale variance-covariance matrix by the delta method, from the log-scale one fitted internally.

Usage

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

Arguments

`object`	An `"slt_fit"` object from `fit_slt()`.
`...`	Currently unused.

Details

`object$vcov_log` (the inverse of the numerical Hessian of the negative log-likelihood in `log(par)`) is transformed to the natural scale by the delta method: with $\eta = \log(\theta)$, $d\theta/d\eta = \mathrm{diag}(\theta)$, so $\mathrm{Var}(\theta) \approx \mathrm{diag}(\theta)\,\mathrm{Var}(\eta)\,\mathrm{diag}(\theta)$.

Value

The natural-scale variance-covariance matrix, or `NULL` if the Hessian was not invertible at the optimum.

See Also

`confint.slt_fit()`, which uses this (or the log scale directly) to build a confidence interval.

Index

∗ **datasets**
slt_bobotas_kateri_2015, 14
slt_han_kundu_2014, 24
slt_wang_fei_2003, 29
slt_zhu_2010, 29

coef.slt_fit, 3
confint.slt_fit, 3
confint.slt_fit(), *31*

dslt *(*pslt*)*, 11
dslt(), *21, 22*

fit_slt, 4
fit_slt(), *3, 4*, *6*, *10*, *12*, *17*, *24*, *30, 31*

lamW::lambertWm1(), *18, 19*
logLik.slt_fit, 5
loglik_slt, 6
loglik_slt(), *12*, *17*, *22*

neg_loglik_slt(), *17*

parallel::mclapply(), *15*, *23*
plot.slt_fit, 7
plot_cdf, 7
plot_cdf(), *7*
plot_pp, 8
plot_qq, 9
plot_survival, 9
print.slt_fit, 10
print.summary.slt_fit, 10
pslt, 11
pslt(), *7, 8*, *21–23*

qslt *(*pslt*)*, 11
qslt(), *21, 22*

rslt, 12
rslt(), *5–9*, *11, 12*, *15*, *21*, *23*, *27*

slt_bobotas_kateri_2015, 14, *24*, *29, 30*
slt_bootstrap_ci, 14
slt_cases, 16
slt_cases(), *3*, *5*, *22*, *25*
slt_dist, 17
slt_dist(), *22*
slt_equiv_age, 21
slt_equiv_age(), *11, 12*, *21*
slt_expand_params(), *17*, *27*
slt_families, 22
slt_families(), *5, 6*, *11–13*, *18*, *21*, *27, 28*
slt_gof_ks, 23
slt_gof_ks(), *8*, *15*
slt_han_kundu_2014, *14*, 24, *29, 30*
slt_lrt, 25
slt_lrt(), *16, 17*
slt_model_comparison, 25
slt_model_comparison(), *17*
slt_prob_stage1, 26
slt_reduce_params, 27
slt_reduce_params(), *17*
slt_tau, 28
slt_tau(), *22*
slt_wang_fei_2003, *14*, *24*, 29, *30*
slt_zhu_2010, *14*, *24*, 29, *29*
stats::AIC(), *6*
stats::BIC(), *6*
stats::dgamma(), *19*
stats::dnorm(), *19*
stats::optim(), *5*
stats::pgamma(), *19*
stats::plnorm(), *19*
stats::pnorm(), *19*
stats::qgamma(), *19*
stats::qnorm(), *19*
summary.slt_fit, 30
summary.slt_fit(), *10, 11*

vcov.slt_fit, 31