Package 'BetaDanish'

July 31, 2026

Type Package

Title The Beta-Danish Distribution for Lifetime Data Analysis

Version 0.3.0

Description Implements the four-parameter Beta-Danish distribution and its three-parameter Exponentiated Danish submodel for survival, reliability and lifetime data analysis, following Ahmad and Danish (2025) <doi:10.2478/jamsi-2025-0010>. Density, distribution, quantile, survival, hazard and random generation functions are evaluated so as to retain accuracy in the heavy upper tail, where the survival function is regularly varying. Estimation covers maximum likelihood for complete and right-censored samples, ridge-penalized fitting for weakly identified regimes, a grouped likelihood for times recorded on a coarse grid, and Bayesian sampling. Inference provides log-scale Wald and profile likelihood intervals, together with a reparameterization in terms of the identified composite of the two shape parameters. Structural properties include raw, incomplete and conditional moments with their existence conditions, Shannon, Renyi and Tsallis entropies, mean residual life, mean deviations, Lorenz and Bonferroni curves, probability weighted moments, order statistics, stress-strength reliability, hazard shape classification and the tail index. Regression modules cover accelerated failure time models, mixture and promotion-time cure models, and competing risks with Aalen-Johansen comparison and Gray's test. Analyses can be run directly from a delimited text file or spreadsheet.

URL https://bilal-aiou.github.io/BetaDanish/, https://github.com/bilal-aiou/BetaDanish

BugReports https://github.com/bilal-aiou/BetaDanish/issues

License GPL-3

Encoding UTF-8

LazyData true

Imports grDevices, maxLik, survival, stats, graphics, utils, tools

Suggests readxl, flexsurv, testthat (>= 3.0.0), knitr, rmarkdown, cmprsk, coda, MCMCpack

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

Config/roxygen2/version 8.0.0

Language en-US

NeedsCompilation no

Author Bilal Ahmad [aut, cre],
Muhammad Yameen Danish [aut]

Maintainer Bilal Ahmad <bilalahmad.imcbh9@gmail.com>

Repository CRAN

Date/Publication 2026-07-31 15:20:40 UTC

Contents

`aarset` *Aarset Device Failure Times*

Description

Times to failure of 50 devices, exhibiting a classic bathtub-shaped hazard rate. This is a standard benchmark dataset in reliability engineering.

Usage

```
aarset
```

Format

A data frame with 50 rows and 2 columns:

time Failure time

status Event indicator (1 = event occurred)

Source

Aarset, M. V. (1987). How to Identify a Bathtub Hazard Rate. IEEE Transactions on Reliability, R-36(1), 106-108.

Examples

```
data(aarset)

fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = aarset)
plot(fit, type = "hazard")
```

---

analyze_betadanish *Comprehensive Beta-Danish Analysis Pipeline*

Description

Runs a complete end-to-end analysis: reads data, fits the 4-parameter and 3-parameter models, compares them, benchmarks against standard distributions, and generates diagnostic plots.

Usage

```
analyze_betadanish(file, time_col, status_col = NULL)
```

Arguments

`file`	Path to the CSV or Excel file containing the data.
`time_col`	Name of the time column.
`status_col`	Name of the status column (optional).

Value

Invisibly returns a list containing the fitted full model and submodel objects. The function is mainly called for its side effects of printing an analysis report and producing diagnostic plots.

`bayes_betadanish` *Bayesian Estimation for the Beta-Danish Distribution*

Description

Samples from the posterior of the Beta-Danish or Exponentiated Danish parameters using a random-walk Metropolis sampler with vague $\Gamma(0.01, 0.01)$ priors on the positive parameters.

Usage

```
bayes_betadanish(
  time,
  status = NULL,
  submodel = TRUE,
  burnin = 5000,
  mcmc = 15000,
  tune = 0.5,
  theta_init = NULL,
  seed = NULL,
  verbose = 0
)
```

Arguments

`time`	Numeric vector of observed times.
`status`	Numeric vector of event indicators (1 = event, 0 = right-censored).
`submodel`	Logical; `TRUE` for the 3-parameter ED submodel, `FALSE` for the 4-parameter full model.
`burnin`	Burn-in iterations.
`mcmc`	Post-burnin iterations.
`tune`	Random-walk tuning parameter.
`theta_init`	Optional starting values on the log scale.
`seed`	Optional integer seed.
`verbose`	Integer; passed to MCMCmetrop1R (0 = silent).

Details

Requires **MCMCpack** and **coda** (Suggests).

Value

An object of class `"bd_bayes"` with components `draws` (mcmc object), `summary`, `HPD`, `submodel`, `call`.

Examples

```
## Not run:
set.seed(1)
dat <- rbetadanish(100, a = 1.5, b = 2, c = 3, k = 0.5)
fit <- bayes_betadanish(time = dat, submodel = TRUE,
                        burnin = 500, mcmc = 1500)
fit$summary

## End(Not run)
```

---

bd_analyze_csv *Analyse Survival Data Straight from a CSV File*

Description

A single entry point that reads a delimited or Excel file, fits the requested Beta-Danish model or models, assembles the results into tidy tables, and optionally writes those tables and the diagnostic figures to a directory.

Usage

```
bd_analyze_csv(
  file,
  time_col = NULL,
  status_col = NULL,
  covariates = NULL,
  cause_col = NULL,
  analysis = c("univariate", "aft", "cure", "competing"),
  model = c("both", "BD", "ED"),
  compare = TRUE,
  bayes = FALSE,
  cure_formula = NULL,
  cure_type = c("mixture", "promotion"),
  output_dir = NULL,
  n_starts = 10,
  seed = NULL,
  quiet = FALSE
)

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

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

## S3 method for class 'summary.bd_analysis'
```

```
print(x, ...)

## S3 method for class 'bd_analysis'
plot(x, type = "survival", ...)
```

Arguments

`file`	Path to a ‘.csv‘, ‘.txt‘, ‘.tsv‘, ‘.xls‘ or ‘.xlsx‘ file.
`time_col`, `status_col`, `cause_col`	Column names, passed to [read_survival_data()]. ‘NULL‘ means guess.
`covariates`	Covariate columns to use. ‘NULL‘ means none for ‘analysis = "univariate"‘, and every non-response column for the regression analyses.
`analysis`	Which analysis to run: ‘"univariate"‘ (default), ‘"aft"‘, ‘"cure"‘ or ‘"competing"‘.
`model`	For ‘analysis = "univariate"‘, whether to fit the four-parameter Beta-Danish (‘"BD"‘), the three-parameter Exponentiated Danish submodel (‘"ED"‘), or ‘"both"‘ (default) and compare them.
`compare`	Logical; benchmark against standard lifetime distributions via [compare_distributions()]. Requires the ‘flexsurv‘ package; skipped with a recorded note if it is absent. Default ‘TRUE‘.
`bayes`	Logical; also run [bayes_betadanish()]. Requires ‘MCMCpack‘. Default ‘FALSE‘, since it is slow.
`cure_formula`	Right-hand-side formula for the cure fraction when ‘analysis = "cure"‘, for example ‘~ group‘. Defaults to the retained covariates.
`cure_type`	‘"mixture"‘ (default) or ‘"promotion"‘.
`output_dir`	Directory to write results to. ‘NULL‘ (default) writes nothing. The directory is created if it does not exist.
`n_starts`	Number of random starting points for each fit.
`seed`	Optional integer seed, for reproducible multi-start fitting.
`quiet`	Logical; suppress progress messages.
`x`	A ‘"bd_analysis"‘ object.
`...`	Ignored.
`object`	A ‘"bd_analysis"‘ object.
`type`	Plot type passed to [plot.betadanish()].

Details

Nothing is written to disk unless ‘output_dir‘ is supplied. When it is, the function writes one CSV per result table and a PNG per diagnostic figure, and records the paths in the ‘files‘ component.

Each fit is attempted independently. If one fails, the message is recorded in ‘failures‘ and the remaining analyses still run, so a difficult four-parameter fit does not cost you the submodel results alongside it.

Warnings raised during fitting are captured in ‘warnings‘ rather than printed as they occur. This is where the identifiability diagnostics appear, so it is worth reading: a four-parameter fit that converges cleanly but sits on the ‘b = 1‘ ridge will say so there.

For ‘analysis = "univariate"‘ with ‘model = "both"‘, a likelihood ratio test of the submodel is included. Read it alongside those diagnostics: see the Identifiability section of [fit_betadanish()].

Value

An object of class ‘"bd_analysis"‘: a list with ‘call‘, ‘analysis‘, ‘data‘, ‘data_report‘, ‘fits‘, ‘tables‘, ‘extras‘, ‘warnings‘, ‘failures‘, ‘output_dir‘ and ‘files‘. It has ‘print‘, ‘summary‘ and ‘plot‘ methods.

See Also

[read_survival_data()], [bd_csv_template()], [fit_betadanish()], [fit_bd_aft()], [fit_bd_cure()], [fit_bd_competing()]

Examples

```
f <- system.file("extdata", "censored_sample.csv", package = "BetaDanish")

# Submodel only, no benchmarking: fast enough for an example
res <- bd_analyze_csv(f, analysis = "univariate", model = "ED",
                      compare = FALSE, n_starts = 3, seed = 1, quiet = TRUE)
res
res$tables$information_criteria


# Both models, with tables and figures written to a temporary directory
out <- file.path(tempdir(), "bd_results")
res2 <- bd_analyze_csv(f, model = "both", compare = FALSE,
                       output_dir = out, n_starts = 5, seed = 1)
basename(res2$files)

# AFT regression, covariates taken from the file
g <- system.file("extdata", "covariate_sample.csv", package = "BetaDanish")
res3 <- bd_analyze_csv(g, analysis = "aft", covariates = c("age", "thickness"),
                       n_starts = 5, seed = 1)
summary(res3)
```

`bd_conditional_moment` *Conditional Moments*

Description

$E(Z^r \mid Z > t)$ or $E(Z^r \mid Z \leq t)$.

Usage

```
bd_conditional_moment(r, t, a, b, c, k, upper = TRUE, rel.tol = 1e-10)
```

Arguments

r	Order of the moment.
t	Vector of thresholds.
a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
upper	Logical; 'TRUE' (default) conditions on $Z > t$.
rel.tol	Relative accuracy, passed to [stats::integrate()].

Value

A numeric vector the length of 't'.

Examples

```
bd_conditional_moment(1, t = c(0.5, 1, 2), a = 1.5, b = 3, c = 2, k = 1)
```

---

bd_csv_template *Write a Skeleton CSV for BetaDanish*

Description

Writes a small, correctly shaped CSV showing the layout [bd_analyze_csv()] and [read_survival_data()] expect. Fill it in with your own data, or use it as a reference for renaming the columns of an existing file.

Usage

```
bd_csv_template(
  path,
  type = c("univariate", "complete", "covariate", "competing"),
  n = 10,
  overwrite = FALSE
)
```

Arguments

path	Destination file path. Use [tempfile()] if you only want to inspect the result.
type	Which layout to write: '"univariate"' gives 'time' and 'status'; '"complete"' gives 'time' alone, for an uncensored sample; '"covariate"' adds two example covariate columns; '"competing"' gives 'time' and 'cause'.
n	Number of illustrative rows. Default 10.
overwrite	Logical; overwrite an existing file. Default 'FALSE'.

Details

The illustrative values are plausible but arbitrary. ‘status‘ is coded 1 = event observed, 0 = right-censored. For ‘"competing"‘, ‘cause‘ is coded 0 = censored and 1, 2, ... for the competing causes.

Value

The path, invisibly.

See Also

[bd_analyze_csv()], [read_survival_data()]

Examples

```
p <- bd_csv_template(tempfile(fileext = ".csv"), type = "covariate", n = 5)
read.csv(p)
unlink(p)
```

`bd_entropy` *Entropies of the Beta-Danish Distribution*

Description

Shannon, Renyi and Tsallis entropies. All three share one documentation topic so that every argument is described exactly once.

Usage

```
bd_entropy_shannon(
  a,
  b,
  c,
  k,
  terms = 20000L,
  method = c("closed", "quadrature"),
  rel.tol = 1e-10,
  subdivisions = 4000L
)

bd_entropy_renyi(a, b, c, k, order = 2, rel.tol = 1e-10, subdivisions = 4000L)

bd_entropy_tsallis(
  a,
  b,
  c,
  k,
  order = 2,
```

```
  rel.tol = 1e-10,
  subdivisions = 4000L
)
```

Arguments

a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
terms	Number of series terms before the analytic tail is applied (Shannon only).
method	'"closed"' (default) for the closed form, or '"quadrature"' for direct numerical integration of $-f \log f$, which is slower but independent of the series (Shannon only).
rel.tol	Relative accuracy, passed to [stats::integrate()].
subdivisions	Subdivision limit, passed to [stats::integrate()].
order	Entropy order q, positive and not equal to one (Renyi and Tsallis only).

Details

Writing $u = G(Z) \sim \mathrm{Beta}(a, b)$ and $v = u^{1/c}$, the Shannon entropy has the closed form

$$H = \log B(a,b) - \log(ck) - (a-1/c)\{\psi(a) - \psi(a+b)\} - (b-1)\{\psi(b) - \psi(a+b)\} + 2\sum_{i \geq 1} \frac{B(a + i/c,\, b)}{i\, B(a,b)},$$

the final sum arising from $-2E\log(1-v)$ expanded as a power series.

With $I_q = \int_0^\infty f(z)^q dz$, the Renyi entropy is $\log(I_q)/(1-q)$ and the Tsallis entropy is $(1-I_q)/(q-1)$. Both reduce to the Shannon entropy as $q \to 1$, which is excluded; use 'bd_entropy_shannon()' at $q = 1$.

I_q is evaluated as $\int_0^1 f(z(u))^{q-1} d\mathrm{Beta}(u; a, b)$ on the finite Beta scale rather than over the half line, which keeps the heavy upper tail from dominating the quadrature.

Value

A single number. The Shannon entropy is in nats.

Series truncation

The Shannon series terms decay like $i^{-(b+1)}$, so truncating at M omits a tail of order M^{-b}. That is negligible for large b and is not for small b: against high-precision integration the plain truncated sum at $M = 2000$ is out by about 10^{-8} at $b = 3$ but by about 0.11 at $b = 0.5$. The analytic tail $\Gamma(b) c^b M^{-b} / \{bB(a,b)\}$ is therefore added, restoring agreement to roughly eight digits across the range.

Examples

```
bd_entropy_shannon(a = 1.5, b = 3, c = 2, k = 1)

# Independent check by quadrature
bd_entropy_shannon(a = 1.5, b = 3, c = 2, k = 1, method = "quadrature")

bd_entropy_renyi(a = 1.5, b = 3, c = 2, k = 1, order = 2)
bd_entropy_tsallis(a = 1.5, b = 3, c = 2, k = 1, order = 2)
```

---

bd_hazard_shape *Hazard Rate Shape and the Glaser Diagnostic*

Description

Classifies the hazard rate as increasing, decreasing, bathtub-shaped or upside-down bathtub-shaped, and returns the mode of the density.

Usage

```
bd_hazard_shape(a, b, c, k, n_grid = 400L, range_p = c(1e-04, 1 - 1e-04))

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

Arguments

a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
n_grid	Number of grid points between the lower and upper evaluation quantiles.
range_p	Two probabilities bounding the evaluation range.
x	A ‘"bd_shape"‘ object.
...	Ignored.

Details

Glaser's criterion works through $\eta(t) = -f'(t)/f(t)$: monotone increasing η gives an increasing hazard, monotone decreasing gives a decreasing hazard, and a single interior turning point gives a bathtub or upside-down bathtub. Here η is formed analytically from the log-density and evaluated on a grid, and the hazard itself is checked alongside it, so the two must agree before a shape is reported.

Value

A list of class ‘"bd_shape"‘ with the classification, the grid of times, the hazard, Glaser's $\eta(t) = -f'(t)/f(t)$, and the mode.

Examples

```
s <- bd_hazard_shape(a = 1.5, b = 3, c = 2, k = 1)
s$shape
s$mode
```

`bd_identified_coef` *Coefficients in the Identified Parametrisation*

Description

Reports the four-parameter fit in terms of the composite ac, which is identified, rather than a and c separately, which are not.

Usage

```
bd_identified_coef(object, level = 0.95)

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

Arguments

`object`	A fitted ‘"betadanish"‘ object.
`level`	Confidence level for the reported intervals. Default 0.95.
`x`	A ‘"bd_identified"‘ object.
`...`	Ignored.

Details

Near the lower tail a and c enter the density almost entirely through their product, so the expected information is close to singular in the direction that separates them. Writing $p = ac$ and $r = a/c$, the ratio r carries almost no information while p is estimated precisely.

Reporting (ac, b, k) is therefore the honest summary of a four-parameter fit that sits on the flat direction: it has finite standard errors and a far better conditioned covariance matrix, and it says exactly what the data determine. The thesis pipeline records a condition number falling from 2750 to 152 on the remission data, with the log-likelihood unchanged, since this is a reparametrisation and not a different model.

For the ED submodel $a = 1$, so $ac = c$ and nothing is gained; the function returns the coefficients unchanged with a note.

Value

An object of class ‘"bd_identified"‘ holding the reparametrised estimates with delta-method standard errors, and the condition numbers of the covariance matrix before and after.

See Also

[fit_betadanish()] and its Identifiability section, [bd_profile_ci()]

Examples

```
dat <- simulate_bd_data(200, a = 1.5, b = 3, c = 2, k = 0.5, seed = 4)
fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                      n_starts = 5)
bd_identified_coef(fit)
```

`bd_incomplete_moment` *Incomplete Moments*

Description

The lower incomplete moment $M_r(t) = \int_0^t z^r f(z)\,dz$ and the upper incomplete moment $\Upsilon_r(t) = \int_t^\infty z^r f(z)\,dz$.

Usage

```
bd_incomplete_moment(r, t, a, b, c, k, lower = TRUE, rel.tol = 1e-10)
```

Arguments

`r`	Order of the moment.
`t`	Vector of thresholds.
`a`	Shape parameter (beta generator).
`b`	Shape parameter governing the tail.
`c`	Shape parameter (baseline).
`k`	Scale parameter (baseline).
`lower`	Logical; 'TRUE' (default) for $M_r(t)$, 'FALSE' for $\Upsilon_r(t)$.
`rel.tol`	Passed to [stats::integrate()].

Details

The threshold is mapped to the Beta scale by $u_t = G(t)$, computed on the log scale so it stays accurate when $G(t)$ approaches one, and the integral is taken over $(0, u_t)$ or $(u_t, 1)$.

Value

A numeric vector the length of 't'.

See Also

[bd_mrl()], [bd_lorenz()], [bd_mean_deviation()]

Examples

```
a <- 1.5; b <- 3; c <- 2; k <- 1
t <- 1
# The two halves sum to the complete moment
bd_incomplete_moment(1, t, a, b, c, k, lower = TRUE) +
  bd_incomplete_moment(1, t, a, b, c, k, lower = FALSE)
bd_moments(1, a, b, c, k)
```

---

bd_lorenz *Lorenz and Bonferroni Curves*

Description

Lorenz and Bonferroni Curves

Usage

```
bd_lorenz(p, a, b, c, k, rel.tol = 1e-10)

bd_bonferroni(p, a, b, c, k, rel.tol = 1e-10)
```

Arguments

p	Vector of probabilities in $[0, 1]$.
a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
rel.tol	Passed to [stats::integrate()].

Details

$L(p) = M_1(Q(p))/\mu$ and $B(p) = L(p)/p$, with Q the quantile function. Both require $b > 1$.

Value

A numeric vector the length of ‘p‘.

Examples

```
p <- c(0.1, 0.25, 0.5, 0.75, 0.9)
bd_lorenz(p, a = 1.5, b = 3, c = 2, k = 1)
bd_bonferroni(p, a = 1.5, b = 3, c = 2, k = 1)
```

bd_mean_deviation *Mean Deviations*

Description

Mean deviation about the mean, $E|Z - \mu|$, or about the median.

Usage

```
bd_mean_deviation(a, b, c, k, about = c("mean", "median"), rel.tol = 1e-10)
```

Arguments

a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
about	‘"mean"‘ (default) or ‘"median"‘.
rel.tol	Passed to [stats::integrate()].

Details

$E|Z - \mu| = 2\mu F(\mu) - 2M_1(\mu)$ and $E|Z - M| = \mu - 2M_1(M)$, with M the median.

Value

A single number, or ‘NA‘ when $b \leq 1$.

Examples

```
bd_mean_deviation(a = 1.5, b = 3, c = 2, k = 1)
bd_mean_deviation(a = 1.5, b = 3, c = 2, k = 1, about = "median")
```

bd_moments *Raw Moments of the Beta-Danish Distribution*

Description

Raw Moments of the Beta-Danish Distribution

Usage

```
bd_moments(r, a, b, c, k, rel.tol = 1e-10, subdivisions = 4000L)
```

Arguments

r	Order or orders of the moment. Non-negative.
a	Shape parameter (beta generator).
b	Shape parameter governing the tail; 'E(Z^r)' is finite iff 'b > r'.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
rel.tol	Relative accuracy, passed to [stats::integrate()].
subdivisions	Subdivision limit, passed to [stats::integrate()].

Details

The survival function is regularly varying with index $-b$, so $E(Z^r)$ is finite if and only if $b > r$. That condition is checked rather than assumed: a request for a moment that does not exist returns 'Inf', not a large finite number produced by a truncated sum.

Value

A numeric vector the length of 'r'. Entries with 'b <= r' are 'Inf'.

See Also

[bd_moment_summary()], [bd_incomplete_moment()]

Examples

```
bd_moments(1:2, a = 1.5, b = 3, c = 2, k = 1)

# The fourth moment does not exist when b = 3
bd_moments(4, a = 1.5, b = 3, c = 2, k = 1)
```

---

bd_moment_summary *Summary Moments: Mean, Variance, Skewness and Kurtosis*

Description

Summary Moments: Mean, Variance, Skewness and Kurtosis

Usage

```
bd_moment_summary(a, b, c, k, rel.tol = 1e-10, subdivisions = 4000L)
```

Arguments

a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
rel.tol	Relative accuracy, passed to [stats::integrate()].
subdivisions	Subdivision limit, passed to [stats::integrate()].

Details

Skewness needs $b > 3$ and kurtosis $b > 4$. Both are frequently unavailable for fitted values of b, which is a property of the family rather than a limitation of the computation.

Value

A named numeric vector. Entries requiring a moment that does not exist are 'NA', with the governing condition given in the 'condition' attribute.

Examples

```
bd_moment_summary(a = 1.5, b = 5, c = 2, k = 1)
bd_moment_summary(a = 1.5, b = 2.5, c = 2, k = 1)   # skew/kurt unavailable
```

---

bd_mrl *Mean Residual Life and Reversed Mean Residual Life*

Description

Mean Residual Life and Reversed Mean Residual Life

Usage

```
bd_mrl(t, a, b, c, k, rel.tol = 1e-10)

bd_rmrl(t, a, b, c, k, rel.tol = 1e-10)
```

Arguments

t	Vector of times.
a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
rel.tol	Passed to [stats::integrate()].

Details

Following the definitions in the underlying dissertation, with Υ_1 the upper and M_1 the lower incomplete first moment,

$$m(t) = E(Z - t \mid Z > t) = \Upsilon_1(t)/S(t) - t,$$

$$\bar{m}(t) = E(t - Z \mid Z \leq t) = t - M_1(t)/F(t).$$

The reversed form is also called the mean inactivity time. Both require the first moment to exist, hence $b > 1$.

No monotonicity is asserted. Ageing classification requires a separate argument and is not implied by these values; see [bd_hazard_shape()] for the hazard-rate counterpart.

Value

A numeric vector the length of 't', or all 'NA' when $b \leq 1$.

Examples

```
bd_mrl(c(0.5, 1, 2, 5), a = 1.5, b = 3, c = 2, k = 1)
bd_rmrl(c(0.5, 1, 2, 5), a = 1.5, b = 3, c = 2, k = 1)
```

`bd_order_stat_cdf` *Order Statistic Distribution and Moments*

Description

Order Statistic Distribution and Moments

Usage

```
bd_order_stat_cdf(t, i, n, a, b, c, k)

bd_order_stat_moments(r, i, n, a, b, c, k, rel.tol = 1e-10)
```

Arguments

`t`	Vector of times, for 'bd_order_stat_cdf'.
`i`	Index of the order statistic, from 1 to 'n'.
`n`	Sample size.
`a`	Shape parameter (beta generator).
`b`	Shape parameter governing the tail.
`c`	Shape parameter (baseline).
`k`	Scale parameter (baseline).
`r`	Order of the moment, for 'bd_order_stat_moments'.
`rel.tol`	Passed to [stats::integrate()].

Details

$F_{i:n}(t) = I_{F(t)}(i, n - i + 1)$, and moments are taken against the order-statistic density on the Beta scale. The i-th order statistic of a sample of size n has a finite r-th moment when $b(n - i + 1) > r$, so extremes need heavier restrictions than the parent distribution, and the maximum needs $b > r$ exactly as the parent does.

Value

'bd_order_stat_cdf' returns a vector the length of 't'; 'bd_order_stat_moments' returns a single number.

See Also

[bd_order_stat_pdf()]

Examples

```
bd_order_stat_cdf(c(0.5, 1, 2), i = 3, n = 5, a = 1.5, b = 3, c = 2, k = 1)
bd_order_stat_moments(1, i = 1, n = 5, a = 1.5, b = 3, c = 2, k = 1)
```

---

bd_order_stat_pdf *Density of the r-th Order Statistic*

Description

Evaluates the probability density function of the r-th order statistic from a sample of size n drawn from the Beta-Danish distribution.

Usage

```
bd_order_stat_pdf(x, r, n, a, b, c, k, log = FALSE)
```

Arguments

x	Numeric vector of time points.
r	Integer order (1 = minimum, n = maximum).
n	Integer sample size.
a, b, c, k	Positive parameters of the Beta-Danish distribution.
log	Logical; if TRUE return the log-density.

Value

Numeric vector (or its log).

Examples

```
tgrid <- seq(0.01, 5, length.out = 50)
bd_order_stat_pdf(tgrid, r = 5, n = 20,
                  a = 1.5, b = 2.5, c = 2, k = 1)
```

`bd_profile_ci` *Profile Likelihood Confidence Interval*

Description

Profiles one parameter, maximising over the others, and inverts the likelihood ratio test to obtain an interval.

Usage

```
bd_profile_ci(
  object,
  parameter = "b",
  level = 0.95,
  grid = NULL,
  n_grid = 60L,
  method = "Nelder-Mead"
)

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

Arguments

`object`	A fitted ‘"betadanish"‘ object.
`parameter`	Name of the parameter to profile, one of ‘"a"‘, ‘"b"‘, ‘"c"‘ or ‘"k"‘. Default ‘"b"‘.
`level`	Confidence level. Default 0.95.
`grid`	Optional numeric vector of values to scan. If ‘NULL‘ (default) a log-spaced grid is built around the estimate.
`n_grid`	Number of grid points when ‘grid‘ is ‘NULL‘.
`method`	Optimisation method for the nuisance parameters, passed to [stats::optim()].
`x`	A ‘"bd_profile"‘ object.
`...`	Ignored.

Details

The interval is the set of values θ_0 for which $2\{\ell_{\max} - \ell_p(\theta_0)\} \leq \chi^2_{1,\,\mathrm{level}}$, with ℓ_p the profile log-likelihood.

An open upper bound is a result, not a failure. For a weakly identified tail index the profile can stay inside the critical region all the way to the top of the grid, in which case ‘upper‘ is returned as ‘Inf‘ and the honest report is a lower bound rather than a point estimate with an interval. The thesis pipeline records precisely this on the breaking-stress data, where the profile gives $b \geq 26.1$ with no finite upper bound.

Widening ‘grid‘ will not manufacture a bound; it only confirms the flatness.

Value

An object of class ‘"bd_profile"‘ with the interval, the grid, the profile log-likelihood, and the critical value.

See Also

[bd_wald_ci()], [bd_identified_coef()]

Examples

```
dat <- simulate_bd_data(200, a = 1, b = 3, c = 2, k = 0.5, seed = 2)
fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                      submodel = TRUE, n_starts = 3)
p <- bd_profile_ci(fit, "b")
p
```

---

bd_profile_plot *Plot a Profile Likelihood*

Description

Draws the profile log-likelihood produced by [bd_profile_ci()], with the critical threshold and the resulting interval marked.

Usage

```
bd_profile_plot(
  x,
  col = "steelblue",
  lwd = 2,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

Arguments

x	An object of class '"bd_profile"'.
col, lwd	Colour and line width for the profile curve.
main, xlab, ylab	Labels. 'NULL' uses sensible defaults.
...	Further graphical parameters.

Details

The horizontal line sits at $\ell_{\max} - \chi^2_{1,\alpha}/2$. Every parameter value whose profile lies above it is inside the interval.

When the curve does not fall back below that line before the right-hand edge of the grid, there is no finite upper bound and the plot says so. Widening the grid will not produce one; it will only confirm the flatness. That is the situation the underlying dissertation records for the tail index on the breaking-stress data.

Value

Invisibly returns 'x'.

See Also

[bd_profile_ci()]

Examples

```
dat <- simulate_bd_data(120, a = 1, b = 3, c = 2, k = 0.5, seed = 3)
fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                      submodel = TRUE, n_starts = 1)
p <- bd_profile_ci(fit, "b", n_grid = 15L)
bd_profile_plot(p)
```

---

bd_pwm *Probability Weighted Moments*

Description

$\tau_{r,s,w} = E\{Z^r F(Z)^s (1 - F(Z))^w\}$.

Usage

```
bd_pwm(r, s = 0, w = 0, a, b, c, k, rel.tol = 1e-10)
```

Arguments

r	Power on Z.
s	Power on $F(Z)$.
w	Power on $1 - F(Z)$.
a	Shape parameter (beta generator).
b	Shape parameter governing the tail.
c	Shape parameter (baseline).
k	Scale parameter (baseline).
rel.tol	Passed to [stats::integrate()].

Details

Existence follows the same rule as the raw moments in the worst case, $b > r$, and the weights can only reduce the tail contribution.

Value

A single number.

Examples

```
# tau_{1,0,0} is the mean
bd_pwm(1, 0, 0, a = 1.5, b = 3, c = 2, k = 1)
bd_moments(1, a = 1.5, b = 3, c = 2, k = 1)

bd_pwm(1, 1, 0, a = 1.5, b = 3, c = 2, k = 1)
```

---

bd_simulation_competing

Finite-Sample Simulation Study for Competing Risks

Description

Finite-Sample Simulation Study for Competing Risks

Usage

```
bd_simulation_competing(
  n = c(200, 400),
  n_sim = 100,
  pars = list(c(b = 2.5, c = 1.8, k = 0.04), c(b = 3.5, c = 1.2, k = 0.02)),
  gammas = NULL,
  censor_rate = 0.15,
  submodel = TRUE,
  n_starts = 2,
```

```
  level = 0.95,
  seed = NULL,
  quiet = FALSE
)
```

Arguments

n	Vector of sample sizes.
n_sim	Number of replicates at each sample size.
pars	List of true parameter vectors, one per cause.
gammas	Optional accelerated failure time coefficients, one per cause.
censor_rate	Approximate proportion of right-censored observations.
submodel	Logical; fit Exponentiated Danish cause-specific kernels.
n_starts	Starting points per fit.
level	Nominal coverage level.
seed	Optional integer seed.
quiet	Logical; suppress progress messages.

Value

An object of class '"bd_simulation"', with 'parameter' naming the cause and quantity, for example 'c1:b'.

See Also

[bd_simulation_study()], [fit_bd_competing()]

Examples

```
s <- bd_simulation_competing(n = 120, n_sim = 2, n_starts = 1,
                             seed = 1, quiet = TRUE)
s
```

---

bd_simulation_cure *Finite-Sample Simulation Study for the Cure Model*

Description

Finite-Sample Simulation Study for the Cure Model

Usage

```
bd_simulation_cure(
  n = c(100, 200),
  n_sim = 100,
  truth = c(b = 2, c = 1.5, k = 0.5),
  cure_fraction = 0.3,
  censor_rate = 0.2,
  type = c("mixture", "promotion"),
  n_starts = 3,
  level = 0.95,
  seed = NULL,
  quiet = FALSE
)
```

Arguments

n	Vector of sample sizes.
n_sim	Number of replicates at each sample size.
truth	Named vector with ‘b‘, ‘c‘ and ‘k‘ for the susceptible population.
cure_fraction	True proportion of long-term survivors.
censor_rate	Approximate proportion of censoring among susceptibles.
type	‘"mixture"‘ or ‘"promotion"‘.
n_starts	Random starts per fit.
level	Nominal coverage level.
seed	Optional integer seed.
quiet	Logical; suppress progress messages.

Value

An object of class ‘"bd_simulation"‘.

See Also

[bd_simulation_study()], [fit_bd_cure()]

Examples

```
s <- bd_simulation_cure(n = 80, n_sim = 2, n_starts = 1,
                        seed = 1, quiet = TRUE)
s
```

`bd_simulation_study` *Finite-Sample Simulation Study for the Univariate Model*

Description

Draws samples from a known Beta-Danish or Exponentiated Danish distribution, refits each one, and reports the finite-sample behaviour of the maximum likelihood estimator.

Usage

```
bd_simulation_study(
  n = c(50, 100, 200),
  n_sim = 200,
  truth = c(a = 1.5, b = 3, c = 2, k = 0.5),
  submodel = FALSE,
  censor_rate = 0,
  n_starts = 3,
  level = 0.95,
  seed = NULL,
  quiet = FALSE
)

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

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

Arguments

`n`	Vector of sample sizes.
`n_sim`	Number of replicates at each sample size.
`truth`	Named numeric vector of true parameters. Must contain ‘b‘, ‘c‘ and ‘k‘, and ‘a‘ unless ‘submodel‘ is ‘TRUE‘.
`submodel`	Logical; fit the three-parameter ED submodel.
`censor_rate`	Approximate proportion of right-censored observations.
`n_starts`	Random starts per fit, on top of the deterministic grid.
`level`	Nominal coverage level.
`seed`	Optional integer seed.
`quiet`	Logical; suppress progress messages.
`x`	A ‘"bd_simulation"‘ object.
`digits`	Number of significant digits.
`...`	Ignored.
`object`	A ‘"bd_simulation"‘ object.

Details

Read ‘se_ratio‘ and ‘coverage‘ together. A ratio near one with coverage near the nominal level means the asymptotic standard errors are trustworthy at that sample size. A ratio well below one means they are optimistic, and coverage will fall short accordingly.

‘n_fail‘ counts replicates where no admissible optimum was found. A high rate is not a nuisance to be suppressed: it says the parameter region being studied is hard to estimate at that sample size.

Value

An object of class ‘"bd_simulation"‘ with a ‘results‘ data frame, one row per sample size and parameter.

See Also

[bd_simulation_cure()], [bd_simulation_competing()]

Examples

```
# Deliberately tiny so the example runs in seconds. A real study would use
# n_sim in the hundreds; see the vignette.
s <- bd_simulation_study(n = 60, n_sim = 3,
                         truth = c(b = 3, c = 2, k = 0.5),
                         submodel = TRUE, n_starts = 1,
                         seed = 1, quiet = TRUE)
s
```

---

bd_stress_strength *Stress-Strength Reliability*

Description

$R = P(X > Y)$ where X is the strength and Y the stress, each Beta-Danish distributed with its own parameters.

Usage

```
bd_stress_strength(strength, stress, rel.tol = 1e-10)
```

Arguments

strength	Named list or numeric vector giving ‘a‘, ‘b‘, ‘c‘, ‘k‘ for the strength variable X.
stress	Same, for the stress variable Y.
rel.tol	Passed to [stats::integrate()].

Details

$R = \int_0^\infty S_X(y) f_Y(y)\, dy$, evaluated on the Beta scale of Y so the integral is taken over $(0, 1)$.

The convention is the standard reliability one: R is the probability that the strength exceeds the stress, so larger R means a more reliable component. Passing the two arguments the wrong way round returns $1 - R$.

Value

A single probability.

Examples

```
# Identical distributions must give one half
p <- list(a = 1.5, b = 3, c = 2, k = 1)
bd_stress_strength(strength = p, stress = p)

# A stronger component
bd_stress_strength(strength = list(a = 1.5, b = 3, c = 2, k = 0.5),
                   stress   = list(a = 1.5, b = 3, c = 2, k = 2))
```

---

bd_tail_index *Tail Index of the Beta-Danish Distribution*

Description

Tail Index of the Beta-Danish Distribution

Usage

```
bd_tail_index(a, b, c, k)
```

Arguments

a	Shape parameter (beta generator). Accepted for interface consistency; it does not affect the index.
b	Shape parameter. This is the tail index.
c	Shape parameter (baseline). Accepted for interface consistency.
k	Scale parameter (baseline). Accepted for interface consistency.

Details

The survival function is regularly varying at infinity with index $-b$: $S(t) \sim (c/k)^b t^{-b} / \{bB(a, b)\}$. Three consequences follow.

The r-th moment is finite if and only if $b > r$, so the mean needs $b > 1$, the variance $b > 2$, skewness $b > 3$ and kurtosis $b > 4$.

The moment generating function does not exist for any $t > 0$: a regularly varying tail decays polynomially, so $E(e^{tZ})$ diverges. Characteristic-function or Laplace-transform arguments must be used instead of MGF ones anywhere in this family.

The distribution lies in the Frechet domain of attraction, so sample maxima normalise to a Frechet limit with shape b, not to a Gumbel limit.

Value

A list with the tail index, the highest finite moment order, and a note on the moment generating function.

Examples

```
bd_tail_index(a = 1.5, b = 3, c = 2, k = 1)
```

`bd_ttt_plot` *Scaled Total Time on Test Plot*

Description

Draws the scaled total time on test transform, a distribution-free way of judging hazard shape before any model is fitted.

Usage

```
bd_ttt_plot(
  time,
  status = NULL,
  add = FALSE,
  col = "steelblue",
  lwd = 2,
  main = "Scaled total time on test",
  xlab = "i / n",
  ylab = expression(phi(i/n)),
  ...
)
```

Arguments

`time`	Numeric vector of observed times, or a fitted ‘"betadanish"‘ object, from which the times are taken.
`status`	Optional event indicator. Censored observations are dropped, since the transform is defined for complete samples.
`add`	Logical; add to an existing plot rather than starting a new one.
`col`, `lwd`	Colour and line width for the curve.
`main`, `xlab`, `ylab`	Labels.
`...`	Further graphical parameters.

Details

For an ordered sample $x_{(1)} \leq \cdots \leq x_{(n)}$, the scaled transform at i/n is

$$\phi(i/n) = \frac{\sum_{j=1}^{i} x_{(j)} + (n-i)x_{(i)}}{\sum_{j=1}^{n} x_{(j)}}.$$

Read it against the diagonal. A curve entirely above the diagonal indicates an increasing hazard, entirely below a decreasing one; a curve that starts below and crosses above suggests a bathtub shape, and the reverse suggests a unimodal one. A curve close to the diagonal indicates a constant hazard, that is an exponential sample.

This is a shape diagnostic, not a test. It is worth drawing before choosing between the four-parameter model and its submodel, because it says which hazard shapes the data can support without assuming any of them.

Value

Invisibly, a data frame with the plotting coordinates ‘i_n‘ and ‘phi‘, and an attribute ‘"shape"‘ giving the suggested hazard shape.

See Also

[bd_hazard_shape()] for the fitted counterpart

Examples

```
data(guinea_pig)
ttt <- bd_ttt_plot(guinea_pig$time)
attr(ttt, "shape")
```

---

bd_wald_ci *Log-Scale Wald Confidence Intervals*

Description

Symmetric Wald intervals on the log-parameter scale, exponentiated back.

Usage

```
bd_wald_ci(object, level = 0.95)
```

Arguments

object	A fitted ‘"betadanish"‘ object.
level	Confidence level. Default 0.95.

Details

All four parameters are strictly positive, so a symmetric interval on the natural scale can and does cross zero when a parameter is weakly identified. Building the interval on the log scale and exponentiating keeps it inside the parameter space: $\hat{\theta}\exp(\pm z\,\mathrm{SE}(\log\hat{\theta}))$.

This is still a Wald interval, and it inherits the usual weakness: it assumes the log-likelihood is approximately quadratic. Where it is not – which is exactly where these intervals matter – prefer [bd_profile_ci()].

Value

A matrix with one row per parameter and columns ‘estimate‘, ‘se‘, ‘lower‘ and ‘upper‘.

See Also

[bd_profile_ci()], [bd_identified_coef()]

Examples

```
dat <- simulate_bd_data(150, a = 1, b = 3, c = 2, k = 0.5, seed = 1)
fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                      submodel = TRUE, n_starts = 3)
bd_wald_ci(fit)
```

BetaDanish *The Beta-Danish Distribution*

Description

Density, distribution function, quantile function, hazard function, survival function and random generation for the four-parameter Beta-Danish distribution.

Usage

```
dbetadanish(x, a, b, c, k, log = FALSE)

pbetadanish(q, a, b, c, k, lower.tail = TRUE, log.p = FALSE)

qbetadanish(p, a, b, c, k, lower.tail = TRUE, log.p = FALSE)

rbetadanish(n, a, b, c, k)

sbetadanish(x, a, b, c, k, log = FALSE)

hbetadanish(x, a, b, c, k, log = FALSE)
```

Arguments

x, q	Vector of quantiles (time points).
a	Shape parameter (beta generator). Set ‘a = 1‘ for the three-parameter Exponentiated Danish (ED) submodel.
b	Shape parameter (beta generator). Governs the upper tail: the survival function is regularly varying with index ‘-b‘.
c	Shape parameter (baseline shape).
k	Scale parameter (baseline scale).
log, log.p	Logical; if ‘TRUE‘, densities/probabilities are returned on the log scale.
lower.tail	Logical; if ‘TRUE‘ (default) probabilities are $P[X \le x]$, otherwise $P[X > x]$.
p	Vector of probabilities.
n	Number of observations to generate.

Details

With baseline $G(t) = \{kt/(1+kt)\}^c$, the Beta-Danish CDF is the regularised incomplete beta function $F(t) = I_{G(t)}(a,b)$ and the density is

$$f(t) = \frac{c\,k^{ca}\,t^{ca-1}}{B(a,b)\,(1+kt)^{ca+1}}\big\{1-G(t)\big\}^{b-1}.$$

The family accommodates decreasing, increasing, unimodal and bathtub-shaped hazard rates.

Value

‘dbetadanish‘ gives the density, ‘pbetadanish‘ the distribution function, ‘qbetadanish‘ the quantile function, ‘sbetadanish‘ the survival function, ‘hbetadanish‘ the hazard function, and ‘rbetadanish‘ generates random deviates. Length is the maximum of the lengths of the numeric arguments.

Numerical notes

All three of the density, distribution and quantile functions are evaluated so as to retain full relative precision in the upper tail, which is where this family is distinctive and where naive implementations fail:

* $\log\{1-G(t)\}$ is formed as ‘log(-expm1(c * -log1p(1/(k*t))))‘. Writing $\log G$ as $-c\,\log(1+1/kt)$ avoids the cancellation in $\log(kt) - \log(1+kt)$, which loses all significant digits once $kt \gtrsim 10^{15}$. * The survival function uses the beta mirror identity $1 - I_y(a,b) = I_{1-y}(b,a)$, so no probability near one is ever subtracted from one. * ‘qbetadanish‘ obtains $1-u$ directly via $1 - q\beta(p;a,b) = q\beta(p;b,a)$ with the tail flag reversed, so ‘1 - p‘ is never formed.

Parameters recycle element-wise against ‘x‘, ‘q‘ or ‘p‘ following the usual convention for R distribution functions, so a per-observation scale (as produced by an AFT link) may be supplied directly.

As $t \to \infty$, $S(t) \propto t^{-b}$; consequently $E(X^r)$ is finite if and only if $b > r$.

References

Ahmad, B., & Danish, M. Y. (2025). Development and characterization of a flexible three-parameter lifetime distribution: theoretical properties and real-world applications. *Journal of Applied Mathematics, Statistics and Informatics*, 21(1). doi:10.2478/jamsi20250010

Examples

```
dbetadanish(x = 2, a = 1.5, b = 2, c = 3, k = 0.5)
pbetadanish(q = 2, a = 1.5, b = 2, c = 3, k = 0.5)
qbetadanish(p = 0.5, a = 1.5, b = 2, c = 3, k = 0.5)
hbetadanish(x = 2, a = 1.5, b = 2, c = 3, k = 0.5)
rbetadanish(n = 10, a = 1.5, b = 2, c = 3, k = 0.5)

# Per-observation scale, as used by the AFT link
dbetadanish(c(1, 2, 3), a = 1, b = 2, c = 1.5, k = c(0.5, 1, 2))

# The survival tail is regularly varying with index -b
round(log(sbetadanish(c(1e10, 1e12), 1.5, 3, 2, 1)), 4)
```

---

carbon_fibres *Breaking Stress of Carbon Fibres*

Description

Breaking stress (in GPa) of 100 carbon fibre specimens. This dataset exhibits a unimodal (increasing-then-decreasing) hazard pattern that classical distributions like the Weibull cannot adequately capture.

Usage

```
carbon_fibres
```

Format

A data frame with 100 rows and 2 columns:

time Breaking stress in GPa

status Event indicator (1 = event occurred)

Source

Nichols, M. D., & Padgett, W. J. (2006). A bootstrap control chart for Weibull percentiles. Quality and Reliability Engineering International, 22(2), 141-151.

Examples

```
data(carbon_fibres)

fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = carbon_fibres)
```

`cif_betadanish` *Cumulative Incidence Function*

Description

Computes the cumulative incidence of one cause from a fitted Beta-Danish competing risks model, $F_j(t) = \int_0^t f_j(u) \prod_{l \neq j} S_l(u)\, du$.

Usage

```
cif_betadanish(fit, tvec, cause_idx, x = NULL)
```

Arguments

`fit`	An object of class ‘"bd_competing"‘.
`tvec`	Numeric vector of times at which to evaluate the CIF.
`cause_idx`	The cause to evaluate, matching one of the fitted causes.
`x`	Optional covariate values at which to evaluate, one per covariate in the fit. Defaults to zero for every covariate, which is the reference subject. Ignored when the model has no covariates.

Details

The integrand is the cause-specific density multiplied by the survival of every other cause, so the sum of all cause CIFs approaches the overall failure probability rather than one, and each individual CIF is bounded by it.

Any covariate effect is folded into the scale parameter before integration, since accelerating time by λ is equivalent to dividing k by λ.

Value

A numeric vector of probabilities the same length as ‘tvec‘.

See Also

[fit_bd_competing()], [cif_compare()]

Examples

```
d <- simulate_bd_competing_data(120, seed = 2)
fit <- fit_bd_competing(d$time, d$cause, submodel = TRUE, n_starts = 1)
cif_betadanish(fit, tvec = c(1, 5, 10), cause_idx = 1)
```

`cif_compare` *Compare Model-Based CIF to the Aalen-Johansen Estimator*

Description

Computes the nonparametric Aalen-Johansen CIF (via **cmprsk**) for each competing-risks cause, overlays it on the fitted Beta-Danish CIF, and returns the Aalen-Johansen times/estimates, the fitted CIF values on a common grid, and Gray's CIF-equality test where applicable.

Usage

```
cif_compare(fit, tmax = NULL, n_grid = 160, plot = TRUE)
```

Arguments

`fit`	A fitted object of class `"bd_competing"`.
`tmax`	Optional upper time for evaluation; default the 95th percentile of observed times.
`n_grid`	Number of time points on the evaluation grid (default 160).
`plot`	Logical; if `TRUE` (default) draws a panel of overlays.

Details

Requires **cmprsk** (Suggests).

Value

A list with elements `tgrid`, `cif_fit` (data frame long format), `cif_aj` (data frame long format) and optionally `gray_test`.

Examples

```
## Not run:
set.seed(1)
T1 <- rbetadanish(200, 1.2, 1.5, 1.0, 0.4)
T2 <- rbetadanish(200, 1.0, 2.0, 1.0, 0.2)
C  <- stats::rexp(200, 0.05)
time  <- pmin(T1, T2, C)
cause <- ifelse(time == C, 0L, ifelse(T1 <= T2, 1L, 2L))
fit <- fit_bd_competing(time = time, cause = cause)
cif_compare(fit)   # requires cmprsk to be installed

## End(Not run)
```

coef.betadanish *Extract Coefficients*

Description

Extract Coefficients

Usage

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

Arguments

object	An object of class ‘betadanish‘.
...	Further arguments passed to or from other methods.

Value

A named numeric vector of maximum likelihood parameter estimates.

compare_distributions *Compare Beta-Danish with Standard Distributions*

Description

Fits standard lifetime distributions (Weibull, Log-Normal, Log-Logistic, Gamma, Exponential) using the ‘flexsurv‘ package and compares them to the Beta-Danish fit.

Usage

```
compare_distributions(object)
```

Arguments

object	A fitted ‘betadanish‘ object.

Value

A ranked data frame of model comparisons.

compare_models *Compare Nested Beta-Danish Models*

Description

Performs a Likelihood Ratio Test (LRT) between the 4-parameter full model and the 3-parameter submodel.

Usage

```
compare_models(full_model, sub_model)
```

Arguments

full_model	A fitted 4-parameter 'betadanish' object.
sub_model	A fitted 3-parameter 'betadanish' object.

Value

A data frame with the test statistic and p-value.

ExponentiatedDanish *The Exponentiated Danish Distribution*

Description

Density, distribution function, quantile function, survival function, hazard function and random generation for the three-parameter Exponentiated Danish distribution, the $a = 1$ submodel of the Beta-Danish family.

Usage

```
ded(x, b, c, k, log = FALSE)

ped(q, b, c, k, lower.tail = TRUE, log.p = FALSE)

qed(p, b, c, k, lower.tail = TRUE, log.p = FALSE)

red(n, b, c, k)

sed(x, b, c, k, log = FALSE)

hed(x, b, c, k, log = FALSE)
```

Arguments

x, q	Vector of quantiles.
b, c, k	Shape, shape and scale parameters.
log, log.p	Logical; return values on the log scale.
lower.tail	Logical; if 'TRUE' (default) probabilities are $P[X \leq x]$.
p	Vector of probabilities.
n	Number of observations to generate.

Details

These are thin wrappers that fix $a = 1$, provided because the ED submodel is a named distribution in its own right in the underlying work and writing 'dbetadanish(x, 1, b, c, k)' obscures that.

At $a = 1$ the beta generator collapses and the distribution function simplifies to $F(t) = 1 - \{1 - G(t)\}^b$ with $G(t) = \{kt/(1 + kt)\}^c$. The upper tail still has index $-b$, so the moment condition $b > r$ is unchanged.

Value

Vectors of the same form as the Beta-Danish equivalents.

See Also

[BetaDanish] for the four-parameter parent, [fit_betadanish()] with 'submodel = TRUE' for estimation.

Examples

```
ded(2, b = 3, c = 2, k = 1)
ped(2, b = 3, c = 2, k = 1)
qed(0.5, b = 3, c = 2, k = 1)
red(5, b = 3, c = 2, k = 1)

# Identical to the a = 1 parent
all.equal(ded(2, 3, 2, 1), dbetadanish(2, 1, 3, 2, 1))
```

---

fit_bd_aft *Fit Beta-Danish AFT Regression Model*

Description

Fits an Accelerated Failure Time (AFT) regression model using the Exponentiated Danish (ED) kernel, that is the Beta-Danish distribution with a = 1.

Usage

```
fit_bd_aft(formula, data, n_starts = 10, method = "BFGS")
```

Arguments

formula	A survival formula (e.g., 'Surv(time, status) ~ age + treatment').
data	A data frame containing the variables.
n_starts	Integer; number of random starts for optimization.
method	Optimization method passed to 'maxLik'.

Details

To ensure identifiability, the shape parameter 'a' is fixed to 1. The scale parameter 'k' is linked to covariates via 'k_i = exp(X_i Positive coefficients in 'delta' indicate a larger 'k', which corresponds to shorter survival times (accelerated failure).

Value

An object of class 'bd_aft'.

---

fit_bd_competing *Fit a Beta-Danish Competing Risks Model*

Description

Fits a parametric competing risks model assuming independent latent failure times, with each cause-specific baseline following the Beta-Danish distribution. Covariates may be included, entering each cause through an accelerated failure time link.

Usage

```
fit_bd_competing(
  time,
  cause,
  covariates = NULL,
  data = NULL,
  submodel = FALSE,
  n_starts = 5,
  method = "BFGS"
)
```

Arguments

time	Numeric vector of observed times.
cause	Integer vector of event causes: '0' for right-censored and '1, 2, ..., m' for the competing causes.
covariates	Optional covariates. Either a one-sided formula such as '~ age + group', evaluated in 'data', or a numeric matrix or data frame with one row per observation. 'NULL' (default) fits cause-specific baselines with no regression structure.

`data`	Data frame in which to evaluate ‘covariates‘ when it is a formula.
`submodel`	Logical; if ‘TRUE‘, fix ‘a = 1‘ in every cause, giving Exponentiated Danish cause-specific kernels. Default ‘FALSE‘.
`n_starts`	Integer; number of starting points for the joint optimisation.
`method`	Optimisation method passed to [maxLik::maxLik()].

Details

Writing δ_i for the observed cause and d_i for the event indicator, the log-likelihood is

$$\ell = \sum_{i:d_i=1} \Big[\log f_{\delta_i}(y_i) + \sum_{j \neq \delta_i} \log S_j(y_i)\Big] + \sum_{i:d_i=0} \sum_{j=1}^{m} \log S_j(y_i).$$

An event contributes the log-density of its own cause and the log-survival of every other cause at the same time; a censored observation contributes the log-survival of every cause.

Value

An object of class ‘"bd_competing"‘.

Covariates

Each cause carries its own coefficient vector γ_j, acting on the time scale:

$$T_j = T_{0j} \exp(x'\gamma_j),$$

so that a positive coefficient lengthens time to failure from that cause. The likelihood contribution of an event at t is evaluated at the accelerated time $t \exp(-x'\gamma_j)$ with the Jacobian $-x'\gamma_j$ added on the log scale.

The same design matrix is used for every cause, but the coefficients are free to differ, so a covariate may accelerate one cause and retard another. With ‘m‘ causes, ‘p‘ covariates and the full Beta-Danish kernel the model carries ‘m * (4 + p)‘ parameters, which grows quickly: check ‘summary()‘ for standard errors before interpreting any of them.

Identifiability of the cause-specific marginals

Mutual independence of the latent failure times is an *identifying* assumption, not an empirically testable property. Tsiatis (1975) showed that without it infinitely many joint distributions generate the same observable law of (T, δ), so the marginals cannot be recovered from the observed data alone.

Under positive latent dependence the working independence model overstates each cause-specific survival at moderate times, biasing the fitted cumulative incidence functions downward; negative dependence reverses both. The overall survival $S(t) = \prod_j S_j(t)$ is more robust than the individual marginals. Where conclusions rest on absolute cause-specific CIFs rather than on model selection, supplement this fit with a copula-based sensitivity analysis.

References

Tsiatis, A. (1975). A nonidentifiability aspect of the problem of competing risks. *Proceedings of the National Academy of Sciences*, 72(1), 20-22. doi:10.1073/pnas.72.1.20

See Also

[cif_betadanish()], [cif_compare()], [simulate_bd_competing_data()]

Examples

```
# Without covariates
d <- simulate_bd_competing_data(150, seed = 1)
fit <- fit_bd_competing(d$time, d$cause, submodel = TRUE, n_starts = 1)
fit

# With one binary covariate. Note that simulate_bd_competing_data() only
# creates the covariate column when 'gammas' is supplied.
dx <- simulate_bd_competing_data(150, gammas = c(0.5, -0.5), seed = 2)
fit2 <- fit_bd_competing(dx$time, dx$cause, covariates = ~ x, data = dx,
                         submodel = TRUE, n_starts = 1)
fit2$coefficients
```

`fit_bd_cure` *Fit Beta-Danish Cure Models*

Description

Fits mixture and promotion-time (non-mixture) cure models using the Beta-Danish AFT baseline.

Usage

```
fit_bd_cure(
  formula_aft,
  formula_cure,
  data,
  type = c("mixture", "promotion"),
  n_starts = 10,
  method = "BFGS"
)
```

Arguments

`formula_aft`	A formula for the latency component (e.g., 'Surv(time, status) ~ age').
`formula_cure`	A one-sided formula for the incidence/cure component (e.g., '~ treatment').
`data`	A data frame containing the variables.
`type`	Character; either '"mixture"' or '"promotion"'.
`n_starts`	Integer; number of random starts for optimization.
`method`	Optimization method passed to 'maxLik'.

Details

In the **mixture** model, the population is split into susceptible and cured fractions. The susceptible probability is modeled via logistic regression: ‘pi = exp(Z

In the **promotion-time** (non-mixture) model, the cure fraction is derived from a latent Poisson process of clonogenic cells: ‘theta = exp(Z The cure fraction is ‘exp(-theta)‘.

Value

An object of class ‘bd_cure‘.

`fit_betadanish` *Fit the Beta-Danish Distribution to Survival Data*

Description

Fits the Beta-Danish distribution by maximum likelihood. Complete and right-censored samples are both supported, via a ‘survival::Surv‘ response.

Usage

```
fit_betadanish(
  formula,
  data,
  submodel = FALSE,
  n_starts = 10,
  method = "BFGS",
  check_identifiability = TRUE,
  penalty = 0,
  penalty_center = NULL,
  grouped = FALSE,
  delta = NULL
)
```

Arguments

`formula`	A formula whose left-hand side is a ‘Surv‘ object. Use ‘~ 1‘ for a model without covariates.
`data`	A data frame containing the variables in ‘formula‘.
`submodel`	Logical; if ‘TRUE‘, fits the three-parameter Exponentiated Danish (ED) submodel by fixing ‘a = 1‘.
`n_starts`	Integer; number of random starting points for the multi-start optimisation. Default 10.
`method`	Character; optimisation method passed to ‘maxLik::maxLik‘.
`check_identifiability`	Logical; if ‘TRUE‘ (default), issue warnings when the fit lands in a region where the parameters are weakly identified.

penalty	Non-negative ridge penalty applied on the log-parameter scale. ‘0‘ (default) is ordinary maximum likelihood. A small positive value stabilises the fit when the likelihood is nearly flat, at the cost of bias toward ‘penalty_center‘.
penalty_center	Numeric vector on the log-parameter scale toward which the penalty shrinks, or ‘NULL‘ (default) to shrink toward the best unpenalised start.
grouped	Logical; if ‘TRUE‘, use the grouped (interval) likelihood appropriate to times recorded on a coarse grid. Default ‘FALSE‘.
delta	Recording increment for ‘grouped = TRUE‘. ‘NULL‘ (default) infers it from the spacing of the observed times.

Details

Optimisation is carried out on log-transformed parameters so that positivity is enforced without constraints; estimates and the variance-covariance matrix are returned on the natural scale, the latter via the delta method.

Value

An object of S3 class ‘"betadanish"‘ with components including ‘coefficients‘, ‘logLik‘, ‘vcov‘, ‘npar‘, ‘nobs‘, ‘convergence‘ and ‘diagnostics‘.

Grouped data

Survival times are often recorded on a grid – whole days, whole months – and the point-density likelihood is not appropriate for them. It treats a rounded value as an exact observation, which overstates the information in the sample and understates every standard error. With ‘grouped = TRUE‘ an event recorded at t contributes $\log\{F(t+\delta/2)-F(t-\delta/2)\}$ instead of $\log f(t)$; censored observations are unchanged. The cell probability falls back to $f(t)\delta$ only where the difference of two nearly equal distribution values has cancelled to zero.

‘read_survival_data()‘ reports an inferred ‘grid_step‘ for exactly this purpose, and this function warns when the times look grid-recorded but ‘grouped = FALSE‘.

Penalised fitting

‘penalty > 0‘ adds $\lambda\sum(\theta-\mu)^2$ on the log-parameter scale. This is worth reaching for when the likelihood is flat along the (a, c) direction and the unpenalised optimiser wanders, but it is a deliberate bias: the estimates are shrunk toward ‘penalty_center‘.

The reported ‘logLik‘ is always the **unpenalised** log-likelihood evaluated at the penalised estimate, so that AIC, BIC and likelihood ratio tests remain comparable across fits. The objective actually maximised is stored separately as ‘penalised_logLik‘. Treat the degrees of freedom as nominal: shrinkage reduces the effective number of parameters, so information criteria are conservative under penalisation.

Identifiability

The four-parameter model is not uniformly well identified, and a converged fit is not by itself evidence that it is. Two regions warrant care.

* **The $b = 1$ ridge.** At $b = 1$ the beta generator collapses and the model is non-identifiable. A fit with $\hat{b}$ within about two standard errors of one lies close to that ridge; the likelihood is nearly flat along it, so the individual estimates carry little information even though the fitted survival curve may look excellent. * **Lower-tail (a, c) confounding.** Near the lower tail, a and c enter almost exclusively through the product ca, so the expected Fisher information is close to singular in that direction. A fitted correlation between $\hat{a}$ and $\hat{c}$ above about 0.95 in absolute value indicates that only the product is being estimated.

In either case the ED submodel ('submodel = TRUE') is usually the honest report, and a likelihood ratio test via [compare_models()] will normally fail to reject it. Set 'check_identifiability = FALSE' to silence the warnings once you have satisfied yourself that they are understood.

See Also

[compare_models()], [gof_betadanish()], [compare_distributions()]

Examples

```
set.seed(123)
sim_time   <- rbetadanish(150, a = 1.5, b = 3, c = 2, k = 0.5)
sim_status <- rbinom(150, 1, 0.85)
dat <- data.frame(time = sim_time, status = sim_status)

fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat)
summary(fit)

fit_sub <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                          submodel = TRUE)
compare_models(fit, fit_sub)
```

`gof_betadanish` *Goodness-of-Fit Statistics for Beta-Danish Models*

Description

Computes Information Criteria (AIC, BIC, HQIC, AICC) and the Kolmogorov-Smirnov (K-S) statistic for a fitted Beta-Danish model.

Usage

```
gof_betadanish(object)
```

Arguments

`object`	A fitted 'betadanish' object.

Value

A list containing the Information Criteria and the K-S statistic.

guinea_pig *Guinea Pig Survival Times*

Description

Survival times, in days, of 72 guinea pigs injected with virulent tubercle bacilli, taken from the principal regimen of the study of Bjerkedal (1960). A complete sample: every animal was observed to death, so ‘status‘ is 1 throughout.

Usage

```
guinea_pig
```

Format

A data frame with 72 rows and 2 columns:

time Survival time in days

status Event indicator, 1 for all observations (no censoring)

Details

These data are the reference case for a well-identified four-parameter fit. The scaled total-time-on-test transform is unimodal and the upper tail is determinate rather than heavy, and the maximum-likelihood fit of the full Beta-Danish model attains a finite interior optimum: every estimate lies strictly inside the parameter space, with $\hat{b} = 3.64$ (Wald standard error 1.20), so that $(\hat{b} - 1)/\mathrm{SE} = 2.20$ places b more than two standard errors clear of the $b = 1$ identifiability ridge.

By contrast, on ‘remission‘ and ‘carbon_fibres‘ the four-parameter fit sits on the flat (a, c) direction of the likelihood with $\hat{a} < 1$ and is only weakly identified. Use this dataset when you want to see the parent model behaving well; see the Identifiability section of [fit_betadanish()] for what to watch for elsewhere.

Source

Bjerkedal, T. (1960). Acquisition of resistance in guinea pigs infected with different doses of virulent tubercle bacilli. *American Journal of Epidemiology*, 72(1), 130-148. doi:10.1093/oxfordjournals.aje.a120129

Examples

```
data(guinea_pig)
summary(guinea_pig$time)

fit_full <- fit_betadanish(survival::Surv(time, status) ~ 1,
                           data = guinea_pig)
fit_sub  <- fit_betadanish(survival::Surv(time, status) ~ 1,
                           data = guinea_pig, submodel = TRUE)
compare_models(fit_full, fit_sub)
```

leukemia *Acute Myelogenous Leukemia Survival*

Description

Survival times (in weeks) for 23 patients with acute myelogenous leukemia. A classic, small dataset perfect for fast testing of censored data workflows.

Usage

```
leukemia
```

Format

A data frame with 23 rows and 3 columns:

time Survival time in weeks

status Event indicator (1 = event, 0 = censored)

group Treatment group (Maintained vs Non-maintained)

Source

Miller, R. G. (1997). Survival Analysis. Wiley.

Examples

```
data(leukemia)

fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = leukemia)
```

logLik.betadanish *Extract Log-Likelihood*

Description

Extract Log-Likelihood

Usage

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

Arguments

object	An object of class ‘betadanish‘.
...	Further arguments passed to or from other methods.

Value

An object of class logLik containing the maximized log-likelihood value, with degrees of freedom and number of observations stored as attributes.

`melanoma` *Malignant Melanoma Survival After Surgery*

Description

Survival times for 205 patients with malignant melanoma after surgery. This rich clinical dataset includes multiple covariates and heavy censoring.

Usage

```
melanoma
```

Format

A data frame with 205 rows and 7 columns:

time Survival time in days

status Event indicator (1 = died from melanoma, 0 = alive, 2 = other death)

thickness Tumor thickness in mm

sex Patient sex (1 = male, 0 = female)

age Patient age in years

ulcer Ulceration indicator (1 = present, 0 = absent)

year Year of operation

Source

Andersen, P. K., Borgan, O., Gill, R. D., & Keiding, N. (1993). Statistical Models Based on Counting Processes. Springer.

Examples

```
data(melanoma)

# Treat status 1 as event, others as censored
melanoma$event <- ifelse(melanoma$status == 1, 1, 0)
fit <- fit_betadanish(survival::Surv(time, event) ~ age + thickness, data = melanoma)
```

plot.bd_aft *Cox-Snell Residual Plot for AFT and Cure Fits*

Description

Diagnostic Cox-Snell residual plot for a fitted AFT or cure model. Under a correctly specified model the residuals behave like a unit exponential sample, so the Kaplan-Meier estimate of their cumulative hazard should follow the 45-degree line.

Usage

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

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

Arguments

x	A fitted ‘"bd_aft"‘ or ‘"bd_cure"‘ object.
...	Further graphical parameters passed to ‘plot‘.

Value

Invisibly returns ‘x‘.

Examples

```
set.seed(42)
n <- 300
xcov  <- stats::rnorm(n)
k_i   <- exp(-0.5 - 0.3 * xcov)
t_sim <- rbetadanish(n, a = 1, b = 2, c = 1.5, k = k_i)
dat   <- data.frame(time = t_sim,
                    status = stats::rbinom(n, 1, 0.85),
                    x = xcov)
fit <- fit_bd_aft(survival::Surv(time, status) ~ x, data = dat, n_starts = 5)
plot(fit)
```

`plot.bd_bayes` *Diagnostic Plots for a Bayesian Fit*

Description

Trace and posterior density plots for each parameter of a [bayes_betadanish()] fit.

Usage

```
## S3 method for class 'bd_bayes'
plot(
  x,
  which = NULL,
  type = c("both", "trace", "density"),
  col = "steelblue",
  ...
)
```

Arguments

`x`	An object of class ‘"bd_bayes"‘.
`which`	Optional character vector of parameters to show. Defaults to all of them.
`type`	‘"both"‘ (default) for trace and density side by side, or ‘"trace"‘ or ‘"density"‘ alone.
`col`	Colour for the traces and densities.
`...`	Further graphical parameters.

Details

Read the traces first. A well-mixed chain looks like noise around a stable level, with no drift and no long excursions. Visible trend means the burn-in was too short; a chain that sticks at one value for many iterations means the proposal is too wide and almost every move is being rejected, which is worth fixing with ‘tune‘ before interpreting anything.

The graphical parameters are restored on exit, so the function leaves the device as it found it.

Value

Invisibly returns ‘x‘.

See Also

[bayes_betadanish()]

Examples

```
if (requireNamespace("MCMCpack", quietly = TRUE) &&
    requireNamespace("coda", quietly = TRUE)) {
  dat <- simulate_bd_data(80, a = 1, b = 3, c = 2, k = 0.5, seed = 6)
  bfit <- bayes_betadanish(dat$time, dat$status, submodel = TRUE,
                           burnin = 200, mcmc = 800, seed = 1)
  plot(bfit)
}
```

`plot.betadanish` *Plot Diagnostics for Beta-Danish Fit*

Description

Generates diagnostic plots for a fitted Beta-Danish model, including survival, hazard, density, PP, and QQ plots.

Usage

```
## S3 method for class 'betadanish'
plot(x, type = c("survival", "hazard", "density", "pp", "qq", "all"), ...)
```

Arguments

`x`	A fitted ‘betadanish‘ object.
`type`	Character string specifying the plot type: ‘"survival"‘, ‘"hazard"‘, ‘"density"‘, ‘"pp"‘, ‘"qq"‘, or ‘"all"‘.
`...`	Additional arguments passed to the base ‘plot‘ function.

Value

Invisibly returns the input betadanish object. Called mainly for its side effect of producing diagnostic plots.

`print.betadanish` *Print Method for Beta-Danish Fit*

Description

Print Method for Beta-Danish Fit

Usage

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

Arguments

x	An object of class ‘betadanish‘.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the input betadanish object. Called mainly for its side effect of printing the fitted model summary.

---

print.summary.betadanish *Print Summary Method for Beta-Danish Fit*

Description

Print Summary Method for Beta-Danish Fit

Usage

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

Arguments

x	An object of class ‘summary.betadanish‘.
...	Further arguments passed to or from other methods.

Value

Invisibly returns the input summary.betadanish object. Called mainly for its side effect of printing the coefficient table and fit statistics.

---

read_survival_data *Read and Prepare Survival Data*

Description

Reads survival data from a delimited text file or an Excel workbook and returns a clean data frame ready for [fit_betadanish()] and the regression fitters. Columns are selected by name, covariates keep their original type, and the result carries a report describing what was read.

Usage

```
read_survival_data(
  file,
  time_col = NULL,
  status_col = NULL,
  covar_cols = NULL,
  cause_col = NULL,
  sep = ",",
  dec = ".",
  encoding = "unknown",
  na.strings = c("NA", "", ".", "#N/A"),
  drop_na = TRUE,
  quiet = FALSE
)
```

Arguments

file	Path to a '.csv', '.txt', '.tsv', '.xls' or '.xlsx' file.
time_col	Name of the time column. If 'NULL' (default), the column is guessed from a list of common names and the choice is reported.
status_col	Name of the event indicator (1 = event, 0 = censored). If 'NULL', guessed the same way; if no candidate is found, all observations are treated as uncensored.
covar_cols	Covariate columns to retain. Either a character vector of column names, the string '"all"' to retain every column that is not part of the response, or 'NULL' (default) to retain none. Whatever is dropped is named in a message unless 'quiet = TRUE'.
cause_col	Name of a competing-risks cause column, or 'NULL'. When supplied, code 0 means censored and positive integers index the causes.
sep	Field separator. Defaults to a comma. Use '"\t"' for tab-delimited input, '";"' for semicolon-delimited.
dec	Decimal mark. Use '","' for European-format numerics.
encoding	File encoding passed to [utils::read.table()], for example '"UTF-8"' or '"latin1"'.
na.strings	Strings to treat as missing.
drop_na	Logical; drop incomplete rows. Default 'TRUE'.
quiet	Logical; suppress the informational messages.

Details

Column guessing is deliberately conservative. Names meaning "censoring indicator" are excluded from the candidate list, because 'censor = 1' means *censored* in some conventions and *observed* in others, and guessing wrong would silently invert every event. Name the column explicitly if in doubt.

A covariate that happens to be called 'time', 'status' or 'cause' is renamed with a '_cov' suffix and a warning rather than colliding with the response.

Excel input requires the 'readxl' package.

Value

A data frame with columns ‘time‘, ‘status‘, an optional ‘cause‘, and any retained covariates. The ‘"bd_data_report"‘ attribute is a list recording the file, which columns were used, rows read and kept, number of events, censoring proportion, retained and dropped columns, and the inferred recording grid.

See Also

[fit_betadanish()], [fit_bd_aft()], [fit_bd_competing()]

Examples

```
# A complete uncensored sample with a non-standard column name
f <- system.file("extdata", "complete_sample.csv", package = "BetaDanish")
dat <- read_survival_data(f, time_col = "survival_days", quiet = TRUE)
attr(dat, "bd_data_report")$rows_kept

# Column names guessed, and every covariate retained
f2 <- system.file("extdata", "censored_sample.csv", package = "BetaDanish")
dat2 <- read_survival_data(f2, covar_cols = "all", quiet = TRUE)
names(dat2)

# Competing risks
f3 <- system.file("extdata", "competing_sample.csv", package = "BetaDanish")
dat3 <- read_survival_data(f3, time_col = "time", cause_col = "cause",
                           quiet = TRUE)
table(dat3$cause)
```

---

remission *Bladder Cancer Remission Times*

Description

Remission times (in months) for 128 bladder cancer patients. This is a complete (uncensored) sample widely used in lifetime distribution literature to demonstrate decreasing or right-skewed hazard rates.

Usage

```
remission
```

Format

A data frame with 128 rows and 2 columns:

time Remission time in months

status Event indicator (1 = event occurred)

Source

Lee, E. T., & Wang, J. W. (2003). Statistical Methods for Survival Data Analysis (3rd ed.). Wiley.

Examples

```
data(remission)

fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = remission)
summary(fit)
```

`report_betadanish` *Create a Compact Report from a Beta-Danish Model Fit*

Description

Collects the headline quantities from a fitted model into a small object with a ‘print‘ method. Information criteria are computed from the fitted log-likelihood via the ‘logLik‘ method, so they cannot fall out of step with the fit.

Usage

```
report_betadanish(fit)

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

Arguments

`fit`	A fitted ‘"betadanish"‘ object.
`x`	A ‘"betadanish_report"‘ object.
`...`	Ignored.

Value

An object of class ‘"betadanish_report"‘.

Invisibly returns ‘x‘.

Examples

```
set.seed(1)
dat <- simulate_bd_data(120, a = 1, b = 3, c = 2, k = 0.5)
fit <- fit_betadanish(survival::Surv(time, status) ~ 1, data = dat,
                      submodel = TRUE)
report_betadanish(fit)
```

`simulate_bd_competing_data` *Simulate Competing Risks Data*

Description

Simulate Competing Risks Data

Usage

```
simulate_bd_competing_data(
  n,
 pars = list(c(a = 1, b = 2.5, c = 1.8, k = 0.04), c(a = 1, b = 3.5, c = 1.2, k = 0.02)),
  gammas = NULL,
  censor_rate = 0.15,
  seed = NULL
)
```

Arguments

`n`	Sample size.
`pars`	List of parameter vectors, one per cause, each a named vector with ‘b‘, ‘c‘, ‘k‘ and optionally ‘a‘. Defaults to two causes.
`gammas`	Numeric vector of accelerated failure time coefficients, one per cause, acting on a single binary covariate. ‘NULL‘ for no covariate.
`censor_rate`	Approximate proportion of right-censored observations.
`seed`	Optional integer seed.

Value

A data frame with ‘time‘, ‘cause‘ (0 for censored) and, when ‘gammas‘ is supplied, the covariate ‘x‘.

See Also

[fit_bd_competing()]

Examples

```
d <- simulate_bd_competing_data(200, seed = 1)
table(d$cause)
```

`simulate_bd_cure_data` *Simulate Beta-Danish Cure Data*

Description

Generates synthetic survival data from a Beta-Danish mixture or promotion-time cure model, incorporating covariates.

Usage

```
simulate_bd_cure_data(
  n,
  type = c("mixture", "promotion"),
  a = 1,
  b = 2,
  c = 1.5,
  delta,
  gamma,
  X,
  Z,
  target_censor = 0.3,
  seed = NULL
)
```

Arguments

`n`	Integer; number of observations.
`type`	Character; ‘"mixture"‘ or ‘"promotion"‘.
`a, b, c`	Numeric; baseline shape parameters.
`delta`	Numeric vector; coefficients for the latency scale ‘k‘.
`gamma`	Numeric vector; coefficients for the incidence/cure component.
`X`	Matrix; design matrix for latency (must match length of ‘delta‘).
`Z`	Matrix; design matrix for incidence (must match length of ‘gamma‘).
`target_censor`	Numeric; target proportion of censoring to calibrate the exponential censoring rate. Default is 0.3.
`seed`	Integer; optional seed.

Value

A list containing the simulated ‘data‘ (time, status), the ‘cured‘ indicator, and the true parameters.

simulate_bd_data *Simulate Data from the Beta-Danish Distribution*

Description

Generates synthetic survival data from the Beta-Danish distribution, with optional right-censoring.

Usage

```
simulate_bd_data(n, a, b, c, k, censor_rate = 0, seed = NULL)
```

Arguments

n	Integer; number of observations to simulate.
a, b, c, k	Numeric; parameters of the Beta-Danish distribution.
censor_rate	Numeric; rate parameter for the exponential censoring distribution. If '0' (default), no censoring is applied.
seed	Integer; optional seed for reproducibility.

Value

A data frame with columns 'time' and 'status'.

Examples

```
# Simulate complete data
dat <- simulate_bd_data(n = 100, a = 1.5, b = 2, c = 3, k = 0.5)

# Simulate censored data
dat_cens <- simulate_bd_data(n = 100, a = 1.5, b = 2, c = 3, k = 0.5, censor_rate = 0.1)
```

summary.betadanish *Summary Method for Beta-Danish Fit*

Description

Summary Method for Beta-Danish Fit

Usage

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

Arguments

`object`	An object of class 'betadanish'.
`...`	Further arguments passed to or from other methods.

Value

An object of class summary.betadanish containing coefficient estimates, standard errors, test statistics, p-values, log-likelihood, and model selection criteria.

`transplant` *Bone Marrow Transplant Survival*

Description

Survival times for 91 patients with refractory acute lymphoblastic leukemia who received either an allogeneic or autologous bone marrow transplant. This dataset includes right-censoring and a treatment covariate, making it ideal for demonstrating cure-rate models and AFT regression.

Usage

```
transplant
```

Format

A data frame with 91 rows and 3 columns:

time Survival time in days

status Event indicator (1 = death/relapse, 0 = censored)

group Treatment group (0 = Allogeneic, 1 = Autologous)

Source

Klein, J. P., & Moeschberger, M. L. (2003). Survival Analysis: Techniques for Censored and Truncated Data (2nd ed.). Springer.

Examples

```
data(transplant)

# Fit a model with a covariate
fit <- fit_bd_aft(survival::Surv(time, status) ~ group, data = transplant)
```

`vcov.betadanish` *Extract Variance-Covariance Matrix*

Description

Extract Variance-Covariance Matrix

Usage

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

Arguments

`object`	An object of class ‘betadanish‘.
`...`	Further arguments passed to or from other methods.

Value

A numeric variance-covariance matrix for the estimated model parameters.

Index