Package 'gpcihybridIIEM'

August 21, 2026

Type Package

Title Generalized Process Capability Indices via EM for Hybrid Type-II Data

Version 0.1.0

Description Implements the Expectation-Maximization (EM) algorithm of Dempster, Laird, and Rubin (1977) <doi:10.1111/j.2517-6161.1977.tb01600.x> for parameter estimation under Hybrid Type-II censored data (Childs et al. (2003) <doi:10.1007/BF02517803>; Balakrishnan and Kundu (2013) <doi:10.1002/nav.21545>) using the 'UniCensorEM' package and computes Generalized Process Capability Indices (GPCIs). Supports classical and generalized capability indices including Cpy (Maiti et al. (2010) <doi:10.1080/16843703.2010.11673233>), Cp, Cpk, Cpu, Cpl, Cpm, Cpmk, Spmk (Dey and Saha (2019) <doi:10.1007/s41872-019-00081-4>), CpTk (Saha et al. (2018) <doi:10.1080/21681015.2018.1437793>), Cpc, CNpmc (Alotaibi et al. (2022) <doi:10.1155/2022/3135264>), CNpmkc (Saha et al. (2024) <doi:10.1142/S021853932450013X>), and CNpk (Saha et al. (2022) <doi:10.1080/02664763.2021.1971632>). Computes point estimates, bias, mean squared error, risk, Heidelberger and Welch convergence diagnostics, convergence probability, Bayesian MCMC sampling chains, goodness-of-fit testing via 'gofPHCS', and bootstrap confidence intervals at 90 percent, 95 percent, and 99 percent levels. Accommodates user-defined probability density or mass functions, cumulative distribution functions, and survival functions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, graphics, UniCensorEM, gofPHCS

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Shikhar Tyagi [aut, cre] (ORCID: <https://orcid.org/0000-0003-1606-0844>)

Maintainer Shikhar Tyagi <shikhar1093tyagi@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 13:00:31 UTC

Contents

`boot_ci_hybrid2_em` *Bootstrap Confidence Intervals for Hybrid Type-II GPCI (EM)*

Description

Computes parametric or non-parametric bootstrap confidence intervals for Generalized Process Capability Indices and model parameters under Hybrid Type-II censoring using EM algorithm estimation. Provides bootstrap CIs at 90%, 95%, and 99% levels, plus estimates of bias, MSE, Risk values, Heidelberger-Welch convergence diagnostics, and convergence probabilities.

Usage

```
boot_ci_hybrid2_em(
  fit,
  B = 1000,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "normal", "basic", "BCp", "BCa", "studentized"),
  type = c("parametric", "nonparametric"),
  max_iter = 1000,
  tol = 1e-08
)

gpc_boot_hybrid2_em(
  fit,
  B = 1000,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "normal", "basic", "BCp", "BCa", "studentized"),
  type = c("parametric", "nonparametric"),
  max_iter = 1000,
  tol = 1e-08
)

boot_ci_hybrid_ty2_em(
  fit,
  B = 1000,
  alpha = c(0.1, 0.05, 0.01),
  method = c("percentile", "normal", "basic", "BCp", "BCa", "studentized"),
  type = c("parametric", "nonparametric"),
  max_iter = 1000,
  tol = 1e-08
)
```

Arguments

`fit`	A `gpc_hybrid2_em_fit` object returned by `capability_hybrid2_em`.
`B`	Number of bootstrap replicates (default 1000).
`alpha`	Numeric vector of significance levels (default `c(0.10, 0.05, 0.01)` for 90%, 95%, 99% CIs).
`method`	CI method: `"percentile"`, `"normal"`, `"basic"`, `"BCp"`, `"BCa"`, or `"studentized"`.
`type`	Resampling type: `"parametric"` (default) or `"nonparametric"`.
`max_iter`	Maximum EM iterations per bootstrap (default 1000).
`tol`	Convergence tolerance per bootstrap (default 1e-8).

Value

An object of class `"gpc_hybrid2_em_ci"` containing:

`ci_table`	Data frame of bootstrap confidence intervals with columns: `type`, `index`, `estimate`, `method`, `bootstrap_type`, `alpha`, `conf_level`, `lower`, `upper`, and `width`.

`param_perf`	Data frame of parameter performance metrics (`item`, `estimate`, `se`, `bias`, `mse`, `risk`).
`index_perf`	Data frame of capability index performance metrics (`item`, `estimate`, `se`, `bias`, `mse`, `risk`).
`convergence`	Data frame with Heidelberger-Welch stationarity test statistics, half-width test results, and convergence probability for each index.
`param_reps`	Numeric matrix of bootstrap parameter replicates across all B iterations.
`index_reps`	Numeric matrix of bootstrap capability index replicates across all B iterations.
`fit`	Original `gpc_hybrid2_em_fit` object.
`B`	Integer number of bootstrap replicates requested.

References

Efron, B., & Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*. CRC press.

Heidelberger, P., & Welch, P. D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6), 1109-1144. doi:10.1287/opre.31.6.1109

Maiti, S. S., Saha, M., & Nanda, A. K. (2010). On generalizing process capability indices. *Quality Technology & Quantitative Management*, 7(3), 279-300. doi:10.1080/16843703.2010.11673233

Saha, M., Maiti, S. S., & Dey, S. (2018). Bootstrap confidence intervals of CpTk for Birnbaum-Saunders distribution. *Journal of Industrial and Production Engineering*, 35(6), 397-408. doi:10.1080/21681015.2018.1437793

Examples

```
my_dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fit <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = my_dist, USL = 3, LSL = 0)
ci <- boot_ci_hybrid2_em(fit, B = 30, alpha = c(0.10, 0.05, 0.01),
  method = "percentile")
print(ci)
```

---

capability_hybrid2_em *Compute Generalized Process Capability Indices via EM Algorithm for Hybrid Type-II Data*

Description

Computes classical and generalized Process Capability Indices (PCIs) for Hybrid Type-II censored lifetime data using model parameter estimates obtained from the Expectation-Maximization (EM) algorithm.

Usage

```
capability_hybrid2_em(
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL,
  max_iter = 1000,
  tol = 1e-08
)

gpc_hybrid2_em_fit(
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
```

```
  UDL = USL,
  max_iter = 1000,
  tol = 1e-08
)

capability_hybrid_ty2_em(
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  C0 = 1,
  C1 = 0,
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL,
  max_iter = 1000,
  tol = 1e-08
)

gpc_hybrid2_em(
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  u = 1,
  v = 1,
  mode = c("moments", "quantile"),
  fit = TRUE,
  C0 = 1,
  C1 = 0,
```

```
  C2 = 1,
  tolerance_t = USL - LSL,
  P0 = 0.9973002,
  LDL = LSL,
  UDL = USL,
  max_iter = 1000,
  tol = 1e-08
)
```

Arguments

x	Numeric vector of observed failure times under Hybrid Type-II censoring. Can be NULL if parameters are already fitted.
r	Positive integer scalar specifying target number of failures.
tc	Positive numeric scalar specifying fixed censoring time.
n	Positive integer scalar specifying total initial sample size placed on test.
distribution	A gpc_dist distribution object.
USL	Upper Specification Limit.
LSL	Lower Specification Limit.
target	Process target (defaults to (USL + LSL) / 2).
indices	Character vector of capability indices to compute. Choices: "Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Cp_uv", "Spmk", "CpTk", "Cpc", "CNpmc", "CNpmkc", "CNpk", "Cp_q", "Cpk_q".
u	Weight parameter for Cp(u,v) family (default 1).
v	Weight parameter for Cp(u,v) family (default 1).
mode	Mode of evaluation: "moments" (mean and variance) or "quantile" (quantile-based).
fit	Logical. If TRUE (default), estimates distribution parameters using the EM algorithm before computing indices. If FALSE, computes indices directly using the distribution's current parameter values.
C0	Cost function parameter (default 1).
C1	Cost function parameter (default 0).
C2	Cost function parameter (default 1).
tolerance_t	Tolerance length (defaults to USL - LSL).
P0	Desired conformance level for Cpy and Cpc (default 0.9973002).
LDL	Lower Desired Limit for CpTk (defaults to LSL).
UDL	Upper Desired Limit for CpTk (defaults to USL).
max_iter	Maximum EM iterations (default 1000).
tol	Convergence tolerance for EM (default 1e-8).

Value

An object of class `c("gpc_hybrid2_em_fit", "gpcifit")` containing:

`x`	Numeric vector of observed failure times, or `NULL` if not supplied.
`r`	Target number of failures.
`tc`	Fixed censoring time.
`n`	Total sample size.
`distribution`	Fitted `gpc_dist` distribution object.
`USL`	Numeric Upper Specification Limit.
`LSL`	Numeric Lower Specification Limit.
`target`	Numeric process target value.
`indices`	Character vector of computed capability index names.
`estimates`	Named numeric vector of capability index estimates.
`p_hat`	Estimated nonconforming proportion ($\hat{p} = 1 - \text{Yield}$).
`mode`	Computation mode used (`"moments"` or `"quantile"`).
`options`	List of all computation options and tolerance limits.

References

Maiti, S. S., Saha, M., & Nanda, A. K. (2010). On generalizing process capability indices. *Quality Technology & Quantitative Management*, 7(3), 279-300. doi:10.1080/16843703.2010.11673233

Pearn, W. L., Kotz, S., & Johnson, N. L. (1992). Testing process capability indices. *Journal of Quality Technology*, 24(4), 216-231. doi:10.1080/00224065.1992.11979403

Dey, S., & Saha, M. (2019). Assessing the process capability index Spmk for Weibull distribution. *Journal of Statistical Theory and Practice*, 13(4), 62. doi:10.1007/s41872019000814

Saha, M., Maiti, S. S., & Dey, S. (2018). Bootstrap confidence intervals of CpTk for Birnbaum-Saunders distribution. *Journal of Industrial and Production Engineering*, 35(6), 397-408. doi:10.1080/21681015.2018.1437793

Alotaibi, N., Elbatal, I., Almetwally, E. M., Alyami, S. A., & Al-Moisheer, A. S. (2022). Estimation and confidence intervals of a new PCI CNpmc for logistic-exponential distribution. *Journal of Mathematics*, 2022, 3135264. doi:10.1155/2022/3135264

Saha, M., Smarandache, F., & Dey, S. (2024). Classical and Bayesian inference of a process capability index for non-normal distribution. *International Journal of Reliability, Quality and Safety Engineering*, 31(01), 2450013. doi:10.1142/S021853932450013X

Saha, M., Dey, S., & Maiti, S. S. (2022). On generalizing a process capability index for non-normal populations. *Journal of Applied Statistics*, 49(5), 1133-1153. doi:10.1080/02664763.2021.1971632

Vannman, K. (1995). A unified approach to capability indices. *Statistica Sinica*, 5(2), 805-820.

Examples

```
my_dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
cap <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = my_dist, USL = 3, LSL = 0)
print(cap)
```

coef.gpc_hybrid2_em_fit

Coef Method for Hybrid Type-II GPCI EM Fit

Description

Coef Method for Hybrid Type-II GPCI EM Fit

Usage

```
## S3 method for class 'gpc_hybrid2_em_fit'
coef(object, what = c("indices", "parameters"), ...)
```

Arguments

object	Object of class gpc_hybrid2_em_fit.
what	Character string specifying whether to extract "indices" (default) or "parameters".
...	Additional arguments passed to coef methods.

Value

Named numeric vector of estimated index values or distribution parameter estimates.

eval_performance_hybrid2_em

Monte Carlo Simulation Framework for Hybrid Type-II Censoring (EM)

Description

Evaluates empirical standard errors, bias, MSE, and coverage probabilities for model parameters and Generalized Process Capability Indices (GPCIs) across Monte Carlo simulation runs using the EM algorithm.

Usage

```
eval_performance_hybrid2_em(
  distribution,
  r,
  tc,
  n,
  USL,
  LSL,
  target = (USL + LSL)/2,
  true_indices = NULL,
```

```
  M = 100,
  B = 200,
  alpha = c(0.1, 0.05, 0.01),
  max_iter = 1000,
  tol = 1e-08
)

gpc_sim_hybrid2_em(
  distribution,
  r,
  tc,
  n,
  USL,
  LSL,
  target = (USL + LSL)/2,
  true_indices = NULL,
  M = 100,
  B = 200,
  alpha = c(0.1, 0.05, 0.01),
  max_iter = 1000,
  tol = 1e-08
)

eval_performance_em(
  distribution,
  r,
  tc,
  n,
  USL,
  LSL,
  target = (USL + LSL)/2,
  true_indices = NULL,
  M = 100,
  B = 200,
  alpha = c(0.1, 0.05, 0.01),
  max_iter = 1000,
  tol = 1e-08
)
```

Arguments

distribution	A gpc_dist distribution object with known true parameters.
r	Positive integer target number of failures.
tc	Positive numeric fixed censoring time.
n	Positive integer total number of initial units placed on life test.
USL	Numeric Upper Specification Limit.
LSL	Numeric Lower Specification Limit.

target	Numeric process target. Defaults to (USL + LSL) / 2.
true_indices	Optional named numeric vector of true capability index values.
M	Positive integer scalar specifying number of Monte Carlo simulation runs (default 100).
B	Positive integer scalar specifying number of bootstrap iterations per simulation run (default 200).
alpha	Numeric vector of significance levels (default c(0.10, 0.05, 0.01)).
max_iter	Maximum EM iterations (default 1000).
tol	Convergence tolerance (default 1e-8).

Value

An S3 object of class "gpc_sim_hybrid2_em" containing:

param_summary	Data frame of true values, mean estimates, standard errors (SE), bias, and MSE for model parameters.
index_summary	Data frame of true values, mean estimates, standard errors (SE), bias, and MSE for capability indices.
coverage_probabilities	Matrix of empirical coverage probabilities across all significance levels.
M	Total number of simulation runs requested.
valid_runs	Number of successfully converged simulation runs.
B	Number of bootstrap replicates per run.
true_parameters	Vector of true parameter values.
true_indices	Vector of true capability index values.

Examples

```
dist_true <- dist_exponential(rate = 1)
sim_res <- eval_performance_hybrid2_em(
  distribution = dist_true, r = 4, tc = 1.0, n = 10,
  USL = 3, LSL = 0, M = 2, B = 10
)
print(sim_res)
```

`fit_hybrid2_em` *Parameter Estimation via EM Algorithm for Hybrid Type-II Censored Data*

Description

Fits process distribution parameters to Hybrid Type-II censored lifetime data using the Expectation-Maximization (EM) algorithm via the `UniCensorEM` package.

Usage

```
fit_hybrid2_em(
  x,
  r,
  tc,
  n,
  distribution,
  start = NULL,
  max_iter = 1000,
  tol = 1e-08,
  ...
)

gpci_fit_hybrid2_em(
  x,
  r,
  tc,
  n,
  distribution,
  start = NULL,
  max_iter = 1000,
  tol = 1e-08,
  ...
)

fit_hybrid_ty2_em(
  x,
  r,
  tc,
  n,
  distribution,
  start = NULL,
  max_iter = 1000,
  tol = 1e-08,
  ...
)
```

Arguments

x	Numeric vector of observed failure times (sorted in ascending order).
r	Positive integer scalar specifying the target number of failures.
tc	Positive numeric scalar specifying the fixed censoring time.
n	Positive integer scalar specifying the total initial sample size placed on life test.
distribution	A gpc_dist distribution object created by make_gpc_dist or a predefined distribution constructor.
start	Optional vector of initial parameter values. If NULL, the default parameters from the distribution object are used.
max_iter	Maximum number of EM iterations (default 1000).
tol	Convergence tolerance (default 1e-8).
...	Additional arguments passed to UniCensorEM::em_fit or optimization routines.

Value

The input distribution object of class "gpc_dist" with updated parameter estimates and attributes:

em_fit	The raw emfit object returned by UniCensorEM.
converged	Logical value indicating whether the EM algorithm converged (TRUE or FALSE).
iterations	Integer count of iterations completed until convergence or stopping.
loglik	Numeric scalar log-likelihood value evaluated at the estimated parameters.
se	Named numeric vector of asymptotic standard errors.
vcov	Asymptotic variance-covariance matrix of estimated parameters.
vdata	Validated list of Hybrid Type-II censoring data components.

References

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1), 1-22. doi:10.1111/j.25176161.1977.tb01600.x

Childs, A., Chandrasekar, B., Balakrishnan, N., & Kundu, D. (2003). Exact likelihood inference based on Type-I and Type-II hybrid censored samples from the exponential distribution. *Annals of the Institute of Statistical Mathematics*, 55(2), 319-330. doi:10.1007/BF02517803

Balakrishnan, N., & Kundu, D. (2013). Hybrid censoring: models, inferential methods and applications. *Naval Research Logistics (NRL)*, 60(5), 359-380. doi:10.1002/nav.21545

Examples

```
my_dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fitted <- fit_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10, distribution = my_dist)
print(fitted)
```

gof_test_hybrid2 *Goodness-of-Fit Testing for Hybrid Type-II Censored Data using gofPHCS*

Description

Performs goodness-of-fit testing for Hybrid Type-II censored lifetime data using the gofPHCS package (gofPHCS::gof_test).

Usage

```
gof_test_hybrid2(
  fit = NULL,
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution = NULL,
  statistic = "auto",
  p.method = c("auto", "asymptotic", "montecarlo"),
  nsim = 999,
  seed = NULL,
  conf.level = 0.95,
  ...
)

gof_test(
  fit = NULL,
  x = NULL,
  r = NULL,
  tc = NULL,
  n = NULL,
  distribution = NULL,
  statistic = "auto",
  p.method = c("auto", "asymptotic", "montecarlo"),
  nsim = 999,
  seed = NULL,
  conf.level = 0.95,
  ...
)
```

Arguments

fit	Optional gpc_hybrid2_em_fit object returned by capability_hybrid2_em. If supplied, x, r, tc, n, and distribution are extracted automatically.
x	Numeric vector of observed failure times (required if fit is not supplied).
r	Positive integer target number of failures (required if fit is not supplied).

tc	Positive numeric fixed censoring time (required if fit is not supplied).
n	Positive integer total sample size placed on test (required if fit is not supplied).
distribution	A gpc_dist distribution object (required if fit is not supplied).
statistic	Character string specifying the test statistic (e.g. "auto", "AD", "CvM", "KS"). Defaults to "auto".
p.method	Character string specifying method for calculating p-values: "auto" (default), "asymptotic", or "montecarlo".
nsim	Positive integer scalar specifying number of Monte Carlo replicates. Defaults to 999.
seed	Optional integer seed for reproducibility. Defaults to NULL.
conf.level	Numeric confidence level for test. Defaults to 0.95.
...	Additional arguments passed to gofPHCS::gof_test.

Value

An S3 object of class "gpc_hybrid2_em_gof" containing:

fit	The original gpc_hybrid2_em_fit object, or NULL if raw data was passed.
cens_data	The gofPHCS censored data structure created for Hybrid Type-II data.
gof_result	The returned test result object from gofPHCS::gof_test.
distribution	The tested gpc_dist distribution object.

References

Childs, A., Chandrasekar, B., Balakrishnan, N., & Kundu, D. (2003). Exact likelihood inference based on Type-I and Type-II hybrid censored samples from the exponential distribution. *Annals of the Institute of Statistical Mathematics*, 55(2), 319-330. doi:10.1007/BF02517803

Balakrishnan, N., & Kundu, D. (2013). Hybrid censoring: models, inferential methods and applications. *Naval Research Logistics (NRL)*, 60(5), 359-380. doi:10.1002/nav.21545

Examples

```
dist_exp <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fit <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = dist_exp, USL = 3, LSL = 0)
gof_res <- gof_test_hybrid2(fit, p.method = "montecarlo", nsim = 20)
print(gof_res)
```

gpc_index_hybrid2_em *Retrieve a Single GPCI Value*

Description

Extracts or computes a single capability index from a fitted object.

Usage

```
gpc_index_hybrid2_em(fit, index)
```

Arguments

fit	A gpc_hybrid2_em_fit object.
index	Character name of the index to extract or compute.

Value

Numeric scalar value of the specified index.

Examples

```
my_dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fit <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = my_dist, USL = 3, LSL = 0, fit = FALSE)
gpc_index_hybrid2_em(fit, "Cp")
```

make_gpc_dist *Custom and Predefined Distribution Construction*

Description

Builds a distribution specification object for parameter estimation using the EM algorithm and Generalized Process Capability Indices (GPCIs) under Hybrid Type-II censored lifetime data.

Usage

```
make_gpc_dist(
  name,
  pdf,
  cdf,
  surv = NULL,
  quantile = NULL,
  start,
  param_names = paste0("param", seq_along(start)),
  support = c(0, Inf)
)
```

Arguments

name	Character string giving the name of the distribution.
pdf	Function f(x, theta) for the probability density function (or probability mass function).
cdf	Function F(x, theta) for the cumulative distribution function.
surv	Optional function S(x, theta) for the survival function (defaults to 1 - cdf(x, theta)).
quantile	Optional function Q(p, theta) for the quantile function. If NULL, quantiles are computed by numerical inversion of the CDF.
start	Numeric vector of initial parameter values for estimation.
param_names	Character vector of parameter names.
support	Numeric vector of length 2 giving the domain support of the distribution (default c(0, Inf)).

Value

An object of class "gpc_dist" representing the distribution specification, containing:

name	Character name of the distribution.
pdf	Probability density or mass function.
cdf	Cumulative distribution function.
surv	Survival function.
sf	Alias for survival function.
quantile	Quantile function.
start	Numeric vector of initial starting values.
params	Named list of current parameter values.
param_names	Character vector of parameter names.
support	Numeric vector of length 2 specifying distribution bounds.

References

Maiti, S. S., Saha, M., & Nanda, A. K. (2010). On generalizing process capability indices. *Quality Technology & Quantitative Management*, 7(3), 279-300. doi:10.1080/16843703.2010.11673233

Alotaibi, N., Elbatal, I., Almetwally, E. M., Alyami, S. A., & Al-Moisheer, A. S. (2022). Estimation and confidence intervals of a new PCI CNpmc for logistic-exponential distribution. *Journal of Mathematics*, 2022, 3135264. doi:10.1155/2022/3135264

Examples

```
pdf_exp <- function(x, theta) dexp(x, rate = theta[1])
cdf_exp <- function(x, theta) pexp(x, rate = theta[1])
surv_exp <- function(x, theta) 1 - pexp(x, rate = theta[1])
my_dist <- make_gpc_dist("Exponential", pdf_exp, cdf_exp, surv_exp,
  start = c(1), param_names = "rate")
print(my_dist)
```

mcmc_hybrid2_chain *MCMC Parameter and GPCI Chain Generator for Hybrid Type-II Censored Data*

Description

Generates Bayesian Markov Chain Monte Carlo (MCMC) sampling chains for model parameters and Generalized Process Capability Indices (GPCIs) under Hybrid Type-II censored data using a Metropolis-Hastings within Gibbs sampler. Computes thinned post-burn-in chains, point estimates, bias, MSE, Risk values, HPD intervals at 90%, 95%, and 99% levels, Heidelberger-Welch convergence diagnostics, and convergence probabilities.

Usage

```
mcmc_hybrid2_chain(
  x,
  r,
  tc,
  n,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
 indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  start = NULL,
  priors = NULL,
  length_chain = 5000,
  burn_in = 1000,
  thinning = 5,
  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
)

mcmc_hybrid_ty2_chain(
  x,
  r,
  tc,
  n,
```

```
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
  indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  start = NULL,
  priors = NULL,
  length_chain = 5000,
  burn_in = 1000,
  thinning = 5,
  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
)

gpci_mcmc_hybrid2(
  x,
  r,
  tc,
  n,
  distribution,
  USL,
  LSL,
  target = (USL + LSL)/2,
  indices = c("Cpy", "Cp", "Cpk", "Cpu", "Cpl", "Cpm", "Cpmk", "Spmk", "CpTk", "Cpc",
    "CNpmc"),
  start = NULL,
  priors = NULL,
  length_chain = 5000,
  burn_in = 1000,
  thinning = 5,
  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

x	Numeric vector of observed failure times.
r	Positive integer scalar specifying target number of failures.
tc	Positive numeric scalar specifying fixed censoring time.
n	Positive integer scalar specifying total initial sample size placed on test.
distribution	A gpc_dist distribution object.
USL	Upper Specification Limit.
LSL	Lower Specification Limit.
target	Process target (defaults to (USL + LSL) / 2).
indices	Character vector of capability indices to compute.
start	Named vector of starting parameter values.
priors	List of prior specifications for parameters or a custom log-prior function. Each parameter prior list can specify type ("gamma", "normal", "uniform", "lognormal") and hyperparameters (e.g. shape, rate, mean, sd, min, max).
length_chain	Total length of MCMC chain (default 5000).
burn_in	Number of initial burn-in iterations (default 1000).
thinning	Thinning interval for retaining chain samples (default 5).
u	Weight parameter for Cp(u,v) family (default 1).
v	Weight parameter for Cp(u,v) family (default 1).
mode	Mode of evaluation: "moments" or "quantile".
C0	Cost function parameter (default 1).
C1	Cost function parameter (default 0).
C2	Cost function parameter (default 1).
tolerance_t	Tolerance length (defaults to USL - LSL).
P0	Desired conformance level (default 0.9973002).
LDL	Lower Desired Limit (defaults to LSL).
UDL	Upper Desired Limit (defaults to USL).

Value

An object of class "gpc_hybrid2_em_mcmc" containing:

param_chain	Numeric matrix of thinned post-burn-in parameter samples.
gpci_chain	Numeric matrix of thinned post-burn-in capability index samples.
full_param_chain	Numeric matrix containing the complete unthinned parameter MCMC trajectory.

`param_summary`	Data frame of parameter statistics (`item`, `estimate`, `posterior_mean`, `bias`, `mse`, `risk`).
`gpci_summary`	Data frame of GPCI statistics (`item`, `estimate`, `posterior_mean`, `bias`, `mse`, `risk`).
`hpd_table`	Data frame of High Posterior Density (HPD) intervals at 90%, 95%, and 99% confidence levels.
`convergence`	Data frame of Heidelberger-Welch stationarity and half-width diagnostics and convergence probabilities.
`accept_rates`	Named numeric vector of parameter acceptance rates.
`length_chain`	Total length of MCMC chain run.
`burn_in`	Number of initial iterations discarded as burn-in.
`thinning`	Thinning interval applied.
`distribution`	Original distribution object with initial fitted parameters.
`indices`	Character vector of capability index names.

References

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1), 97-109. doi:10.1093/biomet/57.1.97

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6), 721-741. doi:10.1109/TPAMI.1984.4767596

Heidelberger, P., & Welch, P. D. (1983). Simulation run length control in the presence of an initial transient. *Operations Research*, 31(6), 1109-1144. doi:10.1287/opre.31.6.1109

Maiti, S. S., Saha, M., & Nanda, A. K. (2010). On generalizing process capability indices. *Quality Technology & Quantitative Management*, 7(3), 279-300. doi:10.1080/16843703.2010.11673233

Examples

```
dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
priors <- list(rate = list(type = "gamma", shape = 1, rate = 1))
mcmc_res <- mcmc_hybrid2_chain(x = x, r = 3, tc = 1.0, n = 10, distribution = dist,
                               USL = 3, LSL = 0, priors = priors,
                               length_chain = 100, burn_in = 20, thinning = 2)
print(mcmc_res)
```

plot.gpc_hybrid2_em_ci

Plot Method for Hybrid Type-II Bootstrap CIs

Description

Plots histograms and density estimates of bootstrap replicates for parameters and GPCIs.

Usage

```
## S3 method for class 'gpc_hybrid2_em_ci'
plot(x, ...)
```

Arguments

x	An object of class gpc_hybrid2_em_ci.
...	Additional graphical parameters.

Value

Invisible NULL.

Examples

```
dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fit <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = dist, USL = 3, LSL = 0)
ci <- boot_ci_hybrid2_em(fit, B = 30)
plot(ci)
```

plot.gpc_hybrid2_em_fit

Plot Method for Hybrid Type-II GPCI EM Fit

Description

Visualizes the fitted distribution density curve alongside the specification limits and process target.

Usage

```
## S3 method for class 'gpc_hybrid2_em_fit'
plot(x, ...)
```

Arguments

x	An object of class gpc_hybrid2_em_fit.
...	Additional graphical parameters passed to plot.

Value

Invisible NULL.

Examples

```
dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
fit <- capability_hybrid2_em(x = x, r = 3, tc = 1.0, n = 10,
  distribution = dist, USL = 3, LSL = 0)
plot(fit)
```

---

plot.gpc_hybrid2_em_mcmc

Plot Method for Hybrid Type-II MCMC Chains

Description

Trace and density plots for MCMC parameter and GPCI samples.

Usage

```
## S3 method for class 'gpc_hybrid2_em_mcmc'
plot(x, ...)
```

Arguments

x	An object of class gpc_hybrid2_em_mcmc.
...	Additional graphical parameters.

Value

Invisible NULL.

Examples

```
dist <- dist_exponential(rate = 1)
x <- c(0.2, 0.5, 0.8, 1.1)
mcmc_res <- mcmc_hybrid2_chain(x = x, r = 3, tc = 1.0, n = 10, distribution = dist,
                               USL = 3, LSL = 0, length_chain = 100, burn_in = 20,
                               thinning = 2)
plot(mcmc_res)
```

predefined_distributions

Predefined Distribution Specifications

Description

Built-in helper functions creating "gpc_dist" objects for standard lifetime and process distributions.

Usage

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

dist_weibull(shape = 1, scale = 1)

dist_gamma(shape = 1, rate = 1)

dist_exponential(rate = 1)

dist_exp_exp(alpha = 1, lambda = 1)

dist_logistic_exp(alpha = 1, lambda = 1)

dist_lognormal(meanlog = 0, sdlog = 1)

dist_rayleigh(scale = 1)
```

Arguments

mean	Initial mean for Normal distribution.
sd	Initial standard deviation for Normal distribution.
shape	Initial shape parameter for Weibull or Gamma distribution.
scale	Initial scale parameter for Weibull or Rayleigh distribution.
rate	Initial rate parameter for Gamma or Exponential distribution.
alpha	Initial alpha shape parameter for Exponentiated Exponential or Logistic-Exponential.
lambda	Initial lambda scale parameter for Exponentiated Exponential or Logistic-Exponential.
meanlog	Initial mean on log scale for Lognormal distribution.
sdlog	Initial standard deviation on log scale for Lognormal distribution.

Value

A "gpc_dist" object containing the corresponding distribution functions and starting parameters.

Examples

```
d_norm <- dist_normal()
d_weib <- dist_weibull()
d_gamma <- dist_gamma()
d_exp <- dist_exponential()
d_exp_exp <- dist_exp_exp()
d_log_exp <- dist_logistic_exp()
d_lnorm <- dist_lognormal()
d_rayleigh <- dist_rayleigh()
```

---

print.gpc_dist *Print Method for Distribution Objects*

Description

Print Method for Distribution Objects

Usage

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

Arguments

x	A gpc_dist distribution object.
...	Additional arguments passed to print methods.

Value

The input object x, returned invisibly.

---

print.gpc_hybrid2_em_ci *Print Method for Hybrid Type-II Bootstrap CIs*

Description

Print Method for Hybrid Type-II Bootstrap CIs

Usage

```
## S3 method for class 'gpc_hybrid2_em_ci'
print(x, ...)
```

Arguments

x	Object of class gpc_hybrid2_em_ci.
...	Additional arguments passed to print methods.

Value

The input object x, returned invisibly.

---

print.gpc_hybrid2_em_fit

Print Method for Hybrid Type-II GPCI EM Fit

Description

Print Method for Hybrid Type-II GPCI EM Fit

Usage

```
## S3 method for class 'gpc_hybrid2_em_fit'
print(x, ...)
```

Arguments

x	Object of class gpc_hybrid2_em_fit.
...	Additional arguments passed to print methods.

Value

The input object x, returned invisibly.

---

print.gpc_hybrid2_em_gof

Print Method for Hybrid Type-II Goodness-of-Fit

Description

Print Method for Hybrid Type-II Goodness-of-Fit

Usage

```
## S3 method for class 'gpc_hybrid2_em_gof'
print(x, ...)
```

Arguments

x	An object of class gpc_hybrid2_em_gof.
...	Additional print arguments.

Value

The input object x, returned invisibly.

---

print.gpc_hybrid2_em_mcmc *Print Method for MCMC Chain Generator Result*

Description

Print Method for MCMC Chain Generator Result

Usage

```
## S3 method for class 'gpc_hybrid2_em_mcmc'
print(x, ...)
```

Arguments

x	Object of class gpc_hybrid2_em_mcmc.
...	Additional arguments passed to print methods.

Value

The input object x, returned invisibly.

---

print.gpc_sim_hybrid2_em *Print Method for Hybrid Type-II Simulation*

Description

Print Method for Hybrid Type-II Simulation

Usage

```
## S3 method for class 'gpc_sim_hybrid2_em'
print(x, ...)
```

Arguments

x	An object of class gpc_sim_hybrid2_em.
...	Additional print arguments.

Value

The input object, invisibly.

---

simulate_hybrid2_data *Simulate Hybrid Type-II Censored Lifetime Data*

Description

Simulates ordered failure times under the Hybrid Type-II censoring scheme from a specified distribution.

Usage

```
simulate_hybrid2_data(distribution, n, r, tc)
```

Arguments

distribution	A gpc_dist distribution object.
n	Positive integer total number of units placed on test.
r	Positive integer target number of failures.
tc	Positive numeric fixed censoring time.

Value

A numeric vector of observed failure times up to time $T^* = \max(X_{r:n}, T_c)$, with attributes: "t_star" (censoring termination time), "r", "tc", "n", "d" (observed failures count), and "n_cens" (censored count).

Examples

```
dist <- dist_weibull(shape = 2, scale = 5)
sim_data <- simulate_hybrid2_data(distribution = dist, n = 15, r = 8, tc = 4.0)
print(sim_data)
```

summary.gpc_hybrid2_em_ci *Summary Method for Hybrid Type-II Bootstrap CIs*

Description

Summary Method for Hybrid Type-II Bootstrap CIs

Usage

```
## S3 method for class 'gpc_hybrid2_em_ci'
summary(object, ...)
```

Arguments

object	Object of class gpc_hybrid2_em_ci.
...	Additional arguments passed to summary methods.

Value

The input object object, returned invisibly.

summary.gpc_hybrid2_em_fit *Summary Method for Hybrid Type-II GPCI EM Fit*

Description

Summary Method for Hybrid Type-II GPCI EM Fit

Usage

```
## S3 method for class 'gpc_hybrid2_em_fit'
summary(object, ...)
```

Arguments

object	Object of class gpc_hybrid2_em_fit.
...	Additional arguments passed to summary methods.

Value

The input object object, returned invisibly.

`summary.gpc_hybrid2_em_gof`
Summary Method for Hybrid Type-II Goodness-of-Fit

Description

Summary Method for Hybrid Type-II Goodness-of-Fit

Usage

```
## S3 method for class 'gpc_hybrid2_em_gof'
summary(object, ...)
```

Arguments

`object`	An object of class `gpc_hybrid2_em_gof`.
`...`	Additional arguments passed to summary methods.

Value

The input object `object`, returned invisibly.

`summary.gpc_hybrid2_em_mcmc`
Summary Method for MCMC Chain Generator Result

Description

Summary Method for MCMC Chain Generator Result

Usage

```
## S3 method for class 'gpc_hybrid2_em_mcmc'
summary(object, ...)
```

Arguments

`object`	Object of class `gpc_hybrid2_em_mcmc`.
`...`	Additional arguments passed to summary methods.

Value

The input object `object`, returned invisibly.

summary.gpc_sim_hybrid2_em *Summary Method for Hybrid Type-II Simulation*

Description

Summary Method for Hybrid Type-II Simulation

Usage

```
## S3 method for class 'gpc_sim_hybrid2_em'
summary(object, ...)
```

Arguments

object	An object of class gpc_sim_hybrid2_em.
...	Additional arguments.

Value

The input object, invisibly.

Index