Package 'gpci'

August 31, 2026

Type Package

Title Generalized Process Capability Indices and Bootstrap Confidence Intervals

Version 0.1.0

Description A comprehensive, generalized framework for computing, estimating, and validating Generalized Process Capability Indices (GPCIs). Supports user-supplied probability density functions (PDF/PMF), cumulative distribution functions (CDF), survival functions (SF), and quantile functions with uncensored data parameter estimation via Maximum Likelihood Estimation (MLE). Provides classical and non-normal capability indices, including Cpy (Maiti, Saha and Nanda, 2010) <doi:10.1080/16843703.2010.11673233>, Spmk (Dey and Saha, 2019) <doi:10.1007/s41872-019-00081-4>, CpTk (Saha, Dey and Maiti, 2019) <doi:10.1007/s13198-019-00789-7>, Cpc (Saha, Dey and Nadarajah, 2022) <doi:10.1080/02664763.2021.1971632>, CNpmc (Alotaibi, Dey and Saha, 2022) <doi:10.1155/2022/3135264>, CNpmkc (Saha, Tripathi and Dey, 2024) <doi:10.1142/S021853932450013X>, CNpk (Saha, Dey and Maiti, 2018) <doi:10.1080/21681015.2018.1437793>, and Vannman capability indices. Computes parametric and non-parametric bootstrap confidence intervals at 90%, 95%, and 99% confidence levels using percentile, normal, basic, BCa, BCp, and studentized bootstrap methods. Evaluates Highest Posterior Density (HPD) intervals and Heidelberger-Welch convergence diagnostics.
References:
Maiti, Saha and Nanda (2010) <doi:10.1080/16843703.2010.11673233>,
Saha, Dey and Maiti (2018) <doi:10.1080/21681015.2018.1437793>,
Dey and Saha (2019) <doi:10.1007/s41872-019-00081-4>,
Saha, Dey and Maiti (2019) <doi:10.1007/s13198-019-00789-7>,
Alotaibi, Dey and Saha (2022) <doi:10.1155/2022/3135264>,
Saha, Dey and Nadarajah (2022) <doi:10.1080/02664763.2021.1971632>,
Saha, Tripathi and Dey (2024) <doi:10.1142/S021853932450013X>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, ggplot2, numDeriv, boot

Suggests testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Shikhar Tyagi [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1606-0844>),
Sumit Kumar [aut],
Arvind Pandey [aut],
Bhupendra Singh [aut],
Vrijesh Tripathi [aut]

Maintainer Shikhar Tyagi <shikhar1093tyagi@gmail.com>

Repository CRAN

Date/Publication 2026-08-31 12:30:02 UTC

Contents

boot_ci *Compute Bootstrap Confidence Intervals for Process Capability Indices*

Description

Runs parametric or non-parametric bootstrapping to compute confidence intervals for generalized process capability indices at multiple significance levels (e.g. 90 percent, 95 percent, 99 percent).

Usage

```
boot_ci(
  fit,
  B = 2000,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "normal", "basic", "BCa", "BCp", "studentized"),
  type = c("parametric", "nonparametric"),
  parallel = FALSE,
  ncpus = 1
)

gpci_boot(
  fit,
  B = 2000,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "normal", "basic", "BCa", "BCp", "studentized"),
  type = c("parametric", "nonparametric"),
  parallel = FALSE,
  ncpus = 1
)
```

Arguments

fit	A pci_fit object returned by capability.
B	Number of bootstrap replicates (default is 2000).
alpha	Vector of significance levels (default is c(0.10, 0.05, 0.01) corresponding to 90 percent, 95 percent, and 99 percent confidence levels).
method	Confidence interval method. Choices are "percentile", "normal", "basic", "BCa", "BCp" (Bias-Corrected Percentile), or "studentized".
type	Bootstrap type: "parametric" or "nonparametric".
parallel	Logical. If TRUE, bootstrap replicates are run in parallel.
ncpus	Integer. Number of CPUs to use if parallel = TRUE.

Value

An object of class gpci_ci containing a tidy data frame of confidence intervals, the raw bootstrap output, and the original fit object.

Examples

```
dist_norm <- dist_normal()
fit <- capability(rnorm(50, 10, 1), dist_norm, USL = 13, LSL = 7)
ci <- boot_ci(fit, B = 50, alpha = c(0.10, 0.05, 0.01), method = "percentile")
print(ci)
```

---

boot_cv *Bootstrap Cross-Validation for CI Quality Evaluation*

Description

Evaluates the reliability (coverage rate, width, bias, and RMSE) of the bootstrap confidence interval procedure by simulating synthetic samples from the fitted process distribution.

Usage

```
boot_cv(
  fit,
  B2 = 100,
  B = 500,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "BCa", "normal", "basic", "studentized"),
  type = c("parametric", "nonparametric"),
  parallel = FALSE,
  ncpus = 1
)

gpci_cv(
  fit,
  B2 = 100,
  B = 500,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "BCa", "normal", "basic", "studentized"),
  type = c("parametric", "nonparametric"),
  parallel = FALSE,
  ncpus = 1
)
```

Arguments

fit	A pci_fit object representing the reference fit on observed data.
B2	Number of synthetic repetitions to run (default is 100).
B	Number of bootstrap replicates to run for each synthetic sample (default is 500).
alpha	Vector of significance levels to evaluate (default is c(0.10, 0.05, 0.01)).
method	Confidence interval method to evaluate (default is "percentile").

type	Bootstrap type: "parametric" or "nonparametric".
parallel	Logical. If TRUE, run the synthetic repetitions in parallel.
ncpus	Integer. Number of CPUs to use if parallel = TRUE.

Value

An object of class c("gpcicv", "gpci_cv") containing summarized metrics and raw repetition results.

Examples

```
dist_norm <- dist_normal()
fit <- capability(rnorm(50, 10, 1), dist_norm, USL = 13, LSL = 7)
cv <- boot_cv(fit, B2 = 10, B = 20, alpha = 0.05, method = "percentile")
print(cv)
```

---

builtin_dist *Built-in Distributions*

Description

Convenience constructors for standard distributions.

Usage

```
dist_normal(mean = 0, sd = 1)

dist_lognormal(meanlog = 0, sdlog = 1)

dist_gamma(shape = 1, scale = 1)

dist_weibull(shape = 1, scale = 1)

dist_beta(shape1 = 1, shape2 = 1)

dist_logistic_exponential(shape = 1, scale = 1)
```

Arguments

mean, sd	Mean and standard deviation parameters for normal distribution.
meanlog, sdlog	Mean and standard deviation of the distribution on the log scale.
shape, scale	Shape and scale parameters.
shape1, shape2	Non-negative parameters of the Beta distribution.

Value

An object of class gpci_dist representing the specified parametric distribution.

capability *Compute Process Capability Indices*

Description

Computes classical and generalized Process Capability Indices (PCIs) for a given dataset or process distribution under uncensored data parameter estimation.

Usage

```
capability(
  data = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
  indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  fit_method = c("mle", "moments"),
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL
)

gpci_fit(
  data = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
  indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  fit_method = c("mle", "moments"),
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
```

```
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL
)
```

Arguments

data	Numeric vector of observed process data. Can be NULL if distribution parameters are fixed.
distribution	A gpci_dist distribution object.
USL	Numeric value of the Upper Specification Limit.
LSL	Numeric value of the Lower Specification Limit.
target	Numeric value of the process target (defaults to the midpoint of USL and LSL).
indices	Character vector of indices to compute. Choices include Vännman's classical family ("Cp", "Cpk", "Cpl", "Cpu", "Cpm", "Cpmk", "Cp_uv") and their quantile analogs ("Cp_q", "Cpk_q", "Cpl_q", "Cpu_q", "CNpk", "Cpm_q", "Cpmk_q", "CNp_uv"), as well as generalized indices from literature ("Cpy", "CpTk", "Spmk", "Cpc", "CNpmc", "CNpmkc").
u	Parameter u for the generalized Cp(u, v) family (default is 1).
v	Parameter v for the generalized Cp(u, v) family (default is 1).
mode	Mode of computation: "moments" (uses mean and variance) or "quantile" (uses robust quantiles).
fit	Logical. If TRUE (default) and data is supplied, parameter values of the distribution are estimated from uncensored data.
fit_method	Parameter estimation method: "mle" (default) or "moments".
C0, C1, C2	Coefficients for the tolerance cost function in CNpmc and CNpmkc (defaults: C0 = 1, C1 = 0, C2 = 1).
tolerance_t	Process tolerance t for the tolerance cost function (defaults to USL - LSL).
P0	Desirable yield for the Cpc and Cpy indices (default is 0.9973002).
LDL, UDL	Lower and Upper Desired Limits for the CpTk index (default to LSL and USL respectively).

Value

An object of class c("gpcifit", "pci_fit") containing raw data, specification limits, fitted parameters, and computed indices.

Examples

```
dist_norm <- dist_normal()
capability(
  data = rnorm(100, mean = 10, sd = 1),
  distribution = dist_norm,
  USL = 13, LSL = 7, target = 10,
  indices = c("Cpy", "Cp", "Cpk", "Cpm", "Cpmk", "CpTk", "Spmk", "CNpmc"),
  mode = "moments"
)
```

coef.gpcifit *Coef Method for gpcifit*

Description

Extract point estimates of capability indices or fitted distribution parameters.

Usage

```
## S3 method for class 'gpcifit'
coef(object, what = c("indices", "parameters"), ...)
```

Arguments

object	An object of class gpcifit.
what	Character string: "indices" (default) or "parameters".
...	Additional arguments.

Value

A named numeric vector containing either the estimated capability indices (if what = "indices") or the fitted distribution parameters (if what = "parameters").

coef.gpcimcmc *Coef Method for gpcimcmc Objects*

Description

Extract posterior means or point estimates of GPCIs or parameters.

Usage

```
## S3 method for class 'gpcimcmc'
coef(object, what = c("indices", "parameters"), ...)
```

Arguments

object	An object of class gpcimcmc.
what	Character string: "indices" (default) or "parameters".
...	Additional arguments.

Value

A named numeric vector containing either the posterior means of capability indices (if what = "indices") or posterior means of parameters (if what = "parameters").

compute_theoretical_moments *Compute Theoretical Moments of a Distribution*

Description

Computes the mean and variance of a distribution object using numerical integration.

Usage

```
compute_theoretical_moments(dist)
```

Arguments

dist	A gpci_dist object.

Value

A list with mean and var.

confint.gpcifit *Confint Method for gpcifit*

Description

Calculate confidence intervals for capability indices via bootstrap at specified significance levels (90

Usage

```
## S3 method for class 'gpcifit'
confint(
  object,
  parm = NULL,
  level = 0.95,
  B = 1000,
  method = "percentile",
  ...
)
```

Arguments

`object`	An object of class `gpcifit`.
`parm`	Optional vector of index names.
`level`	Confidence level (default is 0.95; can be 0.90, 0.95, 0.99 or a vector).
`B`	Number of bootstrap replicates (default 1000).
`method`	Bootstrap CI method (`"percentile"`, `"normal"`, `"basic"`, `"BCa"`, `"BCp"`, `"studentized"`).
`...`	Additional arguments.

Value

A numeric matrix of class `matrix` containing lower and upper bootstrap confidence bounds for the requested capability indices.

`define_distribution` *Define a Process Distribution*

Description

Constructor to define a probability distribution for process capability analysis. The distribution can be defined by specifying any one (or more) of its PDF/PMF, CDF, Survival Function (SF), or quantile function. The engine will numerically derive whichever representations are missing.

Usage

```
define_distribution(
  name,
  pdf = NULL,
  cdf = NULL,
  sf = NULL,
  quantile = NULL,
  params = list(),
  support = c(-Inf, Inf)
)

gpci_dist(
  name,
  pdf = NULL,
  cdf = NULL,
  sf = NULL,
  quantile = NULL,
  params = list(),
  support = c(-Inf, Inf)
)
```

Arguments

name	Character string naming the distribution.
pdf	Function representing the probability density function (PDF/PMF). Must be of the form function(x, ...).
cdf	Function representing the cumulative distribution function (CDF). Must be of the form function(x, ...).
sf	Function representing the survival function (SF = 1 - CDF). Must be of the form function(x, ...).
quantile	Function representing the quantile function. Must be of the form function(p, ...).
params	Named list of parameters for the distribution.
support	Vector of length 2 defining the lower and upper bounds of the support of the distribution (default is c(-Inf, Inf)).

Value

An object of class gpci_dist representing the completed distribution.

Examples

```
# Define custom Weibull distribution using only the survival function
custom_weib <- define_distribution(
  name = "custom_weibull",
  sf = function(x, shape, scale) pweibull(x, shape, scale, lower.tail = FALSE),
  params = list(shape = 2, scale = 10),
  support = c(0, Inf)
)
```

---

fit_distribution *Fit Distribution Parameters under Uncensored Data*

Description

Fits the parameters of a gpci_dist distribution to uncensored data using Maximum Likelihood Estimation (MLE) or Method of Moments, with Hessian estimation and automatic fallback optimization.

Usage

```
fit_distribution(data, dist, method = c("mle", "moments"), start = NULL)

gpci_mle(data, dist, start = NULL)
```

Arguments

data	Numeric vector of uncensored observations.
dist	A gpci_dist distribution object with initial parameters.
method	Fitting method: "mle" (Maximum Likelihood) or "moments".
start	Optional named list of starting parameter values.

Value

A new gpci_dist object containing fitted parameters and variance-covariance attribute "vcov".

---

gpci_index *Dispatcher for Index Computation*

Description

Dispatcher for Index Computation

Usage

```
gpci_index(fit, index)
```

Arguments

fit	A gpcifit / pci_fit object.
index	Name of index to compute.

Value

Numeric value of index.

---

gpci_mcmc *Bayesian MCMC Estimation of Generalized Process Capability Indices*

Description

Estimates Generalized Process Capability Indices (GPCIs) and model parameters using Metropolis-Hastings within Gibbs sampling for uncensored data. Computes posterior metrics, point estimates (MLE), bias, MSE, Bayes risk value under squared error loss, Highest Posterior Density (HPD) intervals at 90 and Heidelberger & Welch's MCMC convergence diagnostics.

Usage

```
gpci_mcmc(
  data,
  pdf = NULL,
  cdf = NULL,
  sf = NULL,
  distribution = NULL,
  USL,
  LSL,
  target = (USL + LSL)/2,
  priors = NULL,
  length_chain = 10000,
  burn_in = 2000,
  thinning = 5,
  start = NULL,
  proposal_sd = NULL,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "CpTk", "Spmk", "Cpc",
    "CNpmc", "CNpmkc"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL
)
```

Arguments

`data`	Numeric vector of uncensored observations.
`pdf`	Probability density function `function(x, ...)`.
`cdf`	Cumulative distribution function `function(x, ...)`.
`sf`	Survival function `function(x, ...)`. Defaults to `1 - cdf(x, ...)`.
`distribution`	Optional `gpci_dist` distribution object or character string naming a built-in distribution.
`USL`	Upper Specification Limit.
`LSL`	Lower Specification Limit.
`target`	Target process value (default is `(USL + LSL) / 2`).
`priors`	Named list of prior specifications for each model parameter, or custom log-prior evaluator.
`length_chain`	Total number of MCMC iterations (default 10000).
`burn_in`	Number of burn-in iterations to discard (default 2000).

thinning	Thinning factor for MCMC chain (default 5).
start	Optional vector of starting parameter values. Defaults to uncensored MLE estimates.
proposal_sd	Vector or list of proposal standard deviations for M-H sampler.
indices	Character vector of capability indices to compute (e.g., c("Cpy", "Cp", "Cpk", "Cpm", "Cpmk", "CpTk", "Spmk", "Cpc", "CNpmc", "CNpmkc")).
u, v	Parameters for the generalized Cp(u, v) family.
mode	Mode of computation: "moments" (default) or "quantile".
C0, C1, C2	Parameters for tolerance cost function in CNpmc and CNpmkc.
tolerance_t	Process tolerance parameter (default USL - LSL).
P0	Desirable yield benchmark for Cpc and Cpy (default 0.9973002).
LDL, UDL	Lower and Upper Desired Limits for CpTk index.

Value

An object of class c("gpcimcmc", "gpci_mcmc") containing:

summary_table DataFrame containing index estimates, posterior mean, bias, MSE, Risk, 90/95/99 percent HPD intervals, and HW diagnostic metrics.

param_table DataFrame containing parameter MLE, posterior mean, bias, MSE, Risk, HPD intervals, and HW diagnostics.

param_chain Matrix of thinned post-burn-in MCMC samples for parameters.

gpci_chain Matrix of thinned post-burn-in MCMC samples for GPCIs.

diagnostics List of detailed Heidelberger-Welch convergence test results.

Examples

```
set.seed(123)
dat <- rnorm(30, mean = 10, sd = 1)
fit_mcmc <- gpci_mcmc(
  data = dat,
  pdf = function(x, mean, sd) dnorm(x, mean, sd),
  cdf = function(x, mean, sd) pnorm(x, mean, sd),
  USL = 13, LSL = 7, target = 10,
  length_chain = 150, burn_in = 30, thinning = 2
)
print(fit_mcmc)
```

gpci_plot *Generic Dispatcher for gpci Plots*

Description

Generic Dispatcher for gpci Plots

Usage

```
gpci_plot(object, type = "density", ...)
```

Arguments

object	An object of class gpcifit, gpci_ci, gpcicv, or gpcimcmc.
type	Plot type.
...	Additional arguments.

Value

Invisibly returns the input object object, called for its side effect of displaying capability diagnostic plots.

heidelberger_welch *Heidelberger and Welch's MCMC Convergence Diagnostic*

Description

Conducts the stationarity test and relative half-width test under Heidelberger and Welch's MCMC convergence diagnostic.

Usage

```
heidelberger_welch(x, alpha = 0.05, eps = 0.1)
```

Arguments

x	Numeric vector representing an MCMC chain.
alpha	Significance level for the test (default is 0.05).
eps	Target maximum ratio of half-width to sample mean (default is 0.1).

Value

A list containing Cramér-von Mises statistic (stat), p-value (pvalue), stationarity test status (passed), half-width test status (hw_passed), half-width statistic (hw_stat), and convergence probability (convergence_prob).

Examples

```
samples <- rnorm(1000)
heidelberger_welch(samples)
```

`hpd_interval` *Highest Posterior Density (HPD) Interval Calculation*

Description

Computes the Highest Posterior Density (HPD) interval for a sample vector at specified confidence / significance levels (e.g., 90

Usage

```
hpd_interval(x, prob = 0.95)
```

Arguments

`x`	Numeric vector of MCMC posterior samples.
`prob`	Credibility level (1 - significance level). Default is 0.95.

Value

A named numeric vector of length 2 containing `lower` and `upper` HPD bounds.

Examples

```
samples <- rnorm(1000, mean = 5, sd = 1)
hpd_interval(samples, prob = 0.95)
```

`plot_gpci` *Plotting and Visualizations for Process Capability Fits*

Description

Diagnostic and capability plots for gpci fits, confidence intervals, and cross-validation objects.

Usage

```
## S3 method for class 'pci_fit'
autoplot(
  object,
  type = c("density", "scorecard", "sensitivity", "cdf", "qq", "run", "all"),
  ...
)

## S3 method for class 'gpcifit'
autoplot(
  object,
  type = c("density", "scorecard", "sensitivity", "cdf", "qq", "run", "all"),
  ...
)

## S3 method for class 'gpci_ci'
autoplot(object, type = c("boot", "forest", "all"), ...)

## S3 method for class 'gpci_cv'
autoplot(object, type = c("coverage", "width"), ...)

## S3 method for class 'gpcicv'
autoplot(object, type = c("coverage", "width"), ...)

## S3 method for class 'gpcimcmc'
autoplot(object, type = c("density", "trace", "all"), ...)

## S3 method for class 'gpci_mcmc'
autoplot(object, type = c("density", "trace", "all"), ...)

## S3 method for class 'pci_fit'
plot(x, type = "density", ...)

## S3 method for class 'gpcifit'
plot(x, type = "density", ...)

## S3 method for class 'gpci_ci'
plot(x, type = "boot", ...)

## S3 method for class 'gpci_cv'
plot(x, type = "coverage", ...)

## S3 method for class 'gpcicv'
plot(x, type = "coverage", ...)

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

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

Arguments

type	Character string specifying the plot type. For gpcifit/pci_fit: "density" (default), "scorecard", "sensitivity", "cdf", "qq", "run", or "all". For gpci_ci: "boot" (default), "forest", or "all". For gpcicv/gpci_cv: "coverage" (default) or "width".
...	Additional arguments.
x, object	An object of class gpcifit, pci_fit, gpci_ci, gpcicv, or gpci_cv.

Value

A ggplot object (or list of plots).

---

print.gpcimcmc *Print Method for gpcimcmc Objects*

Description

Print Method for gpcimcmc Objects

Usage

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

Arguments

x	An object of class gpcimcmc.
...	Additional print parameters.

Value

Invisibly returns the input object x of class gpcimcmc, printed for side effects.

print.gpci_ci *Print Method for gpci_ci*

Description

Print Method for gpci_ci

Usage

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

Arguments

x	An object of class gpci_ci.
...	Additional print arguments.

Value

Invisibly returns the input object x of class gpci_ci, printed for side effects.

print.gpci_cv *Print Method for gpci_cv / gpcicv*

Description

Print Method for gpci_cv / gpcicv

Usage

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

Arguments

x	An object of class gpci_cv.
...	Additional print arguments.

Value

Invisibly returns the input object x of class gpci_cv, printed for side effects.

print.pci_fit *Print Method for pci_fit / gpcifit*

Description

Print Method for pci_fit / gpcifit

Usage

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

Arguments

x	An object of class pci_fit.
...	Additional print arguments.

Value

Invisibly returns the input object x of class pci_fit, printed for side effects.

summary.gpcimcmc *Summary Method for gpcimcmc Objects*

Description

Summary Method for gpcimcmc Objects

Usage

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

Arguments

object	An object of class gpcimcmc.
...	Additional arguments.

Value

Invisibly returns the input object object of class gpcimcmc, printed for side effects.

summary.gpci_ci *Summary Method for gpci_ci*

Description

Summary Method for gpci_ci

Usage

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

Arguments

object	An object of class gpci_ci.
...	Additional arguments.

Value

Invisibly returns the input object object of class gpci_ci, printed for side effects.

summary.gpci_cv *Summary Method for gpci_cv / gpcicv*

Description

Summary Method for gpci_cv / gpcicv

Usage

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

Arguments

object	An object of class gpci_cv.
...	Additional arguments.

Value

Invisibly returns the input object object of class gpci_cv, printed for side effects.

summary.pci_fit *Summary Method for pci_fit / gpcifit*

Description

Summary Method for pci_fit / gpcifit

Usage

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

Arguments

object	An object of class pci_fit.
...	Additional arguments.

Value

Invisibly returns the input object object of class pci_fit, printed for side effects.

vcov.gpcifit *Vcov Method for gpcifit*

Description

Extract asymptotic variance-covariance matrix of fitted parameters (if MLE was performed).

Usage

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

Arguments

object	An object of class gpcifit.
...	Additional arguments.

Value

A numeric matrix of class matrix containing the variance-covariance matrix of the parameter estimates if Maximum Likelihood Estimation was performed, or NULL with a warning if unavailable.

Index