Package ‘algebraic.mle’

January 9, 2026

Type Package

Title Algebraic Maximum Likelihood Estimators

Version 0.9.0

Maintainer Alexander Towell <lex@metafunctor.com>

Description Defines an algebra over maximum likelihood estimators (MLEs) by providing operators that are closed over MLEs, along with various statistical functions for inference. For background on maximum likelihood estimation, see Casella and Berger (2002, ISBN:978-0534243128). For the delta method and variance estimation, see Lehmann and Casella (1998, ISBN:978-0387985022).

License GPL (>= 3)

Encoding UTF-8

ByteCompile true

Imports algebraic.dist, stats, boot, mvtnorm, MASS, numDeriv

RoxygenNote 7.3.3

URL https://github.com/queelius/algebraic.mle, https://queelius.github.io/algebraic.mle/

BugReports https://github.com/queelius/algebraic.mle/issues

Suggests rmarkdown, dplyr, knitr, ggplot2, tibble, CDFt

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2026-01-09 18:50:02 UTC

Contents

`aic` *Generic method for obtaining the AIC of a fitted distribution object fit.*

Description

Generic method for obtaining the AIC of a fitted distribution object fit.

Usage

```
aic(x)
```

Arguments

`x`	the object to obtain the AIC of

Value

The Akaike Information Criterion value (numeric).

`aic.mle` *Method for obtaining the AIC of an ‘mle‘ object.*

Description

Method for obtaining the AIC of an ‘mle‘ object.

Usage

```
## S3 method for class 'mle'
aic(x)
```

Arguments

`x`	the ‘mle‘ object to obtain the AIC of

Value

Numeric AIC value.

`algebraic.mle` *'algebraic.mle': A package for algebraically operating on and generating maximum likelihood estimators from existing maximum likelihood estimators.*

Description

The object representing a fitted model is a type of 'mle' object, the maximum likelihood estimator of the model with respect to observed data.

Details

It has a relatively rich API for working with these objects to help you understand your MLE estimator.#'

Author(s)

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

See Also

Useful links:

- https://github.com/queelius/algebraic.mle
- https://queelius.github.io/algebraic.mle/
- Report bugs at https://github.com/queelius/algebraic.mle/issues

---

`bias` *Generic method for computing the bias of an estimator object.*

Description

Generic method for computing the bias of an estimator object.

Usage

```
bias(x, theta, ...)
```

Arguments

`x`	the object to compute the bias of.
`theta`	true parameter value. usually, this is unknown (NULL), in which case we estimate the bias
`...`	pass additional arguments

Value

The bias of the estimator. The return type depends on the specific method.

`bias.mle` *Computes the bias of an ‘mle‘ object assuming the large sample approximation is valid and the MLE regularity conditions are satisfied. In this case, the bias is zero (or zero vector).*

Description

This is not a good estimate of the bias in general, but it’s arguably better than returning ‘NULL‘.

Usage

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

Arguments

`x`	the ‘mle‘ object to compute the bias of.
`theta`	true parameter value. normally, unknown (NULL), in which case we estimate the bias (say, using bootstrap)
`...`	additional arguments to pass

Value

Numeric vector of zeros (asymptotic bias is zero under regularity conditions).

`bias.mle_boot` *Computes the estimate of the bias of a ‘mle_boot‘ object.*

Description

Computes the estimate of the bias of a ‘mle_boot‘ object.

Usage

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

Arguments

`x`	the ‘mle_boot‘ object to compute the bias of.
`theta`	true parameter value (not used for ‘mle_boot‘).
`...`	pass additional arguments (not used)

Value

Numeric vector of estimated bias (mean of bootstrap replicates minus original estimate).

confint.mle — *Function to compute the confidence intervals of ‘mle‘ objects.*

Description

Function to compute the confidence intervals of ‘mle‘ objects.

Usage

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

Arguments

`object`	the ‘mle‘ object to compute the confidence intervals for
`parm`	the parameters to compute the confidence intervals for (not used)
`level`	confidence level, defaults to 0.95 (alpha=.05)
`use_t_dist`	logical, whether to use the t-distribution to compute the confidence intervals.
`...`	additional arguments to pass

Value

Matrix of confidence intervals with columns for lower and upper bounds.

confint.mle_boot — *Method for obtained the confidence interval of an ‘mle_boot‘ object. Note: This impelements the ‘vcov‘ method defined in ‘stats‘.*

Description

Method for obtained the confidence interval of an ‘mle_boot‘ object. Note: This impelements the ‘vcov‘ method defined in ‘stats‘.

Usage

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

Arguments

object	the ‘mle_boot‘ object to obtain the confidence interval of
parm	the parameter to obtain the confidence interval of (not used)
level	the confidence level
type	the type of confidence interval to compute
...	additional arguments to pass into ‘boot.ci‘

Value

Matrix of bootstrap confidence intervals with columns for lower and upper bounds.

---

confint_from_sigma — *Function to compute the confidence intervals from a variance-covariance matrix*

Description

Function to compute the confidence intervals from a variance-covariance matrix

Usage

```
confint_from_sigma(sigma, theta, level = 0.95)
```

Arguments

sigma	either the variance-covariance matrix or the vector of variances of the parameter estimator
theta	the point estimate
level	confidence level, defaults to 0.95 (alpha=.05)

Value

Matrix of confidence intervals with rows for each parameter and columns for lower and upper bounds.

Examples

```
# Compute CI for a bivariate parameter
theta <- c(mu = 5.2, sigma2 = 4.1)
vcov_matrix <- diag(c(0.1, 0.5))  # Variance of estimators

confint_from_sigma(vcov_matrix, theta)
confint_from_sigma(vcov_matrix, theta, level = 0.99)
```

`expectation.mle` *Expectation operator applied to ‘x‘ of type ‘mle‘ with respect to a function ‘g‘. That is, ‘E(g(x))‘.*

Description

Optionally, we use the CLT to construct a CI(‘alpha‘) for the estimate of the expectation. That is, we estimate ‘E(g(x))‘ with the sample mean and Var(g(x)) with the sigma^2/n, where sigma^2 is the sample variance of g(x) and n is the number of samples. From these, we construct the CI.

Usage

```
## S3 method for class 'mle'
expectation(x, g = function(t) t, ..., control = list())
```

Arguments

`x`	‘mle‘ object
`g`	characteristic function of interest, defaults to identity
`...`	additional arguments to pass to ‘g‘
`control`	a list of control parameters: compute_stats - Whether to compute CIs for the expectations, defaults to FALSE n - The number of samples to use for the MC estimate, defaults to 10000 alpha - The significance level for the confidence interval, defaults to 0.05

Value

If ‘compute_stats‘ is FALSE, then the estimate of the expectation, otherwise a list with the following components: value - The estimate of the expectation ci - The confidence intervals for each component of the expectation n - The number of samples

`is_mle` *Determine if an object ‘x‘ is an ‘mle‘ object.*

Description

Determine if an object ‘x‘ is an ‘mle‘ object.

Usage

```
is_mle(x)
```

Arguments

`x`	the object to test

Value

Logical TRUE if `x` is an `mle` object, FALSE otherwise.

Examples

```
fit <- mle(theta.hat = c(mu = 5), sigma = matrix(0.1))
is_mle(fit)       # TRUE
is_mle(list(a=1)) # FALSE
```

`is_mle_boot` *Determine if an object is an 'mle_boot' object.*

Description

Determine if an object is an 'mle_boot' object.

Usage

```
is_mle_boot(x)
```

Arguments

`x`	the object to test

Value

Logical TRUE if `x` is an `mle_boot` object, FALSE otherwise.

Examples

```
# Create a simple mle object (not bootstrap)
fit_mle <- mle(theta.hat = 5, sigma = matrix(0.1))
is_mle_boot(fit_mle)  # FALSE

# Bootstrap example would return TRUE
```

`loglik_val` *Generic method for obtaining the log-likelihood value of a fitted MLE object.*

Description

Generic method for obtaining the log-likelihood value of a fitted MLE object.

Usage

```
loglik_val(x, ...)
```

Arguments

`x`	the object to obtain the log-likelihood of
`...`	additional arguments to pass

Value

The log-likelihood value (numeric).

`loglik_val.mle` *Method for obtaining the log-likelihood of an ‘mle‘ object.*

Description

Method for obtaining the log-likelihood of an ‘mle‘ object.

Usage

```
## S3 method for class 'mle'
loglik_val(x, ...)
```

Arguments

`x`	the log-likelihood ‘l‘ evaluated at ‘x‘, ‘l(x)‘.
`...`	additional arguments to pass

Value

the log-likelihood of the fitted mle object ‘x‘

`marginal.mle` *Method for obtaining the marginal distribution of an MLE that is based on asymptotic assumptions:*

Description

‘x ~ MVN(params(x), inv(H)(x))‘

Usage

```
## S3 method for class 'mle'
marginal(x, indices)
```

Arguments

`x`	The distribution object.
`indices`	The indices of the marginal distribution to obtain.

Details

where H is the (observed or expecation) Fisher information matrix.

Value

An `mle` object representing the marginal distribution for the selected parameter indices.

`mle` *Constructor for making ‘mle‘ objects, which provides a common interface for maximum likelihood estimators.*

Description

This MLE makes the asymptotic assumption by default. Other MLEs, like ‘mle_boot‘, may not make this assumption.

Usage

```
mle(
  theta.hat,
  loglike = NULL,
  score = NULL,
  sigma = NULL,
  info = NULL,
  obs = NULL,
  nobs = NULL,
  superclasses = NULL
)
```

Arguments

`theta.hat`	the MLE
`loglike`	the log-likelihood of ‘theta.hat‘ given the data
`score`	the score function evaluated at ‘theta.hat‘
`sigma`	the variance-covariance matrix of ‘theta.hat‘ given that data
`info`	the information matrix of ‘theta.hat‘ given the data
`obs`	observation (sample) data
`nobs`	number of observations in ‘obs‘
`superclasses`	class (or classes) with ‘mle‘ as base

Value

An object of class `mle`.

Examples

```
# MLE for normal distribution (mean and variance)
set.seed(123)
x <- rnorm(100, mean = 5, sd = 2)
n <- length(x)
mu_hat <- mean(x)
var_hat <- mean((x - mu_hat)^2)  # MLE of variance

# Asymptotic variance-covariance of MLE
# For normal: Var(mu_hat) = sigma^2/n, Var(var_hat) = 2*sigma^4/n
sigma_matrix <- diag(c(var_hat/n, 2*var_hat^2/n))

fit <- mle(
  theta.hat = c(mu = mu_hat, var = var_hat),
  sigma = sigma_matrix,
  loglike = sum(dnorm(x, mu_hat, sqrt(var_hat), log = TRUE)),
  nobs = n
)

params(fit)
vcov(fit)
confint(fit)
```

`mle_boot` *Bootstrapped MLE*

Description

Sometimes, the large sample asymptotic theory of MLEs is not applicable. In such cases, we can use the bootstrap to estimate the sampling distribution of the MLE.

Usage

```
mle_boot(x)
```

Arguments

x	the ‘boot‘ return value

Details

This takes an approach similiar to the ‘mle_numerical‘ object, which is a wrapper for a ‘stats::optim‘ return value, or something that is compatible with the ‘optim‘ return value. Here, we take a ‘boot‘ object, which is the sampling distribution of an MLE, and wrap it in an ‘mle_boot‘ object and then provide a number of methods for the ‘mle_boot‘ object that satisfies the concept of an ‘mle‘ object.

Look up the ‘boot‘ package for more information on the bootstrap.

Value

An `mle_boot` object (wrapper for `boot` object).

Examples

```
# Bootstrap MLE for mean of exponential distribution
set.seed(123)
x <- rexp(50, rate = 2)

# Statistic function: MLE of rate parameter
rate_mle <- function(data, indices) {
  d <- data[indices]
  1 / mean(d)  # MLE of rate is 1/mean
}

# Run bootstrap
boot_result <- boot::boot(data = x, statistic = rate_mle, R = 200)

# Wrap in mle_boot
fit <- mle_boot(boot_result)
params(fit)
bias(fit)
confint(fit)
```

`mle_numerical` *This function takes the output of ‘optim‘, ‘newton_raphson‘, or ‘sim_anneal‘ and turns it into an ‘mle_numerical‘ (subclass of ‘mle‘) object.*

Description

This function takes the output of ‘optim‘, ‘newton_raphson‘, or ‘sim_anneal‘ and turns it into an ‘mle_numerical‘ (subclass of ‘mle‘) object.

Usage

```
mle_numerical(sol, options = list(), superclasses = NULL)
```

Arguments

`sol`	the output of ‘optim‘ or ‘newton_raphson‘
`options`	list, options for things like sigma and FIM
`superclasses`	list, superclasses to add to the ‘mle_numerical‘ object

Value

An object of class `mle_numerical` (subclass of `mle`).

Examples

```
# Fit exponential distribution using optim
set.seed(123)
x <- rexp(100, rate = 2)

# Log-likelihood for exponential distribution
loglik <- function(rate) {
  if (rate <= 0) return(-Inf)
  sum(dexp(x, rate = rate, log = TRUE))
}

# Optimize (maximize by setting fnscale = -1)
result <- optim(
  par = 1,
  fn = loglik,
  method = "Brent",
  lower = 0.01, upper = 10,
  hessian = TRUE,
  control = list(fnscale = -1)
)

# Wrap in mle_numerical
fit <- mle_numerical(result, options = list(nobs = length(x)))
params(fit)
se(fit)
```

`mle_weighted` *Accepts a list of ‘mle‘ objects for some parameter, say ‘theta‘, and combines them into a single estimator ‘mle_weighted‘.*

Description

It combines the ‘mle‘ objects by adding them together, weighted by the inverse of their respective variance-covariance matrix (information matrix). Intuitively, the higher the variance, the less weight an ‘mle‘ is given in the summation, or alternatively, the more information it has about the parameter, the more weight it is given in the summation.

Usage

```
mle_weighted(mles)
```

Arguments

`mles`	A list of 'mle' objects, all for the same parameter.

Details

Each 'mle' object should have an 'observed_fim' method, which returns the Fisher information matrix (FIM) for the parameter. The FIM is assumed to be the negative of the expected value of the Hessian of the log-likelihood function. The 'mle' objects should also have a 'params' method, which returns the parameter vector.

We assume that the observations used to estimate each of the MLE objects in 'mles' are independent.

Value

An object of type `mle_weighted` (which inherits from `mle`) that is the weighted sum of the `mle` objects.

Examples

```
# Combine three independent estimates of mean
set.seed(123)

# Three independent samples
x1 <- rnorm(50, mean = 10, sd = 2)
x2 <- rnorm(30, mean = 10, sd = 2)
x3 <- rnorm(70, mean = 10, sd = 2)

# Create MLE objects for each sample
make_mean_mle <- function(x) {
  n <- length(x)
  s2 <- var(x)
  mle(theta.hat = mean(x),
      sigma = matrix(s2/n),
      info = matrix(n/s2),
      nobs = n)
}

fit1 <- make_mean_mle(x1)
fit2 <- make_mean_mle(x2)
fit3 <- make_mean_mle(x3)

# Combine using inverse-variance weighting
combined <- mle_weighted(list(fit1, fit2, fit3))
params(combined)
se(combined)
```

`mse` *Generic method for computing the mean squared error (MSE) of an estimator, ‘mse(x) = E[(x-mu)^2]‘ where ‘mu‘ is the true parameter value.*

Description

Generic method for computing the mean squared error (MSE) of an estimator, ‘mse(x) = E[(x-mu)^2]‘ where ‘mu‘ is the true parameter value.

Usage

```
mse(x, theta)
```

Arguments

`x`	the object to compute the MSE of
`theta`	the true parameter value

Value

The mean squared error (matrix or scalar).

`mse.mle` *Computes the MSE of an ‘mle‘ object.*

Description

The MSE of an estimator is just the expected sum of squared differences, e.g., if the true parameter value is ‘x‘ and we have an estimator ‘x.hat‘, then the MSE is “‘ mse(x.hat) = E[(x.hat-x) vcov(x.hat) + bias(x.hat, x) “‘

Usage

```
## S3 method for class 'mle'
mse(x, theta = NULL)
```

Arguments

`x`	the ‘mle‘ object to compute the MSE of.
`theta`	true parameter value, defaults to ‘NULL‘ for unknown. If ‘NULL‘, then we let the bias method deal with it. Maybe it has a nice way of estimating the bias.

Details

Since ‘x‘ is not typically known, we normally must estimate the bias. Asymptotically, assuming the regularity conditions, the bias of an MLE is zero, so we can estimate the MSE as ‘mse(x.hat) = vcov(x.hat)‘, but for small samples, this is not generally the case. If we can estimate the bias, then we can replace the bias with an estimate of the bias.

Sometimes, we can estimate the bias analytically, but if not, we can use something like the bootstrap. For example, if we have a sample of size ‘n‘, we can bootstrap the bias by sampling ‘n‘ observations with replacement, computing the MLE, and then computing the difference between the bootstrapped MLE and the MLE. We can repeat this process ‘B‘ times, and then average the differences to get an estimate of the bias.

Value

The MSE as a scalar (univariate) or matrix (multivariate).

---

mse.mle_boot — *Computes the estimate of the MSE of a ‘boot‘ object.*

Description

Computes the estimate of the MSE of a ‘boot‘ object.

Usage

```
## S3 method for class 'mle_boot'
mse(x, theta = NULL, ...)
```

Arguments

x	the ‘boot‘ object to compute the MSE of.
theta	true parameter value (not used for ‘mle_boot‘)
...	pass additional arguments into ‘vcov‘

Value

The MSE matrix estimated from bootstrap variance and bias.

nobs.mle

Method for obtaining the number of observations in the sample used by an ‘mle‘.

Description

Method for obtaining the number of observations in the sample used by an ‘mle‘.

Usage

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

Arguments

object	the ‘mle‘ object to obtain the number of observations for
...	additional arguments to pass (not used)

Value

Integer number of observations, or NULL if not available.

nobs.mle_boot

Method for obtaining the number of observations in the sample used by an ‘mle‘.

Description

Method for obtaining the number of observations in the sample used by an ‘mle‘.

Usage

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

Arguments

object	the ‘mle‘ object to obtain the number of observations for
...	additional arguments to pass (not used)

Value

Integer number of observations in the original sample.

nparams.mle — *Method for obtaining the number of parameters of an ‘mle‘ object.*

Description

Method for obtaining the number of parameters of an ‘mle‘ object.

Usage

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

Arguments

x	the ‘mle‘ object to obtain the number of parameters of

Value

Integer number of parameters.

nparams.mle_boot — *Method for obtaining the number of parameters of an ‘boot‘ object.*

Description

Method for obtaining the number of parameters of an ‘boot‘ object.

Usage

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

Arguments

x	the ‘boot‘ object to obtain the number of parameters of

Value

Integer number of parameters.

`obs.mle` *Method for obtaining the observations used by the ‘mle‘ object ‘x‘.*

Description

Method for obtaining the observations used by the ‘mle‘ object ‘x‘.

Usage

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

Arguments

x	the ‘mle‘ object to obtain the number of observations for

Value

The observation data used to fit the MLE, or NULL if not stored.

`obs.mle_boot` *Method for obtaining the observations used by the ‘mle‘.*

Description

Method for obtaining the observations used by the ‘mle‘.

Usage

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

Arguments

x	the ‘mle‘ object to obtain the number of observations for

Value

The original data used for bootstrapping.

`observed_fim` *Generic method for computing the observed FIM of an 'mle' object.*

Description

Fisher information is a way of measuring the amount of information that an observable random variable 'X' carries about an unknown parameter 'theta' upon which the probability of 'X' depends.

Usage

```
observed_fim(x, ...)
```

Arguments

x	the object to obtain the fisher information of
...	additional arguments to pass

Details

The inverse of the Fisher information matrix is the variance-covariance of the MLE for 'theta'.

Some MLE objects do not have an observed FIM, e.g., if the MLE's sampling distribution was bootstrapped.

Value

The observed Fisher Information Matrix.

`observed_fim.mle` *Function for obtaining the observed FIM of an 'mle' object.*

Description

Function for obtaining the observed FIM of an 'mle' object.

Usage

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

Arguments

x	the 'mle' object to obtain the FIM of.
...	pass additional arguments

Value

The observed Fisher Information Matrix, or NULL if not available.

`orthogonal` *Generic method for determining the orthogonal parameters of an estimator.*

Description

Generic method for determining the orthogonal parameters of an estimator.

Usage

```
orthogonal(x, tol, ...)
```

Arguments

`x`	the estimator
`tol`	the tolerance for determining if a number is close enough to zero
`...`	additional arguments to pass

Value

Logical vector or matrix indicating which parameters are orthogonal.

`orthogonal.mle` *Method for determining the orthogonal components of an ‘mle‘ object ‘x‘.*

Description

Method for determining the orthogonal components of an ‘mle‘ object ‘x‘.

Usage

```
## S3 method for class 'mle'
orthogonal(x, tol = sqrt(.Machine$double.eps), ...)
```

Arguments

`x`	the ‘mle‘ object
`tol`	the tolerance for determining if a number is close enough to zero
`...`	pass additional arguments

Value

Logical matrix indicating which off-diagonal FIM elements are approximately zero (orthogonal parameters), or NULL if FIM unavailable.

`params.mle` *Method for obtaining the parameters of an ‘mle‘ object.*

Description

Method for obtaining the parameters of an ‘mle‘ object.

Usage

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

Arguments

`x`	the ‘mle‘ object to obtain the parameters of

Value

Numeric vector of parameter estimates.

`params.mle_boot` *Method for obtaining the parameters of an ‘boot‘ object.*

Description

Method for obtaining the parameters of an ‘boot‘ object.

Usage

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

Arguments

`x`	the ‘boot‘ object to obtain the parameters of.

Value

Numeric vector of parameter estimates (the original MLE).

pred — *Generic method for computing the predictive confidence interval given an estimator object ‘x‘.*

Description

Generic method for computing the predictive confidence interval given an estimator object ‘x‘.

Usage

```
pred(x, samp = NULL, alpha = 0.05, ...)
```

Arguments

x	the estimator object
samp	a sampler for random variable that is parameterized by mle ‘x‘
alpha	(1-alpha)/2 confidence interval
...	additional arguments to pass

Value

Matrix of predictive confidence intervals.

pred.mle — *Estimate of predictive interval of ‘T|data‘ using Monte Carlo integration.*

Description

Let ‘T|x ~ f(t|x)“ be the pdf of vector ‘T‘ given MLE ‘x‘ and ‘x ~ MVN(params(x),vcov(x))“ be the estimate of the sampling distribution of the MLE for the parameters of ‘T‘. Then, ‘(T,x) ~ f(t,x) = f(t|x) f(x) is the joint distribution of ‘(T,x)‘. To find ‘f(t)‘ for a fixed ‘t‘, we integrate ‘f(t,x)‘ over ‘x‘ using Monte Carlo integration to find the marginal distribution of ‘T‘. That is, we:

Usage

```
## S3 method for class 'mle'
pred(x, samp, alpha = 0.05, R = 50000, ...)
```

Arguments

x	an ‘mle‘ object.
samp	The sampler for the distribution that is parameterized by the MLE ‘x‘, i.e., ‘T|x‘.
alpha	(1-alpha)-predictive interval for ‘T|x‘. Defaults to 0.05.
R	number of samples to draw from the sampling distribution of ‘x‘. Defaults to 50000.
...	additional arguments to pass into ‘samp‘.

Details

1. Sample from MVN ‘x‘ 2. Compute ‘f(t,x)‘ for each sample 3. Take the mean of the ‘f(t,x)‘ values asn an estimate of ‘f(t)‘.

The ‘samp‘ function is used to sample from the distribution of ‘T|x‘. It should be designed to take

Value

Matrix with columns for mean, lower, and upper bounds of the predictive interval.

---

print.mle *Print method for ‘mle‘ objects.*

Description

Print method for ‘mle‘ objects.

Usage

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

Arguments

`x`	the ‘mle‘ object to print
`...`	additional arguments to pass

Value

Invisibly returns `x`.

---

print.summary_mle *Function for printing a ‘summary‘ object for an ‘mle‘ object.*

Description

Function for printing a ‘summary‘ object for an ‘mle‘ object.

Usage

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

Arguments

`x`	the ‘summary_mle‘ object
`...`	pass additional arguments

Value

Invisibly returns x.

---

rmap.mle *Computes the distribution of ‘g(x)‘ where ‘x‘ is an ‘mle‘ object.*

Description

By the invariance property of the MLE, if ‘x‘ is an ‘mle‘ object, then under the right conditions, asymptotically, ‘g(x)‘ is normally distributed, g(x) ~ normal(g(point(x)),sigma) where ‘sigma‘ is the variance-covariance of ‘f(x)‘

Usage

```
## S3 method for class 'mle'
rmap(x, g, ..., n = 1000L, method = c("mc", "delta"))
```

Arguments

x	an ‘mle‘ object
g	a function
...	additional arguments to pass to the ‘g‘ function
n	number of samples to take to estimate distribution of ‘g(x)‘ if ‘method == "mc"‘.
method	method to use to estimate distribution of ‘g(x)‘, "delta" or "mc".

Details

We provide two different methods for estimating the variance-covariance of ‘f(x)‘: method = "delta" -> delta method method = "mc" -> monte carlo method

Value

An mle object of class rmap_mle representing the transformed MLE with variance estimated by the specified method.

Examples

```
# MLE for normal distribution
set.seed(123)
x <- rnorm(100, mean = 5, sd = 2)
n <- length(x)
fit <- mle(
  theta.hat = c(mu = mean(x), var = var(x)),
  sigma = diag(c(var(x)/n, 2*var(x)^2/n)),
  nobs = n
)
```

```
# Transform: compute MLE of standard deviation (sqrt of variance)
# Using delta method
g <- function(theta) sqrt(theta[2])
sd_mle <- rmap(fit, g, method = "delta")
params(sd_mle)
se(sd_mle)
```

`sampler.mle` *Method for sampling from an ‘mle‘ object.*

Description

It creates a sampler for the ‘mle‘ object. It returns a function that accepts a single parameter ‘n‘ denoting the number of samples to draw from the ‘mle‘ object.

Usage

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

Arguments

`x`	the ‘mle‘ object to create sampler for
`...`	additional arguments to pass

Value

A function that takes parameter `n` and returns `n` samples from the asymptotic distribution of the MLE.

`sampler.mle_boot` *Method for sampling from an ‘mle_boot‘ object.*

Description

It creates a sampler for the ‘mle_boot‘ object. It returns a function that accepts a single parameter ‘n‘ denoting the number of samples to draw from the ‘mle_boot‘ object.

Usage

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

Arguments

`x`	the ‘mle_boot‘ object to create sampler for
`...`	additional arguments to pass (not used)

Details

Unlike the ‘sampler‘ method for the more general ‘mle‘ objects, for ‘mle_boot‘ objects, we sample from the bootstrap replicates, which are more representative of the sampling distribution, particularly for small samples.

Value

A function that takes parameter `n` and returns `n` samples drawn from the bootstrap replicates.

`score_val` *Generic method for computing the score of an estimator object (gradient of its log-likelihood function evaluated at the MLE).*

Description

Generic method for computing the score of an estimator object (gradient of its log-likelihood function evaluated at the MLE).

Usage

```
score_val(x, ...)
```

Arguments

`x`	the object to compute the score of.
`...`	pass additional arguments

Value

The score vector evaluated at the MLE.

`score_val.mle` *Computes the score of an ‘mle‘ object (score evaluated at the MLE).*

Description

If reguarlity conditions are satisfied, it should be zero (or approximately, if rounding errors occur).

Usage

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

Arguments

`x`	the ‘mle‘ object to compute the score of.
`...`	additional arguments to pass (not used)

Value

The score vector evaluated at the MLE, or NULL if not available.

`se` *Generic method for obtaining the standard errors of an estimator.*

Description

Generic method for obtaining the standard errors of an estimator.

Usage

```
se(x, ...)
```

Arguments

`x`	the estimator
`...`	additional arguments to pass

Value

Vector of standard errors for each parameter.

`se.mle` *Function for obtaining an estimate of the standard error of the MLE object ‘x‘.*

Description

Function for obtaining an estimate of the standard error of the MLE object ‘x‘.

Usage

```
## S3 method for class 'mle'
se(x, se.matrix = FALSE, ...)
```

Arguments

`x`	the MLE object
`se.matrix`	if ‘TRUE‘, return the square root of the variance-covariance
`...`	additional arguments to pass (not used)

Value

Vector of standard errors, or matrix if `se.matrix = TRUE`, or NULL if variance-covariance is not available.

`summary.mle` *Function for obtaining a summary of ‘object‘, which is a fitted ‘mle‘ object.*

Description

Function for obtaining a summary of ‘object‘, which is a fitted ‘mle‘ object.

Usage

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

Arguments

`object`	the ‘mle‘ object
`...`	pass additional arguments

Value

An object of class `summary_mle`.

`vcov.mle` *Computes the variance-covariance matrix of ‘mle‘ object.*

Description

Computes the variance-covariance matrix of ‘mle‘ object.

Usage

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

Arguments

`object`	the ‘mle‘ object to obtain the variance-covariance of
`...`	additional arguments to pass (not used)

Value

the variance-covariance matrix

`vcov.mle_boot` *Computes the variance-covariance matrix of ‘boot‘ object. Note: This impelements the ‘vcov‘ method defined in ‘stats‘.*

Description

Computes the variance-covariance matrix of ‘boot‘ object. Note: This impelements the ‘vcov‘ method defined in ‘stats‘.

Usage

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

Arguments

`object`	the ‘boot‘ object to obtain the variance-covariance of
`...`	additional arguments to pass into ‘stats::cov‘

Value

The variance-covariance matrix estimated from bootstrap replicates.

Index