Package 'likelihood.model'

May 24, 2026

Type Package

Title Likelihood-Based Statistical Inference in the Fisherian Tradition

Version 1.0.1

Description Facilitates building likelihood models in the Fisherian tradition following Richard Royall (1997, ISBN:978-0412044113) ``Statistical Evidence: A Likelihood Paradigm''. Defines generic methods for working with likelihoods (loglik(), score(), hess_loglik(), fim()) and provides functions for pure likelihood-based inference (support(), relative_likelihood(), likelihood_interval(), profile_loglik()).

License MIT + file LICENSE

Encoding UTF-8

Depends algebraic.mle (>= 2.0.0)

Imports algebraic.dist, generics, stats, numDeriv, boot

URL https://github.com/queelius/likelihood.model, https://queelius.github.io/likelihood.model/

BugReports https://github.com/queelius/likelihood.model/issues

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

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

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

Maintainer Alexander Towell <lex@metafunctor.com>

Repository CRAN

Date/Publication 2026-05-24 09:20:02 UTC

Contents

`likelihood.model-package`

likelihood.model: Likelihood-Based Inference in the Fisherian Tradition

---

Description

The `likelihood.model` package provides a framework for likelihood-based inference. The package is organized in layers:

Core Concept (`core-generics.R`): The `likelihood_model` "concept" – an abstract interface that any model can implement. At minimum, implement `loglik()`. Optionally provide `score()` and `hess_loglik()` for analytical derivatives; defaults use numerical differentiation via `numDeriv`.

Core Infrastructure:

- fisher_mle / fisher_boot: Result objects from MLE fitting, with methods for `coef()`, `vcov()`, `confint()`, `algebraic.mle::se()`, `stats::AIC()`, `stats::BIC()`, `summary()`.
- `fit()`: Default MLE solver using `optim()`. Models can override this with closed-form solutions (see exponential_lifetime for an example).
- Fisherian inference: `support()`, `relative_likelihood()`, `likelihood_interval()`, `profile_loglik()`, `evidence()` – pure likelihood-based inference without probability statements.
- `lrt()`: Likelihood ratio test for nested models.

Example Implementation: exponential_lifetime: Exponential with right-censoring support. Demonstrates closed-form MLE (no optim needed), analytical FIM, and `rdata()` for Monte Carlo validation.

For contribution-based models with heterogeneous observation types, see the companion package **likelihood.contr**.

Author(s)

Maintainer: Alexander Towell <lex@metafunctor.com> (ORCID)

See Also

Useful links:

- https://github.com/queelius/likelihood.model
- https://queelius.github.io/likelihood.model/
- Report bugs at https://github.com/queelius/likelihood.model/issues

---

`assumptions` *Retrieve the assumptions the likelihood model makes about the data.*

Description

Retrieve the assumptions the likelihood model makes about the data.

Usage

```
assumptions(model, ...)
```

Arguments

`model`	The likelihood model
`...`	Additional arguments

Value

A list of assumptions

`assumptions.exponential_lifetime` *Assumptions for exponential_lifetime*

Description

Assumptions for exponential_lifetime

Usage

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

Arguments

`model`	An exponential_lifetime model
`...`	Additional arguments (ignored)

Value

Character vector of assumptions

bias.fisher_boot *Compute bootstrap bias estimate*

Description

Estimates bias from bootstrap replicates: bias = mean(bootstrap estimates) - original estimate

Usage

```
## S3 method for class 'fisher_boot'
bias(x, theta = NULL, ...)
```

Arguments

x	A fisher_boot object
theta	Ignored (for compatibility)
...	Additional arguments (ignored)

Value

Bias estimate vector

bias.fisher_mle *Bias for fisher_mle*

Description

Estimates the bias of the MLE. Without a model and true parameter value, returns zeros (asymptotic bias is zero under regularity conditions). When both `theta` and `model` are provided, performs a Monte Carlo simulation study to estimate finite-sample bias.

Usage

```
## S3 method for class 'fisher_mle'
bias(x, theta = NULL, model = NULL, n_sim = 1000, ...)
```

Arguments

x	A fisher_mle object
theta	True parameter value (for simulation studies)
model	A likelihood model with `rdata` and `fit` methods (optional)
n_sim	Number of Monte Carlo replicates (default 1000)
...	Additional arguments (ignored)

Value

Bias estimate vector

coef.fisher_mle *Extract coefficients from fisher_mle object*

Description

Extract coefficients from fisher_mle object

Usage

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

Arguments

object	A fisher_mle object
...	Additional arguments (ignored)

Value

Named numeric vector of parameter estimates

confint.fisher_boot *Confidence intervals from bootstrap*

Description

Computes bootstrap confidence intervals using various methods.

Usage

```
## S3 method for class 'fisher_boot'
confint(
  object,
  parm = NULL,
  level = 0.95,
  type = c("perc", "bca", "norm", "basic"),
  ...
)
```

Arguments

object	A fisher_boot object
parm	Parameter names or indices (NULL for all)
level	Confidence level (default 0.95)
type	Type of bootstrap CI: "perc", "bca", "norm", or "basic"
...	Additional arguments passed to boot::boot.ci

Value

Matrix with columns for lower and upper bounds

`confint.fisher_mle` *Compute confidence intervals for fisher_mle parameters*

Description

Computes asymptotic Wald confidence intervals based on the estimated variance-covariance matrix.

Usage

```
## S3 method for class 'fisher_mle'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`object`	A fisher_mle object
`parm`	Parameter names or indices (NULL for all)
`level`	Confidence level (default 0.95)
`...`	Additional arguments (ignored)

Value

Matrix with columns for lower and upper bounds

`deviance.fisher_mle` *Deviance for likelihood models*

Description

Computes the deviance, which is useful for model comparison.

Usage

```
## S3 method for class 'fisher_mle'
deviance(object, null_model = NULL, ...)
```

Arguments

`object`	A fisher_mle object
`null_model`	Optional reduced/null model for comparison
`...`	Additional arguments (ignored)

Details

When called with a single model, returns -2 * logL (the deviance relative to a saturated model).

When called with two models, returns the deviance difference: D = 2 * (logL_full - logL_reduced)

Under the null hypothesis that the reduced model is correct, D is asymptotically chi-squared with df = p_full - p_reduced.

Value

Deviance value

---

evidence *Likelihood-Based Evidence*

Description

Computes the strength of evidence for theta1 vs theta2: E(theta1, theta2) = logL(theta1) - logL(theta2)

Usage

```
evidence(model, ...)

## S3 method for class 'likelihood_model'
evidence(model, data, theta1, theta2, ...)
```

Arguments

`model`	The likelihood model
`...`	Additional arguments
`data`	Data frame for likelihood computation
`theta1`	First parameter value
`theta2`	Second parameter value

Details

Positive values favor theta1, negative values favor theta2.

Conventional interpretation (following Royall):

- |E| > log(8) ~ 2.08: Strong evidence
- |E| > log(32) ~ 3.47: Very strong evidence

Value

Evidence value (log likelihood ratio)

`exponential_lifetime` *Exponential lifetime model with right-censoring support*

Description

A likelihood model for the Exponential(lambda) distribution with optional right-censoring. This is a reference implementation demonstrating:

- **Closed-form MLE**: Overrides `fit()` to compute lambda_hat = d/T directly, bypassing `optim` entirely.
- **Analytical derivatives**: score, Hessian, and FIM in closed form.
- **Right-censoring**: Natural handling via the sufficient statistic (d, T) where d = number of exact observations and T = total time.
- `rdata()` **method**: For Monte Carlo validation and FIM estimation.

The log-likelihood is:

$$\ell(\lambda) = d \log \lambda - \lambda T$$

where d is the number of exact (uncensored) observations and T is the total observation time (sum of all times, whether censored or not).

Usage

```
exponential_lifetime(ob_col, censor_col = NULL)
```

Arguments

`ob_col`	The name of the column containing observation times.
`censor_col`	Optional column name indicating censoring status. When provided, values should be `"exact"` for uncensored observations and `"right"` for right-censored observations. When `NULL`, all observations are treated as exact.

Value

An `exponential_lifetime` likelihood model object

Examples

```
# Uncensored exponential data
model <- exponential_lifetime("t")
df <- data.frame(t = rexp(100, rate = 2))
mle <- fit(model)(df)
coef(mle)  # should be close to 2

# Right-censored data
model_c <- exponential_lifetime("t", censor_col = "status")
df_c <- data.frame(
  t = c(rexp(80, 2), rep(0.5, 20)),
```

```
  status = c(rep("exact", 80), rep("right", 20))
)
mle_c <- fit(model_c)(df_c)
coef(mle_c)
```

`fim` *Fisher information matrix method*

Description

This function calculates the Fisher information matrix (FIM), an expectation over the data-generating process (DGP). The FIM is a crucial concept in statistics because it provides information about the precision of estimates and the amount of information that data carries about an unknown parameter.

Usage

```
fim(model, ...)
```

Arguments

`model`	A likelihood model
`...`	Additional arguments

Details

FIM is a function of the parameters, and is used to compute the standard errors of the parameters. It is also used to compute the covariance matrix of the parameters, which is in turn used to compute standard errors of the parameters.

Additionally, FIM is used to compute the Cramer-Rao lower bound (CRLB), the inverse of the FIM. CRLB represents the lower limit of the variance that an unbiased estimator can attain. This is used to compute the asymptotic relative efficiency (ARE) of an estimator of the parameters, which is the ratio of the variance of the estimator to the CRLB.

The function computes FIM(x)(theta), the FIM of the likelihood model `x`, is based on the following formulas:

```
FIM(x)(theta) = E[-loglik_hessian(x)(ob,theta)]
FIM(x)(theta) = E[score(x)(ob,theta) %*% t(score(x)(ob,theta))]
```

where the expectation is taken with respect to ob ~ DGP. The first formula is the expected hessian of the log-likelihood function, and the second formula is the expected outer product of the score function. The two formulas are equivalent.

Value

Function that computes the FIM given a parameter vector and sample size

fim.exponential_lifetime *Fisher information matrix for exponential_lifetime*

Description

Returns the analytical FIM: n / lambda^2 (1x1 matrix).

Usage

```
## S3 method for class 'exponential_lifetime'
fim(model, ...)
```

Arguments

model	An exponential_lifetime model
...	Additional arguments (ignored)

Value

A function(theta, n_obs, ...) computing the FIM

fim.likelihood_model *Default FIM method using Monte Carlo estimation*

Description

Computes the Fisher information matrix by Monte Carlo simulation using the negative expected Hessian approach. For each of `n_samples` replicates, generates a single-observation dataset via `rdata`, computes `-hess_loglik(single_obs, theta)`, and averages. The result is `n_obs * I_1(theta)` where `I_1(theta)` is the per-observation FIM.

Usage

```
## S3 method for class 'likelihood_model'
fim(model, ...)
```

Arguments

model	A likelihood model
...	Additional arguments (currently unused; reserved for future extension at closure-construction time)

Details

This default requires the model to implement `rdata` and `hess_loglik` (or `loglik`, since `hess_loglik` falls back to numerical differentiation).

Extra arguments via `...` to the returned FIM function are forwarded to `rdata` only (i.e., they parameterize the data-generating process: censoring time, masking probability, observation functor, etc.) and are NOT passed to the likelihood evaluator `hess_loglik`. This honors the separation between the DGP layer and the likelihood layer: the likelihood is computed on data, not on the DGP knobs that produced it. Forwarding DGP kwargs into the likelihood layer was both an axiom violation (under masking condition C3 the likelihood does not depend on the masking probability) and a partial-matching footgun (a kwarg named `p` would collide with the formal `par`).

Value

Function that takes (theta, n_obs, n_samples, ...) and returns FIM matrix. The inner `...` is forwarded to `rdata` only.

`fisherian` *Fisherian Likelihood Inference*

Description

Functions for pure likelihood-based inference in the Fisherian tradition. These emphasize the likelihood function itself as the primary object of inference, without requiring probability statements about parameters.

`fisher_boot` *Bootstrap MLE Estimate*

Description

Creates a `fisher_boot` object representing bootstrap-based inference for maximum likelihood estimates.

Usage

```
fisher_boot(boot_result, original_mle)
```

Arguments

`boot_result`	Result from boot::boot()
`original_mle`	The original fisher_mle object

Value

An object of class `c("fisher_boot", "fisher_mle", "mle_fit", "boot")`

fisher_mle *Maximum Likelihood Estimate (Fisherian)*

Description

Creates a `fisher_mle` object representing a maximum likelihood estimate with methods for standard inference. This class emphasizes the Fisherian approach to likelihood-based inference.

Usage

```
fisher_mle(
  par,
  vcov = NULL,
  loglik_val,
  hessian = NULL,
  score_val = NULL,
  nobs = NULL,
  converged = TRUE,
  optim_result = NULL
)
```

Arguments

`par`	Numeric vector of parameter estimates (may be named)
`vcov`	Variance-covariance matrix of the estimates
`loglik_val`	Log-likelihood value at the MLE
`hessian`	Hessian matrix of the log-likelihood at the MLE
`score_val`	Optional score vector at the MLE (should be near zero)
`nobs`	Number of observations used in estimation
`converged`	Logical indicating if optimization converged
`optim_result`	Raw result from optim() for diagnostics

Value

An object of class `c("fisher_mle", "mle_fit")`

fit *Fit a model*

Description

Re-exported from **generics**. See `generics::fit()` for details.

fit.exponential_lifetime

Closed-form MLE for exponential_lifetime

Description

Computes the MLE directly as lambda_hat = d / T, bypassing `optim`. This demonstrates that specialized models can provide exact solutions.

Usage

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

Arguments

`object`	An exponential_lifetime model
`...`	Additional arguments (ignored)

Value

A solver function that takes (df, par, ...) and returns a `fisher_mle` object. The `par` argument is accepted but ignored since the MLE is computed in closed form.

fit.likelihood_model *Default MLE solver for subclasses of* `likelihood_model`.

Description

Note that `likelihood_model` is not a class, but a concept, that other likelihood models implement. They should add `likelihood_model` to their class definition, and then they can use this function to compute the MLE.

Usage

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

Arguments

`object`	The `likelihood_model` object
`...`	Additional arguments to pass into the likelihood model's `loglik`, `score`, and `hess_loglik` constructors.

Details

This function uses the `optim` function to find the MLE of the parameters of a likelihood model. It uses the `loglik` and `score` methods to compute the log-likelihood and score function, respectively.

There are a few interesting options for the `control` argument:

- `method`: The optimization method to use. The default is `Nelder-Mead`, which is a derivative-free method. Other options like include `BFGS` are gradient-based methods, which may be preferable if you provide a `score` function (rather than using the default finite-difference). There is also the `SANN` method, which is a simulated annealing method. This method is useful for multimodal likelihood functions, where the MLE may be sensitive to the initial guess. The `SANN` method is a more global method, but it is slower and may require some tweaking. Regardless, if you do use `SANN`, you should follow it up with a local search method like `Nelder-Mead` to refine the solution.

Value

An MLE solver (function) that returns an MLE object and accepts as arguments:

- `df`: The data frame
- `par`: The initial guess for the parameters
- `control`: Control parameters for the optimization algorithm
- `...`: Additional arguments to pass into the likelihood model's constructed functions from `loglik`, `score`, and `hess_loglik`.

`hess_loglik` *Hessian of log-likelihood method*

Description

This function returns the hessian of the log-likelihood function of a model

Usage

```
hess_loglik(model, ...)
```

Arguments

`model`	The likelihood model
`...`	Additional arguments

Value

A function to compute the hessian of the log-likelihood given a data frame and parameters

hess_loglik.exponential_lifetime

Hessian of the log-likelihood for exponential_lifetime

Description

Returns a function computing the 1x1 Hessian: -d/lambda^2.

Usage

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

Arguments

model	An exponential_lifetime model
...	Additional arguments (ignored)

Value

A function(df, par, ...) computing the Hessian matrix

hess_loglik.likelihood_model

Default method to compute the hessian of the log-likelihood.

Description

In case a hess_loglik method is not provided, this function will be used. It computes the hessian of the log-likelihood function using numerical differentiation.

Usage

```
## S3 method for class 'likelihood_model'
hess_loglik(model, control = list(), ...)
```

Arguments

model	The likelihood model
control	Custom arguments to pass to numDeriv::hessian.
...	Additional arguments (to pass into loglik)

Value

A function(df, par, ...) that computes the Hessian matrix of the log-likelihood evaluated at par given data df.

is_likelihood_model *Check if an object is a likelihood model*

Description

Tests whether an object satisfies the `likelihood_model` concept by checking if `"likelihood_model"` is in its class hierarchy. To be a likelihood model, at a minimum the object must implement `loglik()`. For optimal results, it may also provide `score()` and `hess_loglik()`.

Usage

```
is_likelihood_model(x)
```

Arguments

`x`	An object to test

Value

Logical indicating whether x is a likelihood model

likelihood_interval *Likelihood Interval*

Description

Computes the likelihood interval for a parameter: `LI(k) = {theta : R(theta) >= 1/k} = {theta : S(theta) >= -log(k)}`

Usage

```
likelihood_interval(x, ...)

## S3 method for class 'fisher_mle'
likelihood_interval(x, data, model, k = 8, param = NULL, ...)
```

Arguments

`x`	A fisher_mle object
`...`	Additional arguments passed to optimization
`data`	Data frame used for likelihood computation
`model`	The likelihood model used for fitting
`k`	Likelihood ratio cutoff (default 8, giving 1/8 interval)
`param`	Index or name of parameter for profile interval (NULL for all)

Details

Unlike confidence intervals, likelihood intervals make no probability statements about the parameter. They simply identify the set of parameter values that are well-supported by the data.

Common choices for k:

- k = 8: 1/8 likelihood interval (~95% CI equivalent)
- k = 15: 1/15 likelihood interval (~99% CI equivalent)
- k = 32: 1/32 likelihood interval (~99.9% CI equivalent)

For multivariate parameters, specify `param` to get a profile likelihood interval for that parameter (profiling over the others).

Value

Matrix with lower and upper bounds for each parameter

`loglik` *Log-likelihood method*

Description

This function returns the log-likelihood function of a model

Usage

```
loglik(model, ...)
```

Arguments

`model`	The likelihood model
`...`	Additional arguments

Value

A log-likelihood function to compute the log-likelihood given a data frame and parameters.

loglik.exponential_lifetime *Log-likelihood for exponential_lifetime*

Description

Returns a function computing ell(lambda) = d * log(lambda) - lambda * T.

Usage

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

Arguments

model	An exponential_lifetime model
...	Additional arguments (ignored)

Value

A function(df, par, ...) computing the log-likelihood

logLik.fisher_mle *Extract log-likelihood from fisher_mle object*

Description

Extract log-likelihood from fisher_mle object

Usage

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

Arguments

object	A fisher_mle object
...	Additional arguments (ignored)

Value

A logLik object

loglik.likelihood_model *Default loglik method*

Description

Provides a clear error when a model class does not implement `loglik`.

Usage

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

Arguments

`model`	A likelihood model
`...`	Additional arguments

Value

Never returns; always errors.

lrt *Likelihood ratio test*

Description

Computes the likelihood ratio test statistic and p-value for nested models.

Usage

```
lrt(null, alt, data, null_par, alt_par, dof = NULL, ...)
```

Arguments

`null`	the likelihood model for the simpler (null) hypothesis nested within the alternative model
`alt`	the likelihood model for the more complicated (alternative) hypothesis
`data`	a data frame
`null_par`	parameter values under the null model
`alt_par`	parameter values under the alternative model
`dof`	degrees of freedom (computed automatically if NULL)
`...`	additional arguments passed to loglik

Value

An `lrt_result` object with components `stat`, `p.value`, and `dof`

mse.fisher_mle *Mean squared error for fisher_mle*

Description

Computes MSE = Var + Bias^2 (scalar) or Vcov + bias %*% t(bias) (matrix). Under regularity conditions, asymptotic bias is zero, so MSE equals the variance-covariance matrix. When `model` is provided, uses Monte Carlo bias estimation via `bias.fisher_mle()`.

Usage

```
## S3 method for class 'fisher_mle'
mse(x, theta = NULL, ..., model = NULL, n_sim = 1000)
```

Arguments

`x`	A fisher_mle object
`theta`	True parameter value (for simulation studies)
`...`	Additional arguments (ignored)
`model`	A likelihood model (optional, enables MC bias estimation)
`n_sim`	Number of MC replicates for bias estimation (default 1000)

Value

MSE matrix or scalar

nobs.fisher_mle *Extract number of observations from fisher_mle object*

Description

Extract number of observations from fisher_mle object

Usage

```
## S3 method for class 'fisher_mle'
nobs(object, ...)
```

Arguments

`object`	A fisher_mle object
`...`	Additional arguments (ignored)

Value

Number of observations

nparams.fisher_mle *Number of parameters in fisher_mle*

Description

Number of parameters in fisher_mle

Usage

```
## S3 method for class 'fisher_mle'
nparams(x)
```

Arguments

x	A fisher_mle object

Value

Integer count of parameters

obs.fisher_mle *Extract observed data from fisher_mle*

Description

`fisher_mle` objects do not store observed data by design, so this always returns NULL.

Usage

```
## S3 method for class 'fisher_mle'
obs(x)
```

Arguments

x	A fisher_mle object

Value

Always `NULL`. `fisher_mle` objects do not store the observed data.

observed_fim.fisher_mle *Observed Fisher information matrix from fisher_mle*

Description

Returns the negative Hessian of the log-likelihood evaluated at the MLE, which estimates the Fisher information matrix.

Usage

```
## S3 method for class 'fisher_mle'
observed_fim(x, ...)
```

Arguments

x	A fisher_mle object
...	Additional arguments (ignored)

Value

A matrix, or NULL if the Hessian was not computed

params.fisher_mle *Extract parameter estimates from fisher_mle*

Description

Extract parameter estimates from fisher_mle

Usage

```
## S3 method for class 'fisher_mle'
params(x)
```

Arguments

x	A fisher_mle object

Value

Named numeric vector of parameter estimates

print.fisher_boot *Print fisher_boot object*

Description

Print fisher_boot object

Usage

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

Arguments

x	A fisher_boot object
...	Additional arguments (ignored)

Value

The fisher_boot object, invisibly

print.fisher_mle *Print fisher_mle object*

Description

Print fisher_mle object

Usage

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

Arguments

x	A fisher_mle object
...	Additional arguments (ignored)

Value

The fisher_mle object, invisibly

print.likelihood_interval *Print likelihood interval*

Description

Print likelihood interval

Usage

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

Arguments

x	A likelihood_interval object
...	Additional arguments (ignored)

Value

The likelihood_interval object, invisibly

print.likelihood_model *Print method for likelihood models*

Description

Print method for likelihood models

Usage

```
## S3 method for class 'likelihood_model'
print(x, show.loglik = FALSE, ...)
```

Arguments

x	A likelihood model
show.loglik	Logical; if TRUE, print the log-likelihood function
...	Additional arguments (ignored)

Value

The likelihood model object, invisibly

print.lrt_result *Print method for likelihood ratio test results*

Description

Print method for likelihood ratio test results

Usage

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

Arguments

x	An lrt_result object
...	Additional arguments (ignored)

Value

The lrt_result object, invisibly

print.profile_loglik *Print profile log-likelihood*

Description

Print profile log-likelihood

Usage

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

Arguments

x	A profile_loglik object
...	Additional arguments (ignored)

Value

The profile_loglik object, invisibly

print.summary_fisher_mle *Print summary of fisher_mle*

Description

Print summary of fisher_mle

Usage

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

Arguments

x	A summary_fisher_mle object
...	Additional arguments (ignored)

Value

The summary_fisher_mle object, invisibly

profile_loglik *Profile Log-Likelihood*

Description

Computes the profile log-likelihood for a subset of parameters. For each value of the parameters of interest, the remaining (nuisance) parameters are optimized out.

Usage

```
profile_loglik(x, ...)

## S3 method for class 'fisher_mle'
profile_loglik(
  x,
  data,
  model,
  param,
  grid = NULL,
  n_grid = 50,
  range_mult = 4,
  ...
)
```

Arguments

x	A fisher_mle object
...	Additional arguments passed to loglik
data	Data frame used for likelihood computation
model	The likelihood model used for fitting
param	Index or name of parameter(s) to profile
grid	Optional grid of values to evaluate (vector or matrix)
n_grid	Number of grid points if grid not specified (default 50)
range_mult	Multiplier for grid range based on SE (default 4)

Details

The profile likelihood is useful for:

- Visualizing the likelihood surface
- Computing likelihood intervals
- Eliminating nuisance parameters

Value

A data frame with parameter values and profile log-likelihood

---

rdata *Random data generation method*

Description

Returns a function that generates random data from the model's data-generating process (DGP) at a given parameter value.

Usage

```
rdata(model, ...)
```

Arguments

model	A likelihood model
...	Additional arguments

Details

This is used by the default `fim` method for Monte Carlo estimation of the Fisher information matrix.

Value

Function that takes (theta, n, ...) and returns a data frame

rdata.exponential_lifetime

Random data generation for exponential_lifetime

Description

Generates exponential data, optionally with right-censoring at a fixed censoring time.

Usage

```
## S3 method for class 'exponential_lifetime'
rdata(model, ...)
```

Arguments

`model`	An exponential_lifetime model
`...`	Additional arguments (ignored)

Value

A function(theta, n, censor_time, ...) that returns a data frame

rdata.likelihood_model

Default rdata method

Description

Provides a clear error when a model class does not implement `rdata`. The `rdata` method is required by the default `fim.likelihood_model()` for Monte Carlo FIM estimation.

Usage

```
## S3 method for class 'likelihood_model'
rdata(model, ...)
```

Arguments

`model`	A likelihood model
`...`	Additional arguments

Value

Never returns; always errors.

`relative_likelihood` *Relative Likelihood*

Description

Computes the relative likelihood (likelihood ratio) for theta: R(theta) = L(theta) / L(theta_hat) = exp(S(theta))

Usage

```
relative_likelihood(x, ...)

## S3 method for class 'fisher_mle'
relative_likelihood(x, theta, data, model, ...)
```

Arguments

`x`	A fisher_mle object
`...`	Additional arguments passed to loglik
`theta`	Parameter value(s) to evaluate
`data`	Data frame used for likelihood computation
`model`	The likelihood model used for fitting

Details

The relative likelihood is always between 0 and 1, with maximum 1 at the MLE. Common cutoff values:

- R >= 0.15 (k=8): roughly equivalent to 95% confidence
- R >= 0.10 (k=10): more conservative
- R >= 0.05 (k=20): very conservative

Value

Relative likelihood value(s): L(theta)/L(theta_hat)

sampler.fisher_boot *Bootstrap sampler for fisher_boot*

Description

Returns a function that resamples from the bootstrap distribution.

Usage

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

Arguments

x	A fisher_boot object
...	Additional arguments (ignored)

Value

Function that takes n and returns n x p matrix of resampled estimates

sampler.fisher_mle *Asymptotic sampler for fisher_mle*

Description

Returns a function that samples from the asymptotic normal distribution of the MLE: N(theta_hat, vcov).

Usage

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

Arguments

x	A fisher_mle object
...	Additional arguments (ignored)

Value

Function that takes n and returns n x p matrix of samples

sampler.likelihood_model

Estimate the sampling distribution of the MLE for a likelihood model.

Description

We use the bootstrap method. In other words, we treat the data as an empirical distribution and sample from it to get a new dataset, then we fit the model to that dataset and return the MLE. We do this R times and return the R MLEs.

Usage

```
## S3 method for class 'likelihood_model'
sampler(x, df, par, ..., nthreads = 1L)
```

Arguments

x	The likelihood model
df	Data frame to bootstrap from
par	Initial parameter values
...	Additional arguments to pass into the likelihood model
nthreads	The number of threads to use for parallelization

Details

This is the default method, but if you want to use a different method, you should define your own method for your likelihood model.

Value

A function that returns a bootstrapped sampling distribution of an MLE (fisher_boot object).

score

Score method

Description

This function returns the score function of a model

Usage

```
score(model, ...)
```

Arguments

model	The likelihood model
...	Additional arguments

Value

A function to compute the score given a data frame and parameters

---

score.exponential_lifetime

Score for exponential_lifetime

Description

Returns a function computing d/lambda - T.

Usage

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

Arguments

model	An exponential_lifetime model
...	Additional arguments (ignored)

Value

A function(df, par, ...) computing the score vector

---

score.likelihood_model

Default score method

Description

In case a score method is not provided, this function will be used. It computes the score by numerical differentiation of the log-likelihood.

Usage

```
## S3 method for class 'likelihood_model'
score(model, control = list(), ...)
```

Arguments

model	The likelihood model
control	Custom arguments to pass to numDeriv::grad.
...	Additional arguments (to pass into loglik)

Value

A function to compute the score given a data frame and parameters

---

score_val.fisher_mle *Extract score vector from fisher_mle*

Description

Extract score vector from fisher_mle

Usage

```
## S3 method for class 'fisher_mle'
score_val(x, ...)
```

Arguments

x	A fisher_mle object
...	Additional arguments (ignored)

Value

The score vector at the MLE (should be near zero)

---

se.fisher_mle *Extract standard errors from fisher_mle*

Description

Computes standard errors as the square root of the diagonal of the variance-covariance matrix.

Usage

```
## S3 method for class 'fisher_mle'
se(x, ...)
```

Arguments

x	A fisher_mle object
...	Additional arguments (ignored)

Value

Numeric vector of standard errors, or NA values if vcov is NULL

---

summary.fisher_mle *Summarize fisher_mle object*

Description

Summarize fisher_mle object

Usage

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

Arguments

object	A fisher_mle object
...	Additional arguments (ignored)

Value

A summary_fisher_mle object

---

support *Support Function (Log Relative Likelihood)*

Description

Computes the support for parameter value theta relative to the MLE: S(theta) = logL(theta) - logL(theta_hat)

Usage

```
support(x, ...)

## S3 method for class 'fisher_mle'
support(x, theta, data, model, ...)
```

Arguments

x	A fisher_mle object
...	Additional arguments passed to loglik
theta	Parameter value(s) to evaluate
data	Data frame used for likelihood computation
model	The likelihood model used for fitting

Details

The support function is always <= 0, with maximum at the MLE. Values of theta with support > -2 are considered well-supported. Values with support > -log(8) ~ -2.08 correspond roughly to a 95% likelihood interval.

Value

Support value(s): logL(theta) - logL(theta_hat)

`vcov.fisher_mle` *Extract variance-covariance matrix from fisher_mle object*

Description

Extract variance-covariance matrix from fisher_mle object

Usage

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

Arguments

`object`	A fisher_mle object
`...`	Additional arguments (ignored)

Value

Variance-covariance matrix

Index