Package 'neuralsbi'

August 3, 2026

Title Neural Simulation-Based Inference

Version 0.3.2

Description
A native R implementation of neural simulation-based inference, focused on Neural Posterior Estimation. Given a prior over parameters and a simulator, 'neuralsbi' trains a conditional neural density estimator to approximate the Bayesian posterior, enabling amortized, likelihood-free inference. Neural estimators run on the 'torch' back end. It targets applied researchers who want an approachable interface with sensible defaults and built-in posterior diagnostics.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, utils

Suggests torch (>= 0.11.0), testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

URL https://pedroliman.github.io/neuralsbi/,
https://github.com/pedroliman/neuralsbi

BugReports https://github.com/pedroliman/neuralsbi/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Pedro Nascimento de Lima [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9057-198X>)

Maintainer Pedro Nascimento de Lima <plima@rand.org>

Repository CRAN

Date/Publication 2026-08-03 18:10:07 UTC

Contents

`c2st` *Classifier two-sample test (C2ST)*

Description

Trains a logistic-regression classifier to distinguish samples in `x` from samples in `y` using cross-validation. A test accuracy near 0.5 means the two sample sets are indistinguishable (good); near 1.0 means they differ. This is the standard SBI metric for comparing an estimated posterior to a reference (e.g. an analytic posterior or long-run MCMC draws).

Usage

```
c2st(x, y, n_folds = 5L, seed = NULL)
```

Arguments

x, y	Matrices of samples (rows = draws, cols = dimensions).
n_folds	Number of cross-validation folds.
seed	Optional seed.

Value

A list with mean CV accuracy and per-fold accuracies.

---

density_estimator *Conditional density estimators*

Description

A conditional density estimator learns $q_\phi(\theta \mid x)$. In neuralsbi every estimator is trained in *standardized* space and exposes two generics:

Details

- de_log_prob(de, theta, x) – log density of theta given x
- de_sample(de, x, n) – draw n parameter vectors given a single x

Two estimators ship today:

- "mdn" – a Mixture Density Network (neural network -> Gaussian mixture), the workhorse, requires the torch back end.
- "linear_gaussian" – a closed-form conditional Gaussian baseline (least-squares mean, residual covariance). No neural network, no torch. It is exact for linear-Gaussian simulators and doubles as a fast baseline and a regression-test oracle.

---

diagnostics *Posterior diagnostics*

Description

Tools to check whether a trained posterior is trustworthy:

Details

- sbc() – Simulation-Based Calibration rank statistics
- expected_coverage() – nominal vs. empirical credible-interval coverage
- c2st() – classifier two-sample test between two sample sets
- posterior_predictive() – draw data from the fitted posterior

`embedding_mlp` *Embedding (summary) networks for structured observations*

Description

Raw observations are often high-dimensional or structured (a time series, a set of summary statistics, an image) where feeding `x` straight into the density estimator wastes capacity. An embedding network learns a low- dimensional summary $h = f_\psi(x)$ jointly with the density estimator, so the conditioning path becomes $q_\phi(\theta \mid f_\psi(x))$. This mirrors `sbi`'s `embedding_net` argument.

Usage

```
embedding_mlp(output_dim = 16L, hidden = c(64L, 64L))
```

Arguments

`output_dim`	Number of summary features the network emits. This is the effective data dimension the density estimator conditions on.
`hidden`	Integer vector of hidden-layer widths (ReLU between layers). An empty vector gives a single linear map to `output_dim`.

Details

`embedding_mlp()` builds a multilayer-perceptron summary network: a stack of fully connected ReLU layers mapping the (standardized) data to a vector of `output_dim` features. Pass the result to `npe()` via `embedding_net`; it is trained end to end with the estimator and its parameters live inside the fitted network, so sampling and `log_prob` route through it automatically.

The embedding consumes the standardized data (the same z-scoring `npe()` applies to `x` without an embedding), which keeps the summary network's inputs on a common scale. Standardization of the *features* is intentionally left to the network itself; the estimators operate on the raw embedding output.

Value

An `nsbi_embedding` specification. It carries no torch objects (the network is built lazily at fit time), so it is safe to construct without `torch` installed.

See Also

`npe()`

Examples

```
emb <- embedding_mlp(output_dim = 8, hidden = c(64, 64))
# fit <- npe(prior, simulator, density_estimator = "maf", embedding_net = emb)
```

expected_coverage *Expected coverage of central credible intervals*

Description

Uses the SBC ranks to compare nominal credible levels with the empirical fraction of trials in which the true parameter falls inside the corresponding central interval. Well-calibrated posteriors lie on the diagonal.

Usage

```
expected_coverage(sbc_result, levels = seq(0.05, 0.95, by = 0.05))
```

Arguments

sbc_result	An nsbi_sbc object from sbc().
levels	Nominal credibility levels to evaluate.

Value

A data frame with nominal and per-parameter empirical coverage.

log_prob *Posterior log-density*

Description

Posterior log-density

Usage

```
log_prob(post, theta, x = NULL, normalize = TRUE, n_normalization = 10000L)
```

Arguments

post	An nsbi_posterior object.
theta	Matrix (or vector) of parameter values to evaluate.
x	Observation to condition on (defaults to x_obs).
normalize	For bounded priors, renormalize by the estimated acceptance probability and return -Inf outside the prior support.
n_normalization	Number of draws used to estimate the normalizing (acceptance) constant when normalize = TRUE.

Value

Numeric vector of log posterior densities.

map_estimate *Maximum a posteriori (MAP) estimate*

Description

Starts from the best of a set of posterior draws and refines with a derivative-free optimizer.

Usage

```
map_estimate(post, x = NULL, n_init = 1000L)
```

Arguments

post	An nsbi_posterior object.
x	Observation to condition on (defaults to x_obs).
n_init	Number of initial draws used to seed the search.

Value

Numeric vector: the MAP parameter estimate.

npe *Neural Posterior Estimation (NPE)*

Description

npe() is the main entry point. Given a prior and either a simulator (which it will call) or a set of pre-computed simulations (theta, x), it trains a conditional density estimator whose output directly approximates the posterior $p(\theta \mid x)$. This is single-round, *amortized* NPE: after training once, you can condition on any observation without re-simulating.

Usage

```
npe(
  prior,
  simulator = NULL,
  n_simulations = 1000,
  theta = NULL,
  x = NULL,
  density_estimator = c("maf", "mdn", "nsf", "linear_gaussian"),
  n_components = 10L,
  n_transforms = 5L,
  hidden = c(50L, 50L),
  embedding_net = NULL,
  max_epochs = 2000L,
```

```
  batch_size = 200L,
  lr = 5e-04,
  validation_fraction = 0.1,
  patience = 20L,
  n_restarts = 1L,
  clip_grad_norm = 5,
  standardize = TRUE,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`prior`	An `nsbi_prior` (see `prior_uniform()`, `prior_normal()`).
`simulator`	A function mapping an `n x dim` matrix of parameters to an `n x d` matrix of simulated data. Ignored if `theta` and `x` are given.
`n_simulations`	Number of prior draws to simulate when `simulator` is used and `theta`/`x` are not supplied.
`theta, x`	Optional pre-computed simulations. If supplied, `simulator` and `n_simulations` are ignored.
`density_estimator`	One of `"maf"` (Masked Autoregressive Flow, needs `torch`; the default, matching Python `sbi`), `"mdn"` (neural Mixture Density Network, needs `torch`), `"nsf"` (Neural Spline Flow, needs `torch`), or `"linear_gaussian"` (closed-form baseline, no `torch`), or a function `function(theta, x)` returning a fitted estimator.
`n_components, hidden`	MDN settings: number of mixture components (default 10, as in `sbi`) and a vector of hidden-layer widths.
`n_transforms`	MAF/NSF setting: number of stacked autoregressive transforms (default 5, as in `sbi`).
`embedding_net`	Optional summary network built with `embedding_mlp()`. When supplied, the neural estimators condition on the learned features $f_\psi(x)$ instead of the raw data, training the embedding jointly. Ignored (with a warning) by `"linear_gaussian"`.
`max_epochs, batch_size, lr, validation_fraction, patience`	Neural training controls (Adam optimizer, early stopping on validation loss). The defaults (`batch_size = 200`, `lr = 5e-4`, `validation_fraction = 0.1`, `patience = 20`) match Python `sbi`; `max_epochs` is a high guard cap that early stopping normally reaches first.
`n_restarts`	Train this many independently initialized networks and keep the one with the best validation loss (guards against bad initializations and MDN mode collapse).
`clip_grad_norm`	Maximum gradient norm during training (`Inf` disables clipping). The learning rate also decays 2x after 10 epochs without validation improvement.
`standardize`	Whether to z-score `theta` and `x` before training (strongly recommended; default `TRUE`).

seed	Optional integer seed for reproducibility.
verbose	Print training progress.
...	Passed to the density estimator.

Value

An object of class nsbi_npe. Turn it into a usable posterior with posterior(), or sample directly with sample().

Examples

```
prior <- prior_uniform(c(-2, -2, -2), c(2, 2, 2))
simulator <- function(theta) theta + 1 + matrix(rnorm(length(theta), sd = 0.1),
                                                nrow = nrow(theta))
fit <- npe(prior, simulator, n_simulations = 2000,
           density_estimator = "linear_gaussian")
post <- posterior(fit, x_obs = c(0.8, 0.6, 0.4))
draws <- sample(post, 1000)
```

---

npe_sequential *Sequential NPE with truncated-prior proposals (TSNPE)*

Description

Multi-round NPE targeting a single observation x_obs. Single-round npe() spends its simulation budget across the whole prior; when only one observation matters, most of those simulations land in regions the posterior never visits. npe_sequential() implements truncated sequential NPE (TSNPE, Deistler et al. 2022): after each round the prior is truncated to the highest-probability region of the current posterior estimate, and the next round's parameters are drawn from that truncated prior. Because every proposal is proportional to the prior on its support, the standard NPE loss stays valid – no importance-weight or atomic correction is needed, which is what makes TSNPE the simplest correct sequential scheme.

Usage

```
npe_sequential(
  prior,
  simulator,
  x_obs,
  n_rounds = 2L,
  n_simulations = 1000L,
  density_estimator = c("maf", "mdn", "nsf", "linear_gaussian"),
  epsilon = 1e-04,
  n_truncation_samples = 5000L,
  max_proposal_batches = 200L,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`prior`	An `nsbi_prior` (see `prior_uniform()`, `prior_normal()`).
`simulator`	A function mapping an `n x dim` matrix of parameters to an `n x d` matrix of simulated data.
`x_obs`	The observation to target. Sequential inference concentrates simulations around the posterior for this observation.
`n_rounds`	Number of rounds. Round 1 is ordinary single-round NPE.
`n_simulations`	Simulation budget per round; either a scalar or a vector of length `n_rounds`.
`density_estimator`	Passed to `npe()` each round.
`epsilon`	Mass cut for the truncation: the proposal region is the `1 - epsilon` highest-probability region of the current posterior.
`n_truncation_samples`	Posterior draws used to locate the truncation threshold each round.
`max_proposal_batches`	Cap on rejection-sampling batches per round.
`seed`	Optional integer seed for reproducibility.
`verbose`	Print per-round progress.
`...`	Passed to `npe()` (estimator and training settings).

Details

The rounds accumulate: each round's estimator is trained on all simulations so far. The final fit is returned as an `nsbi_npe` (subclass `nsbi_snpe`) and works with `posterior()`, `sample()` and the diagnostics, but unlike single-round NPE it is *not* amortized: it is only trustworthy at (or very near) `x_obs`.

Proposal draws are obtained by rejection: prior candidates are kept when their posterior log-density clears the `epsilon`-quantile threshold of the current posterior's own draws. If the posterior is much narrower than the prior the acceptance rate falls; the round then stops after `max_proposal_batches` batches and continues with the draws it has, with a warning.

Value

An object of class `c("nsbi_snpe", "nsbi_npe")` with a `rounds` field recording per-round budgets, acceptance rates, and thresholds.

References

Deistler, Goncalves & Macke (2022), "Truncated proposals for scalable and hassle-free simulation-based inference", NeurIPS. doi:10.48550/arXiv.2210.04815

Examples

```
prior <- prior_normal(mean = c(0, 0), sd = 1)
simulator <- function(theta) theta + matrix(rnorm(length(theta), sd = 0.3),
                                            nrow = nrow(theta))
fit <- npe_sequential(prior, simulator, x_obs = c(0.5, -0.5),
                      n_rounds = 2, n_simulations = 1000,
                      density_estimator = "linear_gaussian")
post <- posterior(fit, x_obs = c(0.5, -0.5))
draws <- sample(post, 1000)
```

`pairplot` *Visualize posterior samples*

Description

A dependency-free (base graphics) pair plot: 1-D marginal densities on the diagonal and 2-D scatter/contours off-diagonal, with optional markers for a reference (e.g. true) parameter value. Analogous to `sbi`'s `pairplot`.

Usage

```
pairplot(
  samples,
  truth = NULL,
  labels = NULL,
  limits = NULL,
  col = grDevices::adjustcolor("steelblue", 0.4),
  ...
)
```

Arguments

`samples`	A matrix of posterior draws (rows = draws), or an `nsbi_samples` object.
`truth`	Optional reference parameter vector to overlay.
`labels`	Optional parameter labels.
`limits`	Optional list/matrix of per-parameter c(lo, hi) axis limits.
`col`	Point colour.
`...`	Passed to plotting calls.

Value

Invisibly, the samples.

plot_coverage *Plot nominal vs. empirical credible-interval coverage*

Description

Well-calibrated posteriors lie on the diagonal. Curves above the diagonal mean the posterior is too wide (conservative); below means overconfident. A shaded band shows the Monte-Carlo uncertainty from the finite number of SBC trials.

Usage

```
plot_coverage(sbc_result, levels = seq(0.05, 0.95, by = 0.05))
```

Arguments

sbc_result	An nsbi_sbc object from sbc().
levels	Nominal credibility levels to evaluate.

Value

Invisibly, the coverage data frame from expected_coverage().

plot_posterior_predictive *Plot posterior predictive checks*

Description

Compares data simulated from posterior parameter draws (see posterior_predictive()) with the observed data, one marginal histogram per data dimension with the observation marked. If the observation falls in the tails of the predictive distribution, the model (or the fit) does not reproduce the data it is conditioned on.

Usage

```
plot_posterior_predictive(pred, x_obs, labels = NULL, bins = 30L)
```

Arguments

pred	A matrix of predictive draws from posterior_predictive().
x_obs	The observed data vector the posterior was conditioned on.
labels	Optional labels for the data dimensions.
bins	Number of histogram bins.

Value

Invisibly, the per-dimension predictive quantile of the observation.

plot_sbc *Plot an SBC rank histogram*

Description

Uniform bars indicate calibration; a U shape means the posterior is too narrow (overconfident); an inverted-U means it is too wide.

Usage

```
plot_sbc(sbc_result, param = 1L, bins = 20L)
```

Arguments

sbc_result	An nsbi_sbc object from sbc().
param	Which parameter index to plot (default 1).
bins	Number of histogram bins.

Value

Invisibly, the rank vector.

plot_tarp *Plot TARP expected coverage*

Description

Draws the expected coverage probability (ECP) curve from tarp() against the nominal credibility level. A calibrated posterior lies on the diagonal; a curve above the diagonal means the posterior is too wide (conservative), below means overconfident. The shaded band shows the Monte-Carlo uncertainty from the finite number of TARP trials.

Usage

```
plot_tarp(tarp_result)
```

Arguments

tarp_result	An nsbi_tarp object from tarp().

Value

Invisibly, a data frame with nominal and ecp columns.

posterior *Posterior objects*

Description

A posterior wraps a trained npe() fit together with (optionally) a default observation x_obs. It knows how to draw posterior samples, evaluate the posterior log-density, and find the maximum-a-posteriori (MAP) estimate. All transforms between standardized training space and the original parameter space are handled internally.

Usage

```
posterior(fit, x_obs = NULL)
```

Arguments

fit	An nsbi_npe object from npe().
x_obs	Optional default observation to condition on. If supplied it becomes the default x for sample(), log_prob() and map_estimate().

Details

For bounded priors, samples that fall outside the prior support are rejected ("leakage" correction), and log_prob() is renormalized by the estimated acceptance probability so it integrates to one over the support.

Value

An nsbi_posterior object.

posterior_predictive *Posterior predictive draws*

Description

Samples parameters from the posterior and pushes them back through the simulator, giving predictive data to compare against the observation.

Usage

```
posterior_predictive(post, simulator, n = 1000L, x = NULL)
```

Arguments

post	An nsbi_posterior object.
simulator	The simulator.
n	Number of predictive draws.
x	Observation to condition on (defaults to x_obs).

Value

An n x d matrix of simulated data from posterior parameter draws.

---

priors *Priors for neural simulation-based inference*

Description

A prior in neuralsbi is a lightweight object (class nsbi_prior) that knows how to (a) draw samples and (b) evaluate its log-density. Bounded priors also carry lower/upper support limits, which are used to reject out-of-support posterior samples ("leakage" correction).

---

prior_custom *Build a prior from arbitrary sampling / density functions*

Description

Build a prior from arbitrary sampling / density functions

Usage

```
prior_custom(sample_fn, log_prob_fn = NULL, dim, lower = NULL, upper = NULL)
```

Arguments

sample_fn	Function function(n) returning an n x dim matrix.
log_prob_fn	Function function(theta) returning a length-n vector of log densities. Optional; required only for methods/diagnostics that need it.
dim	Number of parameters.
lower, upper	Optional support bounds (numeric vectors) enabling out-of-support rejection.

Value

An nsbi_prior object.

prior_normal *Independent normal prior*

Description

Independent normal prior

Usage

```
prior_normal(mean, sd = 1)
```

Arguments

mean	Numeric vector of means (one per parameter).
sd	Numeric scalar or vector of standard deviations.

Value

An nsbi_prior object.

Examples

```
prior <- prior_normal(mean = c(0, 0), sd = 1)
```

prior_uniform *Box-uniform (independent uniform) prior*

Description

Box-uniform (independent uniform) prior

Usage

```
prior_uniform(low, high)
```

Arguments

low	Numeric vector of lower bounds (one per parameter).
high	Numeric vector of upper bounds (one per parameter).

Value

An nsbi_prior object.

Examples

```
prior <- prior_uniform(low = c(-2, -2, -2), high = c(2, 2, 2))
theta <- sample_prior(prior, 5)
```

sample *Draw samples (S3 generic)*

Description

neuralsbi turns base::sample() into an S3 generic so that sample(posterior, n) reads the way statisticians expect. For any object without a dedicated method (vectors, etc.) this falls back to base::sample() unchanged.

Usage

```
sample(x, ...)

## Default S3 method:
sample(x, ...)
```

Arguments

x	Object to sample from.
...	Passed on to methods / base::sample().

Value

Whatever the dispatched method returns. The default method returns the result of base::sample(); sample.nsbi_posterior() returns an n x dim matrix of posterior draws.

sample.nsbi_posterior *Sample from a posterior*

Description

Sample from a posterior

Usage

```
## S3 method for class 'nsbi_posterior'
sample(x, size = 1000, n = size, obs = NULL, max_sampling_batches = 100L, ...)
```

Arguments

x	An nsbi_posterior object (named x to satisfy the sample() generic).
size, n	Number of posterior draws (n is an alias for size).
obs	Observation to condition on (defaults to the posterior's x_obs).
max_sampling_batches	Safety cap on rejection-sampling rounds for bounded priors.
...	Unused.

Value

An `n x dim` matrix of posterior draws (class `nsbi_samples`).

`sample_posterior` *Sample from a posterior (non-generic alias)*

Description

Identical to `sample(post, n)`; provided for users who prefer not to rely on the generic.

Usage

```
sample_posterior(post, n = 1000, obs = NULL, ...)
```

Arguments

`post`	An `nsbi_posterior` object.
`n`	Number of posterior draws.
`obs`	Observation to condition on (defaults to the posterior's `x_obs`).
`...`	Passed to `sample.nsbi_posterior()`.

Value

An `n x dim` matrix of posterior draws.

`sample_prior` *Draw samples from a prior*

Description

Draw samples from a prior

Usage

```
sample_prior(prior, n)
```

Arguments

`prior`	An `nsbi_prior` object.
`n`	Number of samples.

Value

An `n x dim` matrix of parameter draws.

sbc *Simulation-Based Calibration (SBC)*

Description

Repeatedly draws a "true" parameter from the prior, simulates data, and ranks the true parameter within posterior samples conditioned on that data. If the posterior is well calibrated, the ranks are uniformly distributed.

Usage

```
sbc(
  fit,
  simulator,
  prior = fit$prior,
  n_sbc = 200L,
  n_posterior_samples = 1000L,
  seed = NULL
)
```

Arguments

`fit`	An `nsbi_npe` fit (amortized posterior).
`simulator`	The simulator used for inference.
`prior`	The prior used for inference (defaults to `fit$prior`).
`n_sbc`	Number of SBC trials (fresh (theta, x) pairs).
`n_posterior_samples`	Posterior draws per trial (rank resolution).
`seed`	Optional seed.

Value

An object of class `nsbi_sbc` with the rank matrix and a per-parameter uniformity test.

simulate_for_sbi *Run a simulator over prior draws*

Description

Run a simulator over prior draws

Usage

```
simulate_for_sbi(simulator, prior, n, seed = NULL, verbose = FALSE)
```

Arguments

simulator	A function mapping an n x dim matrix of parameters to an n x d matrix of simulated data. Ignored if theta and x are given.
prior	An nsbi_prior (see prior_uniform(), prior_normal()).
n	Number of simulations.
seed	Optional integer seed for reproducibility.
verbose	Print training progress.

Value

A list with theta (n x dim) and x (n x d) matrices.

---

summaries *Summaries and tidy accessors*

Description

summary() methods for fits, posteriors, and samples, plus as.data.frame() for posterior draws so results drop straight into data-frame workflows (dplyr, ggplot2, ...).

Usage

```
## S3 method for class 'nsbi_samples'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'nsbi_samples'
summary(object, probs = c(0.025, 0.25, 0.5, 0.75, 0.975), ...)

## S3 method for class 'nsbi_posterior'
summary(object, n = 1000L, x = NULL, ...)

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

Arguments

x	Observation to condition on (defaults to the posterior's x_obs); for as.data.frame(), the samples object.
row.names, optional	Standard as.data.frame() arguments.
...	Additional arguments passed to methods.
object	An nsbi_samples matrix from sample(), an nsbi_posterior, or an nsbi_npe fit.
probs	Quantiles to report.
n	Number of draws used to summarize a posterior.

Value

For samples and posteriors, a data frame with one row per parameter (mean, sd, and quantiles). For fits, an invisible list of training metadata.

`tarp` *TARP expected coverage*

Description

Tests of Accuracy with Random Points (Lemos et al. 2023). For each trial a true parameter is drawn from the prior, data are simulated, and posterior samples are drawn conditioned on those data. Given a random reference point, the fraction of posterior samples closer to the reference than the truth is the credibility level of the smallest distance-based credible region that contains the truth. For a calibrated posterior these fractions are uniform, so the expected coverage probability (ECP) at credibility level alpha equals alpha.

Usage

```
tarp(
  fit,
  simulator,
  prior = fit$prior,
  n_tarp = 200L,
  n_posterior_samples = 1000L,
  references = c("uniform", "prior"),
  seed = NULL
)
```

Arguments

`fit`	An `nsbi_npe` fit (amortized posterior).
`simulator`	The simulator used for inference.
`prior`	The prior used for inference (defaults to `fit$prior`).
`n_tarp`	Number of TARP trials (fresh (theta, x) pairs).
`n_posterior_samples`	Posterior draws per trial.
`references`	How to draw reference points: `"uniform"` (default, uniform over the hyper-rectangle spanned by the true parameter draws, as in the paper) or `"prior"` (draws from the prior).
`seed`	Optional seed.

Details

Unlike `sbc()`, which ranks each parameter marginally, TARP is a *joint* test: it can detect posteriors whose marginals are calibrated but whose correlation structure is wrong. Distances are computed after z-scoring each parameter (using the spread of the true draws), so parameters on different scales contribute comparably.

Value

An object of class `nsbi_tarp` with the per-trial coverage values and the ECP curve. Plot it with `plot_tarp()`.

References

Lemos, Coogan, Hezaveh & Perreault-Levasseur (2023), "Sampling-based accuracy testing of posterior estimators for general inference", ICML. doi:10.48550/arXiv.2302.03026

---

tasks *Benchmark tasks*

Description

Standard simulation-based-inference benchmark tasks, following the definitions in the `sbibm` benchmark suite. Each task bundles a prior, a simulator, and (where one exists) an analytic reference posterior, so the same object drives unit tests, calibration studies, and the benchmark harness in `inst/benchmarks/`.

- `task_gaussian_linear()` – conjugate Gaussian; analytic posterior.
- `task_two_moons()` – crescent-shaped, bimodal posterior.
- `task_slcp()` – 5 parameters, 8-dimensional data, strongly non-Gaussian posterior.
- `task_sir()` – SIR epidemic dynamics with log-normal priors; no closed-form posterior (verify with SBC / coverage).

Usage

```
task_gaussian_linear(dim = 10L, prior_var = 0.1, noise_var = 0.1)

task_two_moons()

task_slcp()

task_sir(N = 1e6, days = 160, n_points = 10L, n_obs_draws = 1000L)
```

Arguments

`dim`	Parameter/data dimension for the Gaussian linear task (default 10).
`prior_var`, `noise_var`	Prior and likelihood variances for the Gaussian linear task.
`N`, `days`, `n_points`, `n_obs_draws`	SIR task: population size, horizon in days, number of observation times, and binomial trials per observation.

Value

A list of class `nsbi_task` with elements `name`, `prior`, `simulator`, `dim_theta`, `dim_x`, and optionally `reference_posterior(x_obs, n)` returning exact posterior draws.

`within_support` *Test whether parameters lie within the prior support*

Description

Test whether parameters lie within the prior support

Usage

```
within_support(prior, theta)
```

Arguments

`prior`	An `nsbi_prior` object.
`theta`	A matrix (or vector) of parameters.

Value

Logical vector, one entry per row of `theta`.

Index