Package: flexhaz (via r-universe)

October 28, 2026

Title Dynamic Failure Rate Distributions for Survival Analysis

Version 0.5.3

Description Flexible framework for specifying survival distributions through their hazard (failure rate) functions. Define arbitrary time-varying hazard functions to model complex failure patterns including bathtub curves, proportional hazards with covariates, and other non-standard hazard behaviors. Provides automatic computation of survival, CDF, PDF, quantiles, and sampling. Implements the likelihood model interface for maximum likelihood estimation with right-censored and left-censored survival data.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports stats, numDeriv, algebraic.dist, likelihood.model, generics

Depends R (>= 3.5.0)

VignetteBuilder knitr

URL https://github.com/queelius/flexhaz, https://queelius.github.io/flexhaz/

BugReports https://github.com/queelius/flexhaz/issues

NeedsCompilation no

Author Alexander Towell [aut, cre] (ORCID: <https://orcid.org/0000-0001-6443-9897>)

Maintainer Alexander Towell <queelius@gmail.com>

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

Date/Publication 2026-10-28 09:52:06 UTC

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

RemoteRef HEAD

RemoteSha d743d3b867817431a43840a879310bb6ac28a639

Contents

assumptions.dfr_dist . . . 2
cdf.dfr_dist . . . 3
cum_haz . . . 3
cum_haz.dfr_dist . . . 4
density.dfr_dist . . . 4
dfr_dist . . . 5
dfr_exponential . . . 6
dfr_gompertz . . . 7
dfr_loglogistic . . . 8
dfr_weibull . . . 10
diagnostics . . . 11
distributions . . . 11
fit.dfr_dist . . . 12
hazard.dfr_dist . . . 13
hess_loglik.dfr_dist . . . 14
inv_cdf.dfr_dist . . . 14
is_dfr_dist . . . 15
loglik.dfr_dist . . . 15
params.dfr_dist . . . 16
plot.dfr_dist . . . 16
print.dfr_dist . . . 18
qqplot_residuals . . . 19
residuals.dfr_dist . . . 20
sampler.dfr_dist . . . 21
score.dfr_dist . . . 22
sup.dfr_dist . . . 22
surv.dfr_dist . . . 23

Index **24**

`assumptions.dfr_dist` *Retrieve the assumptions a DFR distribution makes about the data*

Description

Returns a list of assumptions that the dynamic failure rate distribution model makes about the underlying data and process.

Usage

```
## S3 method for class 'dfr_dist'
assumptions(model, ...)
```

Arguments

`model`	A `dfr_dist` object
`...`	Additional arguments (ignored)

Value

A character vector of model assumptions

`cdf.dfr_dist` *Method for obtaining the cdf of a* `dfr_dist` *object.*

Description

Method for obtaining the cdf of a `dfr_dist` object.

Usage

```
## S3 method for class 'dfr_dist'
cdf(x, ...)
```

Arguments

`x`	The object to obtain the cdf of.
`...`	Additional arguments to pass into the `cum_haz` constructor.

Value

A function that computes the cdf of the distribution. It accepts `t`, the time at which to compute the cdf, `par`, the parameters of the distribution, `log.p` argument that determines whether to compute the log of the cdf, `lower.limit`, whether to compute the lower limit (F(t)) or upper limit (S(t) = 1-F(t)). Finally, it passes any additional arguments `...` to the `rate` function of the `dfr_dist` object `x`.

`cum_haz` *Method for obtaining the cumulative hazard function of an object.*

Description

Method for obtaining the cumulative hazard function of an object.

Usage

```
cum_haz(x, ...)
```

Arguments

`x`	The object to obtain the cumulative hazard function of.
`...`	Additional arguments to pass.

Value

A function that computes the cumulative hazard function of the distribution.

cum_haz.dfr_dist *Method for obtaining the cumulative hazard function of a* dfr_dist *object.*

Description

Method for obtaining the cumulative hazard function of a dfr_dist object.

Usage

```
## S3 method for class 'dfr_dist'
cum_haz(x, ...)
```

Arguments

x	The object to obtain the cumulative hazard function of.
...	Additional arguments to pass into the integrate function (only used when no analytical cum_haz_rate is provided).

Details

If the dfr_dist object has an analytical cum_haz_rate function, that is used directly for fast, exact computation. Otherwise, numerical integration of the hazard function is performed.

Value

A function that computes the cumulative hazard H(t) of the distribution. It accepts t, the time at which to compute the cumulative hazard, and par, the parameters of the distribution. If par is NULL, then the parameters of the dfr_dist object x are used. Finally, it passes any additional arguments ... to the rate function.

density.dfr_dist *Method for obtaining the density (pdf) of a* dfr_dist *object.*

Description

Method for obtaining the density (pdf) of a dfr_dist object.

Usage

```
## S3 method for class 'dfr_dist'
density(x, ...)
```

Arguments

x	The object to obtain the density of.
...	Additional arguments to pass.

Value

A function that computes the density of the distribution. It accepts t, the time at which to compute the density, par is the parameters of the distribution, and log determines whether to compute the log of the density. Finally, it passes any additional arguments ... to the rate function of the dfr_dist object x.

---

dfr_dist *Constructor for* dfr_dist *objects*

Description

We assume that the hazard rate is a function of time and any other predictors. We also assume that integrate(rate(t), 0, Inf) = infinity and that the support is (0, Inf).

Usage

```
dfr_dist(
  rate,
  par = NULL,
  ob_col = "t",
  delta_col = "delta",
  cum_haz_rate = NULL,
  score_fn = NULL,
  hess_fn = NULL
)
```

Arguments

rate	A function that computes the hazard rate at time t.
par	The parameters of the distribution. Defaults to NULL, which means that the parameters are unknown.
ob_col	The column name for observation times in data frames. Defaults to "t". Inference methods require finite, non-negative numeric times and reject missing values.
delta_col	The column name for event indicators in data frames. Uses standard survival analysis convention: 1 = event observed (exact), 0 = right-censored, -1 = left-censored. Defaults to "delta". A missing column denotes all exact observations. An existing column must contain non-missing numeric codes or logical event indicators; other values are rejected.

`cum_haz_rate` Optional analytical cumulative hazard function H(t, par). If provided, used for faster exact cumulative hazard computation instead of numerical integration. Should return the integral of rate from 0 to t.

`score_fn` Optional score function (gradient of log-likelihood). Signature: score_fn(df, par, ob_col, delta_col, ...) returning a numeric vector. The ob_col and delta_col arguments indicate which columns in df contain observation times and event indicators. If NULL, falls back to numerical gradient via numDeriv::grad. Analytical score functions that only handle delta in {0, 1} are automatically bypassed when left-censored data (delta = -1) is present.

`hess_fn` Optional Hessian function (second derivatives of log-likelihood). Signature: hess_fn(df, par, ob_col, delta_col, ...) returning a matrix. The ob_col and delta_col arguments indicate which columns in df contain observation times and event indicators. If NULL, falls back to numerical Hessian via numDeriv::hessian. Analytical Hessian functions that only handle delta in {0, 1} are automatically bypassed when left-censored data (delta = -1) is present.

Value

A `dfr_dist` object that inherits from `likelihood_model`.

`dfr_exponential` *Exponential Distribution (Constant Hazard)*

Description

Creates a DFR distribution with constant failure rate (exponential). The exponential distribution is "memoryless" - the hazard does not depend on time, making it appropriate for random failures unrelated to age.

Usage

```
dfr_exponential(lambda = NULL)
```

Arguments

`lambda` Rate parameter (failure rate). If NULL, must be provided when calling methods or fitting. Must be positive.

Details

The exponential distribution has:

- Hazard: $h(t) = \lambda$
- Cumulative hazard: $H(t) = \lambda t$
- Survival: $S(t) = e^{-\lambda t}$
- Mean time to failure: $1/\lambda$

Value

A `dfr_dist` object of class `c("dfr_exponential", "dfr_dist", ...)` with analytical rate, cumulative hazard, and score function.

Reliability Interpretation

Use exponential for:

- Electronic components during useful life (random failures)
- Systems with redundancy where failures are independent
- As a baseline model to test against more complex alternatives

Examples

```
# Component with MTBF of 1000 hours (lambda = 0.001)
comp <- dfr_exponential(lambda = 0.001)

# Survival probability at 500 hours
S <- surv(comp)
S(500)  # ~60.6%

# Fit to failure data
set.seed(42)
failures <- data.frame(t = rexp(50, rate = 0.001), delta = 1)
solver <- fit(comp)
result <- solver(failures, par = c(0.002))
coef(result)  # Should be close to 0.001
```

`dfr_gompertz` *Gompertz Distribution (Exponential Growth Hazard)*

Description

Creates a DFR distribution with Gompertz hazard function. The Gompertz models exponentially increasing failure rate, often used for biological aging and wear-out processes that accelerate over time.

Usage

```
dfr_gompertz(a = NULL, b = NULL)
```

Arguments

a	Initial hazard rate at t=0. Must be positive.
b	Growth rate of the hazard. Must be positive.

Details

The Gompertz distribution has:

- Hazard: $h(t) = a \cdot e^{bt}$
- Cumulative hazard: $H(t) = (a/b)(e^{bt} - 1)$
- Survival: $S(t) = \exp(-(a/b)(e^{bt} - 1))$

Value

A `dfr_dist` object of class `c("dfr_gompertz", "dfr_dist", ...)` with analytical rate, cumulative hazard, and score function.

Reliability Interpretation

Use Gompertz for:

- Aging systems where failure rate grows exponentially
- Biological mortality (human lifespans)
- Corrosion/degradation with accelerating kinetics

When b is small, Gompertz approximates exponential early in life. As b increases, wear-out acceleration becomes more pronounced.

Examples

```
# Aging system: initial hazard 0.001, doubling every 1000 hours
# b = log(2)/1000 gives doubling time of 1000
system <- dfr_gompertz(a = 0.001, b = log(2)/1000)

# Hazard at various ages
h <- hazard(system)
h(0)      # 0.001 (initial)
h(1000)   # 0.002 (doubled)
h(2000)   # 0.004 (quadrupled)

# Survival probability
S <- surv(system)
S(5000)   # probability of surviving 5000 hours
```

---

dfr_loglogistic *Log-Logistic Distribution (Non-Monotonic Hazard)*

Description

Creates a DFR distribution with log-logistic hazard function. The log-logistic has a non-monotonic hazard that increases then decreases, useful for modeling processes with an initial risk that diminishes.

Usage

```
dfr_loglogistic(alpha = NULL, beta = NULL)
```

Arguments

alpha	Scale parameter. Median lifetime when beta > 1.
beta	Shape parameter. Controls hazard shape: beta <= 1: monotonically decreasing hazard beta > 1: hazard increases to a peak then decreases

Details

The log-logistic distribution has:

- Hazard: $h(t) = \frac{(\beta/\alpha)(t/\alpha)^{\beta-1}}{1+(t/\alpha)^{\beta}}$
- Cumulative hazard: $H(t) = \log(1 + (t/\alpha)^{\beta})$
- Survival: $S(t) = \frac{1}{1+(t/\alpha)^{\beta}}$
- Median: α (when beta > 1)

Value

A `dfr_dist` object of class `c("dfr_loglogistic", "dfr_dist", ...)` with analytical rate, cumulative hazard, and score function.

Reliability Interpretation

The log-logistic is useful when:

- Initial failures decrease after screening period
- Risk peaks early then declines (therapy response)
- Hazard is not monotonic throughout lifetime

The cumulative hazard has a closed form and is provided analytically.

Examples

```
# Component with peak hazard around t = alpha
comp <- dfr_loglogistic(alpha = 1000, beta = 2)

# Non-monotonic hazard
h <- hazard(comp)
h(500)   # increasing phase
h(1000)  # near peak
h(2000)  # decreasing phase

# Survival function
S <- surv(comp)
S(1000)  # 50% survival at median (alpha)
```

`dfr_weibull` *Weibull Distribution (Power-Law Hazard)*

Description

Creates a DFR distribution with Weibull hazard function. The Weibull is extremely versatile: it can model increasing (wear-out), decreasing (infant mortality), or constant (exponential) failure rates.

Usage

```
dfr_weibull(shape = NULL, scale = NULL)
```

Arguments

`shape`	Shape parameter (k). Controls hazard behavior: k < 1: decreasing hazard (infant mortality) k = 1: constant hazard (exponential) k > 1: increasing hazard (wear-out)
`scale`	Scale parameter (sigma). Controls time scale.

Details

The Weibull distribution has:

- Hazard: $h(t) = (k/\sigma)(t/\sigma)^{k-1}$
- Cumulative hazard: $H(t) = (t/\sigma)^k$
- Survival: $S(t) = e^{-(t/\sigma)^k}$
- Characteristic life (63.2% failure): σ

Value

A `dfr_dist` object of class `c("dfr_weibull", "dfr_dist", ...)` with analytical rate, cumulative hazard, and score function.

Reliability Interpretation

- Shape < 1: Infant mortality (burn-in failures, defects)
- Shape = 1: Random failures (reduces to exponential)
- Shape = 2: Rayleigh distribution (linear hazard increase)
- Shape > 2: Accelerating wear-out (fatigue, corrosion)

B-Life Calculation

The B10 life (10% failure quantile) is commonly used in reliability: $B10 = \sigma \cdot (-\log(0.9))^{1/k}$

Examples

```
# Bearing with wear-out failure (shape > 1)
bearing <- dfr_weibull(shape = 2.5, scale = 50000)

# Hazard increases with time
h <- hazard(bearing)
h(10000)  # hazard at 10k hours
h(40000)  # much higher at 40k hours

# B10 life calculation
Q <- inv_cdf(bearing)
B10 <- Q(0.10)  # 10% failure quantile

# Fit to test data with right-censoring
set.seed(123)
test_data <- data.frame(
  t = pmin(rweibull(100, shape = 2.5, scale = 50000), 30000),
  delta = as.integer(rweibull(100, shape = 2.5, scale = 50000) <= 30000)
)
solver <- fit(dfr_weibull())
result <- solver(test_data, par = c(2, 40000))
coef(result)
```

`diagnostics` *Diagnostic Methods for DFR Distributions*

Description

Methods for assessing model fit and visualizing survival distributions.

`distributions` *Classic Survival Distribution Constructors*

Description

Convenient constructors for commonly-used survival distributions. Each provides the complete specification (rate, cum_haz_rate, score_fn, and where practical, hess_fn) for optimal performance.

Left-Censoring Note

The analytical score and Hessian functions provided by these constructors assume event indicators in {0, 1} (right-censored and exact observations). For left-censored data (delta = -1), these functions are not applicable and the package automatically falls back to numerical differentiation via `numDeriv::grad` and `numDeriv::hessian` through the log-likelihood, which handles all censoring types correctly.

`fit.dfr_dist` *MLE solver for dfr_dist objects*

Description

Returns a solver function that fits a dfr_dist model to survival data using maximum likelihood estimation. The solver uses gradient-based optimization (BFGS by default) with analytical or numerical gradients.

Usage

```
## S3 method for class 'dfr_dist'
fit(object, ...)
```

Arguments

`object`	A `dfr_dist` object
`...`	Additional arguments passed to the log-likelihood, score, and Hessian constructors (e.g., integration parameters for cum_haz)

Details

The solver returns a `fisher_mle` object (from likelihood.model) containing:

- Parameter estimates
- Log-likelihood value at MLE
- Variance-covariance matrix (from Hessian)
- Convergence status

Use methods like `coef()`, `vcov()`, `confint()`, `summary()` on the result.

Value

A solver function that accepts:

- `df`: Data frame with observation times and censoring indicators
- `par`: Initial parameter values (uses object's params if NULL)
- `method`: Optimization method (default "BFGS")
- `control`: Control parameters for optim()
- `...`: Additional arguments passed to likelihood functions

Examples

```
# Exponential distribution
exp_dist <- dfr_dist(
  rate = function(t, par, ...) rep(par[1], length(t)),
  par = c(lambda = 1)
)

# Simulate data
set.seed(42)
df <- data.frame(t = rexp(100, rate = 2), delta = 1)

# Fit model
solver <- fit(exp_dist)
result <- solver(df, par = c(1))
summary(result)
confint(result)
```

---

hazard.dfr_dist *Method for obtaining the hazard function of a* `dfr_dist` *object.*

Description

Method for obtaining the hazard function of a `dfr_dist` object.

Usage

```
## S3 method for class 'dfr_dist'
hazard(x, ...)
```

Arguments

`x`	The object to obtain the hazard function of.
`...`	Additional arguments to pass into the `rate` function.

Value

A function that computes the hazard function of the distribution. It accepts `t`, the time at which to compute the hazard function, and `par`, the parameters of the distribution. If `par` is `NULL`, then the parameters of the `dfr_dist` object `x` are used. It also accepts a `log` argument that determines whether to compute the log of the hazard function. Finally, it passes any additional arguments to the `rate` function of the `dfr_dist` object `x`.

hess_loglik.dfr_dist *Hessian of log-likelihood for dfr_dist*

Description

Returns a function that computes the Hessian matrix of the log-likelihood. Uses user-provided Hessian function if available, otherwise falls back to numerical differentiation via numDeriv::hessian.

Usage

```
## S3 method for class 'dfr_dist'
hess_loglik(model, ...)
```

Arguments

model	A dfr_dist object
...	Additional arguments passed to loglik

Value

A function that computes the Hessian matrix

inv_cdf.dfr_dist *Method for obtaining the quantile (inverse cdf) of an object.*

Description

Method for obtaining the quantile (inverse cdf) of an object.

Usage

```
## S3 method for class 'dfr_dist'
inv_cdf(x, ...)
```

Arguments

x	The object to obtain the inverse cdf of.
...	Additional arguments to pass into cdf constructor.

Value

A function that computes the quantile of the distribution. It accepts p, the probability at which to compute the quantile, par, the parameters of the distribution, and ..., any additional arguments to pass into the constructed cdf.

is_dfr_dist *Function for determining whether an object is a* dfr_dist *object.*

Description

Function for determining whether an object is a dfr_dist object.

Usage

```
is_dfr_dist(x)
```

Arguments

x	The object to test.

Value

A logical value indicating whether x is a dfr_dist object.

loglik.dfr_dist *Log-likelihood method for* dfr_dist *objects*

Description

Returns a function that computes the log-likelihood of the data given the distribution parameters. The log-likelihood for survival data is:

Usage

```
## S3 method for class 'dfr_dist'
loglik(model, ...)
```

Arguments

model	The dfr_dist object
...	Additional arguments to pass to the hazard and cumulative hazard

Details

For exact observations (uncensored): log(f(t)) = log(h(t)) - H(t) For right-censored observations: log(S(t)) = -H(t) For left-censored observations: log(F(t)) = log(1 - exp(-H(t)))

where h(t) is the hazard function, H(t) is the cumulative hazard, f(t) = h(t)*S(t) is the pdf, and S(t) = exp(-H(t)) is the survival function.

Value

A function that computes the log-likelihood. It accepts: - `df`: A data frame with observation times and censoring indicators (delta: 1 = exact, 0 = right-censored, -1 = left-censored) - `par`: The parameters of the distribution - `...`: Additional arguments passed to internal functions

---

`params.dfr_dist` *Method for obtaining the parameters of a* `dfr_dist` *object.*

Description

Method for obtaining the parameters of a `dfr_dist` object.

Usage

```
## S3 method for class 'dfr_dist'
params(x, ...)
```

Arguments

`x`	The object to obtain the parameters of.
`...`	Additional arguments (unused).

Value

The parameters of the distribution.

`plot.dfr_dist` *Plot DFR Distribution Functions*

Description

Visualizes the survival, hazard, or cumulative hazard function of a DFR distribution. Optionally overlays empirical estimates from data.

Usage

```
## S3 method for class 'dfr_dist'
plot(
  x,
  data = NULL,
  par = NULL,
  what = c("survival", "hazard", "cumhaz"),
  xlim = NULL,
  n = 200,
  add = FALSE,
```

```
  col = "black",
  lwd = 2,
  empirical = TRUE,
  empirical_col = "steelblue",
  ...
)
```

Arguments

x	A dfr_dist object
data	Optional data frame with survival data for empirical overlay
par	Parameter vector. If NULL, uses object's stored parameters.
what	Which function to plot: **"survival"** S(t) = exp(-H(t)) **"hazard"** h(t) - instantaneous failure rate **"cumhaz"** H(t) - cumulative hazard
xlim	x-axis limits. If NULL, determined from data or defaults to c(0, 10).
n	Number of points for smooth curve (default 200)
add	If TRUE, add to existing plot
col	Line color for theoretical curve
lwd	Line width for theoretical curve
empirical	If TRUE and data provided, overlay Kaplan-Meier estimate
empirical_col	Color for empirical curve
...	Additional arguments passed to plot()

Details

When empirical = TRUE and data is provided, overlays:

- For survival: Kaplan-Meier estimate (step function)
- For cumhaz: Nelson-Aalen estimate (step function)
- For hazard: Kernel-smoothed hazard estimate

Value

Invisibly returns the plotted values as a list with elements t (time points) and y (function values).

Examples

```
# Plot survival function for Weibull distribution
weib <- dfr_weibull(shape = 2, scale = 5)
plot(weib, what = "survival", xlim = c(0, 10))

# Overlay hazard functions for different shapes
plot(weib, what = "hazard", xlim = c(0, 10), col = "blue")
weib_k1 <- dfr_weibull(shape = 1, scale = 5)  # Exponential
```

```
plot(weib_k1, what = "hazard", add = TRUE, col = "green")
weib_k3 <- dfr_weibull(shape = 3, scale = 5)  # Steeper wear-out
plot(weib_k3, what = "hazard", add = TRUE, col = "red")
legend("topleft", c("k=2", "k=1 (exp)", "k=3"),
       col = c("blue", "green", "red"), lwd = 2)

# Compare fitted model to data
set.seed(123)
true_weib <- dfr_weibull(shape = 2.5, scale = 10)
sim_data <- data.frame(t = sampler(true_weib)(100), delta = 1)
solver <- fit(dfr_weibull())
result <- solver(sim_data, par = c(2, 8))
fitted_weib <- dfr_weibull(shape = coef(result)[1], scale = coef(result)[2])
plot(fitted_weib, data = sim_data, what = "survival",
     xlim = c(0, max(sim_data$t)), empirical = TRUE)
```

`print.dfr_dist` *Print method for* `dfr_dist` *objects.*

Description

Print method for `dfr_dist` objects.

Usage

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

Arguments

`x`	The `dfr_dist` object to print.
`...`	Additional arguments (not used)

Value

Invisibly returns `x`.

`qqplot_residuals` *Q-Q Plot for Cox-Snell Residuals*

Description

Creates a Q-Q plot comparing Cox-Snell residuals to the theoretical Exp(1) distribution. A good fit shows points along the diagonal.

Usage

```
qqplot_residuals(object, data, par = NULL, add_line = TRUE, ...)
```

Arguments

`object`	A `dfr_dist` object
`data`	Data frame with survival data
`par`	Parameter vector. If NULL, uses object's stored parameters.
`add_line`	If TRUE, adds reference line at y = x
`...`	Additional arguments passed to residuals and qqplot

Details

Cox-Snell residuals r_i = H(t_i) should follow an Exp(1) distribution if the model is correctly specified. Departure from the diagonal line in the Q-Q plot indicates model misspecification:

- Points above the line: observations failed earlier than expected
- Points below the line: observations survived longer than expected
- Systematic curvature: wrong distributional form

Value

Invisibly returns Cox-Snell residuals

Examples

```
# Check fit of exponential model
set.seed(42)
df <- data.frame(t = rexp(100, rate = 0.5), delta = 1)
exp_dist <- dfr_exponential(lambda = 0.5)

qqplot_residuals(exp_dist, df)

# Check fit with wrong model (Weibull data, exponential fit)
df_weib <- data.frame(t = sampler(dfr_weibull(shape = 2, scale = 5))(100), delta = 1)
exp_fit <- dfr_exponential(lambda = 1 / mean(df_weib$t))  # Moment estimate
qqplot_residuals(exp_fit, df_weib)  # Should show systematic departure
```

`residuals.dfr_dist` *Residuals for DFR Distribution Fits*

Description

Computes residuals for assessing model fit of a DFR distribution to survival data. Cox-Snell residuals should follow Exp(1) if the model is correct. Martingale residuals identify observations poorly fit by the model.

Usage

```
## S3 method for class 'dfr_dist'
residuals(object, data, par = NULL, type = c("cox-snell", "martingale"), ...)
```

Arguments

`object`	A `dfr_dist` object
`data`	Data frame with survival data (must have time column and optionally delta column for censoring indicator)
`par`	Parameter vector. If NULL, uses object's stored parameters.
`type`	Type of residual: **"cox-snell"** H(t_i) - should follow Exp(1) if model correct **"martingale"** delta_i - H(t_i) - useful for identifying outliers
`...`	Additional arguments passed to cum_haz

Details

Cox-Snell residuals are defined as r_i = H(t_i), the cumulative hazard evaluated at the observation time. If the fitted model is correct, these should follow an Exp(1) distribution (possibly censored).

Martingale residuals are defined as M_i = delta_i - H(t_i), where delta_i is the event indicator. They sum to zero and can identify observations that are poorly fit. Large positive values indicate observations that failed "too early" relative to the model; large negative values indicate observations that survived "too long".

Value

Numeric vector of residuals, same length as data

Diagnostic Use

- Q-Q plot of Cox-Snell residuals against Exp(1) to check overall fit
- Plot Martingale residuals vs. covariates to check functional form
- Plot Martingale residuals vs. fitted values to check homogeneity

Examples

```
# Fit exponential to simulated data
set.seed(42)
df <- data.frame(t = rexp(100, rate = 0.5), delta = 1)
exp_dist <- dfr_exponential(lambda = 0.5)

# Cox-Snell residuals
cs_resid <- residuals(exp_dist, df, type = "cox-snell")

# Should follow Exp(1) - check with Q-Q plot
qqplot(qexp(ppoints(100)), sort(cs_resid),
       main = "Cox-Snell Residuals Q-Q Plot",
       xlab = "Theoretical Exp(1)", ylab = "Sample")
abline(0, 1, col = "red")

# Martingale residuals
mart_resid <- residuals(exp_dist, df, type = "martingale")
summary(mart_resid)  # Should sum to approximately 0
```

---

sampler.dfr_dist *Sampling function for* dfr_dist *objects.*

Description

Uses inverse CDF sampling: generates uniform random values and transforms them through the quantile function (inverse CDF).

Usage

```
## S3 method for class 'dfr_dist'
sampler(x, ...)
```

Arguments

x	The object to obtain the sampler of.
...	Additional arguments to pass into the inverse CDF constructor.

Value

A function that samples from the distribution. It accepts n, the number of samples to take, par, the parameters of the distribution, and ..., additional arguments passed to the quantile function.

score.dfr_dist *Score function (gradient of log-likelihood) for dfr_dist*

Description

Returns a function that computes the score (gradient of log-likelihood) with respect to parameters. Uses user-provided score function if available, otherwise falls back to numerical differentiation via numDeriv::grad.

Usage

```
## S3 method for class 'dfr_dist'
score(model, ...)
```

Arguments

model	A dfr_dist object
...	Additional arguments passed to loglik

Value

A function that computes the score vector

sup.dfr_dist *Method for retrieving the support of an object* x.

Description

Method for retrieving the support of an object x.

Usage

```
## S3 method for class 'dfr_dist'
sup(x, ...)
```

Arguments

x	The object to obtain the support of.
...	Additional arguments to pass.

Value

A support object for x, an interval (0,Inf).

surv.dfr_dist *Method for obtaining the survival function of a* dfr_dist *object.*

Description

Method for obtaining the survival function of a dfr_dist object.

Usage

```
## S3 method for class 'dfr_dist'
surv(x, ...)
```

Arguments

x	The object to obtain the survival function of.
...	Additional arguments to pass into the cum_haz constructor.

Value

A function that computes the survival function of the distribution. It accepts t, the time at which to compute the survival, par, the parameters of the distribution, log.p argument that determines whether to compute the log of the survival, and it passes any additional arguments into the rate function of the dfr_dist object x.

Index

∗ **dfr_dist**
diagnostics, 11
∗ **distributions**
distributions, 11

assumptions.dfr_dist, 2

cdf.dfr_dist, 3
cum_haz, 3
cum_haz.dfr_dist, 4

density.dfr_dist, 4
dfr_dist, 5
dfr_exponential, 6
dfr_gompertz, 7
dfr_loglogistic, 8
dfr_weibull, 10
diagnostics, 11
distributions, 11

fit.dfr_dist, 12

hazard.dfr_dist, 13
hess_loglik.dfr_dist, 14

inv_cdf.dfr_dist, 14
is_dfr_dist, 15

loglik.dfr_dist, 15

params.dfr_dist, 16
plot.dfr_dist, 16
print.dfr_dist, 18

qqplot_residuals, 19

residuals.dfr_dist, 20

sampler.dfr_dist, 21
score.dfr_dist, 22
sup.dfr_dist, 22
surv.dfr_dist, 23