Package 'qslice'

October 22, 2026

Title Implementations of Various Slice Samplers

Version 0.4.0

Description Implementations of the quantile slice sampler of Heiner et al. (2026+, in preparation) <doi:10.48550/arXiv.2407.12608> as well as other popular slice samplers are provided. Helper functions for specifying pseudo-
target distributions are included, both for diagnostics and for tuning the quantile slice sampler. Other implemented methods include the generalized elliptical slice sampler of Nishihara et al. (2014) <https://jmlr.org/papers/v15/nishihara14a.html>,
latent slice sampler of Li and Walker (2023) <doi:10.1016/j.csda.2022.107652>, and stepping-out slice sampler of Neal (2003) <doi:10.1214/aos/1056562461>,
as well as independence Metropolis-Hastings and random-walk Metropolis samplers.

License MIT + file LICENSE | Apache License 2.0

Depends R (>= 4.1.0)

Encoding UTF-8

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Matthew Heiner [aut] (ORCID: <https://orcid.org/0000-0002-7944-5517>),
David B. Dahl [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8173-1547>),
Sam Johnson [aut]

Maintainer David B. Dahl <dahl@stat.byu.edu>

Repository CRAN

Date/Publication 2026-10-22 00:00:09 UTC

Contents

auc 2
imh_pseudo 3
lapprox 4

pseudo_condseq 5
pseudo_condseq_XfromU 7
pseudo_list 9
pseudo_opt 11
rwm_norm 14
slice_elliptical 15
slice_elliptical_mv 16
slice_genelliptical 17
slice_genelliptical_mv 19
slice_hyperrect 20
slice_latent 21
slice_quantile 23
slice_quantile_mv 24
slice_quantile_mv_seq 25
slice_stepping_out 27
utility_pseudo 29
utility_shrinkslice 31

Index **34**

---

auc *Area Under the Curve (histogram)*

Description

Function auc() is deprecated. Use utility_shrinkslice(..., utility_type = "AUC").

Usage

```
auc(u = NULL, x = NULL, y = NULL, nbins = 30)
```

Arguments

u	Numeric vector of samples supported on unit interval with which to create a histogram (use u = NULL if x and y are supplied).
x	Numeric vector of histogram locations. (Not used if u is supplied).
y	Numeric vector of histogram heights OR function evaluating the curve for a given value of u supported on (0,1). (Not used if u is supplied).
nbins	Number of histogram bins to use (defaults to 30).

Value

Scalar value of the AUC (see utility_shrinkslice).

imh_pseudo *Independence Metropolis-Hastings*

Description

Single update using an independence Metropolis-Hastings sampler with a pseudo-target as the proposal distribution.

Usage

```
imh_pseudo(x, log_target, pseudo, diagnostics = NULL)
```

Arguments

x	The current state (scalar or numeric vector).
log_target	A function taking a scalar or numeric vector that evaluates the log-target density, returning a numeric scalar.
pseudo	List specifying the pseudo-target (proposal distribution). If the list length is equal to the number of dimensions in x, each element is itself a list that specifies the pseudo-target independently for each corresponding dimension with functions ld that evaluates the log density for that dimension, and q that evaluates the quantile (inverse-CDF) function for that dimension. If the dimension of x is one, then supply only the inner list specifying the single pseudo-target. If x is a vector but a single pseudo-target is supplied, the list must contain a log-density function ld that accepts a vector, and a function r that takes no arguments and generates a single multivariate draw from the proposal distribution.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing the new state, x, whether the proposed value was accepted, logical accpt, and the number of evaluations of the target function, nEvaluations.

Examples

```
lf <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
n_iter <- 10 # set to 1e3 for more complete illustration
draws <- matrix(0.2, nrow = n_iter, ncol = 2)
nAccpt <- 0L
pseudo <- list( list(ld = function(x) dbeta(x, 2, 2, log = TRUE),
                     q = function(u) qbeta(u, 2, 2)),
                list(ld = function(x) dbeta(x, 2, 2, log = TRUE),
                     q = function(u) qbeta(u, 2, 2))
)
```

```
for (i in seq.int(2, n_iter)) {
 out <- imh_pseudo(draws[i - 1, ], log_target = lf, pseudo = pseudo)
 draws[i,] <- out$x
 nAccpt <- nAccpt + out$accpt
 cat(i, '\r')
}
nAccpt / (nrow(draws) - 1)
plot(draws[,1], draws[,2], xlim = c(0, 1))
hist(draws[,1], freq = FALSE); curve(dbeta(x, 3, 4), col = "blue", add = TRUE)
hist(draws[,2], freq = FALSE); curve(dbeta(x, 5, 3), col = "blue", add = TRUE)
```

`lapprox` *Pseudo-target from Laplace Approximation*

Description

Find the location and scale for an approximating pseudo-target via Laplace approximation.

Usage

```
lapprox(
  log_target,
  init,
  family = "t",
  params = NULL,
  sc_adj = 1,
  lb = -Inf,
  ub = Inf,
  maxit = 100,
  ...
)
```

Arguments

`log_target`	Univariate function evaluating the unnormalized log density to approximate.
`init`	Numeric scalar for an initial value (used in optimization).
`family`	String specifying the family of distributions for the pseudo-target. Can be any of the families accepted by pseudo_list.
`params`	List specifying the parameters for the pseudo-target to be used. The location and scale parameters will be replaced with the Laplace approximation and others (e.g., degrees of freedom) will be retained.
`sc_adj`	Positive numeric scalar; manual multiplicative adjustment to the scale of the output pseudo-target.
`lb`	Numeric scalar giving the value of left truncation of the resulting pseudo-target. Defaults to `-Inf`.

ub	Numeric scalar giving the value of right truncation of the resulting pseudo-target. Defaults to Inf.
maxit	See optim.
...	See optim.

Value

A list with the same outputs as pseudo_list; also includes opt, which gives output of optim.

Examples

```
pseu <- lapprox(function(x) dnorm(x, log = TRUE),
  family = "t",
  params = list(loc = NA, sc = NA, df = 5.0),
  init = 0.5, lb = -1.0)
curve(dnorm(x)/(1 - pnorm(-1)), from = -1, to = 6, col = "blue")
xx <- seq(-1, 6, length = 500)
lines(xx, sapply(xx, FUN = pseu$d), lty = 2, col = "red")
```

---

pseudo_condseq *Sequence of conditional pseudo-targets from a realization*

Description

Given a realization of a random vector, generate the corresponding sequence of conditional pseudo-targets (Heiner et al., 2026+). The pseudo-target is specified as a sequence of growing conditional distributions.

Usage

```
pseudo_condseq(x, pseudo_init, loc_fn, sc_fn, lb, ub)
```

Arguments

x	A numeric vector (length at least two) giving the realization of the random vector on its original scale.
pseudo_init	A list output from pseudo_list describing the marginal pseudo-target for x[1]. All subsequent pseudo-targets will resemble pseudo_init with exception of different location and scale parameters.
loc_fn	A function that specifies the location of a conditional pseudo-target given the elements in x that precede it.
sc_fn	A function that specifies the scale of a conditional pseudo-target given the elements in x that precede it
lb	A numeric vector (same length as x) specifying the lower bound of support for each conditional pseudo-target.
ub	A numeric vector (same length as x) specifying the upper bound of support for each conditional pseudo-target.

Details

See the documentation for slice_quantile_mv_seq for examples.

Value

A list containing a sequence of pseudo-targets, each from pseudo_list.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
# Funnel distribution from Neal (2003).
K <- 10
n_iter <- 50 # MCMC iterations; set to 10e3 for more complete illustration
n <- 100 # number of iid samples from the target; set to 10e3 for more complete illustration
Y <- matrix(NA, nrow = n, ncol = K) # iid samples from the target
Y[, 1] <- rnorm(n, 0.0, 3.0)
for (i in 1:n) {
  Y[i, 2:K] <- rnorm(K - 1, 0.0, exp(0.5 * Y[i, 1]))
}
ltarget <- function(x) {
  dnorm(x[1], 0.0, 3.0, log = TRUE) +
    sum(dnorm(x[2:K], 0.0, exp(0.5 * x[1]), log = TRUE))
}
pseudo_control <- list(
  loc_fn = function(x) {
    0.0
  },
  sc_fn = function(x) {
    if (is.null(x)) {
      out <- 3.0
    } else {
      out <- exp(0.5 * x[1])
    }
    out
  },
  pseudo_init = pseudo_list(
    family = "t",
    params = list(loc = 0.0, sc = 3.0, df = 20),
    lb = -Inf,
    ub = Inf
  ),
  lb = rep(-Inf, K),
  ub = rep(Inf, K)
)
x0 <- runif(K)
draws <- matrix(rep(x0, n_iter + 1), nrow = n_iter + 1, byrow = TRUE)
draws_u <- matrix(rep(x0, n_iter), nrow = n_iter, byrow = TRUE)
n_eval <- 0
```

```
for (i in 2:(n_iter + 1)) {
  tmp <- slice_quantile_mv_seq(
    draws[i - 1, ],
    log_target = ltarget,
    pseudo_control = pseudo_control
  )
  draws[i, ] <- tmp$x
  draws_u[i - 1, ] <- tmp$u
  n_eval <- n_eval + tmp$nEvaluations
}
# (es <- coda::effectiveSize(coda::as.mcmc(draws)))
# mean(es)
n_eval / n_iter
sapply(1:K, function(k) {
  utility_shrinkslice(u = draws_u[, k], utility_type = "AUC")
})
hist(draws_u[, 1])
plot(draws[, 1], draws[, 2])
points(Y[, 1], Y[, 2], col = "blue", cex = 0.5)
```

`pseudo_condseq_XfromU` *Inverse transform from sequence of conditional pseudo-targets*

Description

Given a vector from the unit hypercube, map to the original (back-transformed) vector using a sequence of conditional pseudo-target inverse CDFs. The pseudo-target is specified as a sequence of growing conditional distributions.

Usage

```
pseudo_condseq_XfromU(
  u,
  pseudo_init,
  loc_fn,
  sc_fn,
  lb,
  ub,
  diagnostics = NULL
)
```

Arguments

`u`	A numeric vector (length at least two) of values strictly between 0 and 1.
`pseudo_init`	A list output from pseudo_list describing the marginal pseudo-target for `x[1]`.
`loc_fn`	A function that specifies the location of a conditional pseudo-target given the elements in `x` that precede it.

sc_fn	A function that specifies the scale of a conditional pseudo-target given the elements in x that precede it
lb	A numeric vector (same length as u) specifying the lower bound of support for each conditional pseudo-target.
ub	A numeric vector (same length as u) specifying the upper bound of support for each conditional pseudo-target.
diagnostics	Logical, indicating whether the function should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Details

See the documentation for slice_quantile_mv_seq for examples.

Value

A list containing x obtained from the sequence of inverse CDFs, and pseudo_seq, a list of the corresponding sequential pseudo-targets output from pseudo_list.

Examples

```
# Funnel distribution from Neal (2003).
K <- 10
n_iter <- 50 # MCMC iterations; set to 10e3 for more complete illustration
n <- 100 # number of iid samples from the target; set to 10e3 for more complete illustration
Y <- matrix(NA, nrow = n, ncol = K) # iid samples from the target
Y[, 1] <- rnorm(n, 0.0, 3.0)
for (i in 1:n) {
  Y[i, 2:K] <- rnorm(K - 1, 0.0, exp(0.5 * Y[i, 1]))
}
ltarget <- function(x) {
  dnorm(x[1], 0.0, 3.0, log = TRUE) +
    sum(dnorm(x[2:K], 0.0, exp(0.5 * x[1]), log = TRUE))
}
pseudo_control <- list(
  loc_fn = function(x) {
    0.0
  },
  sc_fn = function(x) {
    if (is.null(x)) {
      out <- 3.0
    } else {
      out <- exp(0.5 * x[1])
    }
    out
  },
  pseudo_init = pseudo_list(
    family = "t",
    params = list(loc = 0.0, sc = 3.0, df = 20),
    lb = -Inf,
```

```
    ub = Inf
  ),
  lb = rep(-Inf, K),
  ub = rep(Inf, K)
)
x0 <- runif(K)
draws <- matrix(rep(x0, n_iter + 1), nrow = n_iter + 1, byrow = TRUE)
draws_u <- matrix(rep(x0, n_iter), nrow = n_iter, byrow = TRUE)
n_eval <- 0
for (i in 2:(n_iter + 1)) {
  tmp <- slice_quantile_mv_seq(
    draws[i - 1, ],
    log_target = ltarget,
    pseudo_control = pseudo_control
  )
  draws[i, ] <- tmp$x
  draws_u[i - 1, ] <- tmp$u
  n_eval <- n_eval + tmp$nEvaluations
}
# (es <- coda::effectiveSize(coda::as.mcmc(draws)))
# mean(es)
n_eval / n_iter
sapply(1:K, function(k) {
  utility_shrinkslice(u = draws_u[, k], utility_type = "AUC")
})
hist(draws_u[, 1])
plot(draws[, 1], draws[, 2])
points(Y[, 1], Y[, 2], col = "blue", cex = 0.5)
```

`pseudo_list` *Specify a pseudo-target within a given class*

Description

Create a list of functions to evaluate a pseudo-target in a given class with supplied parameters (usually location and scale). The distribution is optionally truncated to specified bounds (and renormalized). See Heiner et al. (2026+).

Usage

```
pseudo_list(family, params, lb = -Inf, ub = Inf, name = NULL)
```

Arguments

`family`	String identifying the distribution family. One of `t`, `cauchy`, `normal`, `logistic`, and `beta`.

`params`	Named list identifying parameters, which vary by distribution family.
	`t`: location `loc`, scale `sc`, and degrees of freedom `df`
	`cauchy`: location `loc` and scale `sc`

	`normal`: location `loc` and scale `sc`
	`logistic`: location `loc` and scale `sc`
	`beta`: shape `shape1` and shape `shape2`

`lb`	Numeric scalar giving the value of left truncation. Defaults to `-Inf`. For family `beta`, this is intersected with its natural lower bound of zero.
`ub`	Numeric scalar giving the value of right truncation. Defaults to `Inf`. For family `beta`, this is intersected with its natural upper bound of one.
`name`	String appending optional message to the textual name of the distribution.

Details

The supported classes of pseudo-targets include: `t`, `cauchy`, `normal`, `logistic`, and `beta`.

Value

A list with named components:

`d`: function to evaluate the density (finite boundary points of the support evaluate to `0`)

`ld`: function to evaluate the log density (finite boundary points of the support evaluate to `-Inf`)

`q`: function to evaluate the quantile function

`p`: function to evaluate the distribution function

`txt`: text description of the distribution

`params`: repeats the `params` argument

`lb`: lower boundary of support

`ub`: upper boundary of support

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
pseu <- pseudo_list(family = "t", params = list(loc = 0.0, sc = 1.0, df = 5),
                    lb = 0.0, ub = Inf) # half t
str(pseu)
pseu$d(1.5)
pseu$ld(1.5)
pseu$p(1.5)
# should match (F(x) - F(0)) / (1 - F(0)) below
(pt(1.5, df = 5) - pt(0.0, df = 5)) / pt(0.0, df = 5, lower.tail = FALSE)
pseu$q(0.8060963)
curve(pseu$d(x), from = -1.0, to = 5.0, n = 1000)
pseu <- pseudo_list(family = "cauchy", params = list(loc = 0.0, sc = 1.0),
                    lb = 0.0, ub = Inf) # half Cauchy
str(pseu)
pseu$d(1.5)
pseu$ld(1.5)
```

```
pseu$p(1.5)
pseu$q(0.6256659)
pseu <- pseudo_list(family = "normal", params = list(loc = 0.0, sc = 1.0),
                    lb = 0.0, ub = Inf) # half normal
str(pseu)
pseu$d(1.5)
pseu$ld(1.5)
pseu$p(1.5)
pseu$q(0.8663856)
pseu <- pseudo_list(family = "logistic", params = list(loc = 0.0, sc = 1.0),
                    lb = 0.0, ub = Inf) # half logistic
str(pseu)
pseu$d(1.5)
pseu$ld(1.5)
pseu$p(1.5)
pseu$q(0.635149)
pseu <- pseudo_list(family = "beta", params = list(shape1 = 3.0, shape2 = 2.0), lb = 0.2)
str(pseu)
pseu$d(0.5)
pseu$ld(0.5)
pseu$p(0.5)
# should match (F(x) - F(0.2)) / (1 - F(0.2)) below
(pbeta(0.5, 3.0, 2.0) - pbeta(0.2, 3.0, 2.0)) / pbeta(0.2, 3.0, 2.0, lower.tail = FALSE)
pseu$q(0.2932771)
curve(pseu$d(x), from = 0.0, to = 1.0, n = 1000)
```

---

pseudo_opt *Optimal pseudo-target for a given target*

Description

Find an optimal pseudo-target in a specified family to approximate the given (unnormalized) target (Heiner et al., 2026+). Optimize over the selected utility function.

Usage

```
pseudo_opt(
  log_target = NULL,
  samples = NULL,
  type = "samples",
  family = "t",
  df = NULL,
  lb = -Inf,
  ub = Inf,
  utility_type = "AUC",
  nbins = 100,
  tol_opt = 1e-06,
  init_fn = NULL,
 utility_options = list(tol_int = 0.001, n_opt = 10, interval_opt = c(1e-06, 0.999999),
```

```
    diagnostics = NULL),
  plot = TRUE,
  verbose = FALSE,
  degf = NULL
)
```

Arguments

log_target	Function to evaluate the log density of the unnormalized univariate target.
samples	Optional numeric vector providing samples from the target distribution (for use as alternative to log_target).
type	String specifying the input type. One of "function", "samples", or "grid". Default is to use "samples". Use of "function" requires specification of log_target. Use of "samples" requires specification of samples. Use of "grid" requires specification of log_target.
family	String specifying the family of distributions for the pseudo-target. Can be any of the families accepted by pseudo_list.
df	Numeric vector of degrees of freedom values to try (only if family = "t"). Defaults to c(1, 5, 20).
lb	Numeric scalar giving the value of left truncation. Defaults to -Inf.
ub	Numeric scalar giving the value of right truncation. Defaults to Inf.
utility_type	String identifying utility type, either AUC (default) or MSW.
nbins	Positive integer specifying the number of histogram bins if using "samples" or "grid". Defaults to 100.
tol_opt	Positive numeric scalar that passes to reltol in the call to optim. Defaults to 1.0e-6.
init_fn	Function with no arguments that returns a vector of parameter values for the initial pseudo-target, usually consisting of a real-valued location parameter and a positive-valued scale parameter. If init_fn is NULL and samples are supplied, the sample mean and standard deviation are used; if init_fn is NULL and samples are not supplied, the vector c(0.5, 3.0) is used. Default is NULL.
utility_options	List containing the following. tol_int: Positive numeric scalar that passes to abs.tol in the call to integrate used for type = "function". n_opt: Positive integer giving the number of equally spaced initialization points for numerical optimization (optim) in AUC with type = "function". interval_opt: Numerical vector of length two giving the interval for numerical optimization (optim) in AUC with type = "function". diagnostics: Logical, indicating whether the target log-density and pseudo-target functions should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.
plot	Logical for whether to generate two plots:

1. direct comparison of the target and pseudo-target densities, and
2. transformed target density.

Defaults to TRUE.

verbose — Logical for whether to print intermediate steps of optimization. Defaults to FALSE.

degf — (deprecated) Numeric vector of degrees of freedom values to try (only if family = "t"). Use df instead.

Details

Optionally supply samples from the target distribution.

Value

A list with named components:

pseudo: a list with functions corresponding to the selected pseudo-target; output of pseudo_list.

utility: value of the utility function using the selected pseudo-target; output of utility_pseudo.

utility_type: repeats the input specifying the utility type.

opt: output of optim.

Other outputs repeating inputs.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
oldpar <- par(mfrow = c(1,2))
(pseu <- pseudo_opt(samples = rnorm(1e3), type = "samples",
               family = "t", utility_type = "AUC",
               nbins = 10, plot = TRUE,
               verbose = FALSE))
(pseu <- pseudo_opt(log_target = function(x) dnorm(x, log = TRUE),
                type = "grid",
                family = "logistic", utility_type = "MSW",
                nbins = 30, plot = TRUE,
                verbose = FALSE))
(pseu <- pseudo_opt(log_target = function(x) dbeta(x, 4, 2, log = TRUE),
                lb = 0, ub = 1,
                type = "function",
                family = "cauchy", utility_type = "AUC",
                init_fn = function() {c(rbeta(1, 3.0, 3.0), rgamma(1, 3.0))},
                nbins = 30, plot = TRUE,
                verbose = FALSE))
par(oldpar)
```

rwm_norm *Univariate Random Walk Metropolis with Normal Proposal*

Description

Single update for the univariate random-walk Metropolis sampler with a normal proposal distribution.

Usage

```
rwm_norm(x, log_target, cand_sd, support = c(-Inf, Inf), diagnostics = NULL)
```

Arguments

x	The current state (as a numeric scalar).
log_target	A function taking numeric scalar that evaluates the (potentially unnormalized) log-target density, returning a numeric scalar.
cand_sd	A positive numeric scalar tuning the algorithm. Gives the standard deviation of the normal random-walk proposal.
support	Numeric vector of length two giving the lower and upper boundaries of the support for the target distribution. Default is `c(-Inf, Inf)`, in which case the target density will always be evaluated at the proposed value.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When `diagnostics` is `NULL`, it is reset to match the value of `getOption("qslice.diagnostics")` which defaults to `FALSE` if the option is not set. Default for `diagnostics` is `NULL`.

Value

A list with three elements:

`x` is the new state.

`accept` Logical indicating whether the proposal was accepted.

`nEvaluations` is the number of evaluations of the target function used to obtain the new state.

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
draws <- numeric(10) + 0.5 # set to numeric(1e3) for more complete illustration
nEvaluations <- 0L
nAccept <- 0L
for (i in seq.int(2, length(draws))) {
  out <- rwm_norm(draws[i - 1], log_target = lf, cand_sd = 0.3, support = c(0.0, 1.0))
  draws[i] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
  nAccept <- nAccept + out$accept
}
```

```
nEvaluations / (length(draws) - 1)
nAccept / (length(draws) - 1)
plot(density(draws), xlim = c(0, 1))
curve(exp(lf(x)), 0, 1, col = "blue", add = TRUE)
```

`slice_elliptical` *Univariate Elliptical Slice Sampler*

Description

Algorithm 1 of Nishihara et al. (2014) of the elliptical slice sampler of Murray et al. (2010).

Usage

```
slice_elliptical(x, log_lik, mu, sigma, diagnostics = NULL)
```

Arguments

`x`	The current state (as a numeric scalar).
`log_lik`	A function taking numeric scalar that evaluates the natural logarithm of the "likelihood" L(x) part of the unnormalized target density L(x) * Normal(x; mu, sigma). Returns a numeric scalar.
`mu`	A numeric scalar with the mean of the supporting normal "prior" distribution.
`sigma`	A numeric scalar with the standard deviation of the supporting normal "prior" distribution.
`diagnostics`	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When `diagnostics` is `NULL`, it is reset to match the value of `getOption("qslice.diagnostics")` which defaults to `FALSE` if the option is not set. Default for `diagnostics` is `NULL`.

Value

A list with two elements:

`x` is the new state.

`nEvaluations` is the number of evaluations of the log-likelihood function used to obtain the new state.

References

Murray, I., Adams, R., and MacKay, D., (2010), "Elliptical Slice Sampling," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings. https://proceedings.mlr.press/v9/murray10a

Nishihara, R., Murray, I., and Adams, R. P. (2014), "Parallel MCMC with Generalized Elliptical Slice Sampling," *Journal of Machine Learning Research*, 15, 2087-2112. https://jmlr.org/papers/v15/nishihara14a.html

Examples

```
log_lik <- function(x) dbeta(x, 8, 2, log = TRUE)
lf <- function(x) log_lik(x) + dnorm(x, mean = mu, sd = sig, log = TRUE)
mu <- 0.2
sig <- 0.3
draws <- numeric(10) + 0.5 # set to numeric(1e3) + 0.5 for more complete illustration
nEvaluations <- 0L
for (i in seq.int(2, length(draws))) {
  out <- slice_elliptical(draws[i - 1], log_lik = log_lik, mu = mu, sigma = sig)
  draws[i] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (length(draws) - 1)
plot(density(draws), xlim = c(-0.2, 1.2))
curve(exp(lf(x)) * 4.5, -0.2, 1.2, col = "blue", add = TRUE) # 4.5 to match curve height
```

---

slice_elliptical_mv *Multivariate Elliptical Slice Sampler*

Description

Algorithm 1 of Nishihara et al. (2014) of the elliptical slice sampler of Murray et al. (2010).

Usage

```
slice_elliptical_mv(x, log_lik, mu, Sig, is_chol = FALSE, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric vector).
log_lik	A function taking numeric vector that evaluates the natural logarithm of the "likelihood" L(x) part of the unnormalized target density L(x) * Normal(x; mu, Sig). Returns a numeric scalar.
mu	Numeric vector with the mean of the supporting normal "prior" distribution.
Sig	Positive definite covariance matrix of the supporting normal "prior" distribution. Alternatively, a lower-triangular matrix with the Cholesky factor of the covariance matrix (for faster computation).
is_chol	Logical, is the supplied Sig in Cholesky (lower triangular) format? Default is false.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing two elements: `x` is the new state and `nEvaluations` is the number of evaluations of the log-likelihood function used to obtain the new state.

References

Murray, I., Adams, R., and MacKay, D., (2010), "Elliptical Slice Sampling," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings. https://proceedings.mlr.press/v9/murray10a

Nishihara, R., Murray, I., and Adams, R. P. (2014), "Parallel MCMC with Generalized Elliptical Slice Sampling," *Journal of Machine Learning Research*, 15, 2087-2112. https://jmlr.org/papers/v15/nishihara14a.html

Examples

```
mu <- c(0.5, 0.5)
Sig <- matrix(c(0.5, 0.25, 0.25, 0.5), nrow = 2)
SigL <- t(chol(Sig))
log_lik <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
lf <- function(x) log_lik(x) - 0.5 * drop(crossprod(forwardsolve(SigL, x - mu)))
n_iter <- 10 # set to 1e3 for more complete illustration
draws <- matrix(0.3, nrow = n_iter, ncol = 2)
nEvaluations <- 0L
for (i in seq.int(2, n_iter)) {
  out <- slice_elliptical_mv(draws[i - 1,], log_lik = log_lik,
              mu = mu, Sig = SigL, is_chol = TRUE)
  draws[i,] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (n_iter - 1)
plot(draws[,1], draws[,2], xlim = c(0, 1))
hist(draws[,1], freq = FALSE); curve(dbeta(x, 3, 4), col = "blue", add = TRUE)
hist(draws[,2], freq = FALSE); curve(dbeta(x, 5, 3), col = "blue", add = TRUE)
```

---

slice_genelliptical *Generalized Elliptical Slice Sampler (univariate)*

Description

Single update using the generalized elliptical slice sampler of Nishihara et al. (2014).

Usage

```
slice_genelliptical(x, log_target, mu, sigma, df, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric scalar).
log_target	A function taking numeric scalar that evaluates the (potentially unnormalized) log-target density, returning a numeric scalar.
mu	A numeric scalar with the location parameter of the Student t pseudo-target.
sigma	A positive numeric scalar with the scale parameter of the Student t pseudo-target.
df	Degrees of freedom of Student t pseudo-target.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing two elements:

x is the new state.

nEvaluations is the number of evaluations of the target function used to obtain the new state.

References

Nishihara, R., Murray, I., and Adams, R. P. (2014), "Parallel MCMC with Generalized Elliptical Slice Sampling," *Journal of Machine Learning Research*, 15, 2087-2112. https://jmlr.org/papers/v15/nishihara14a.html

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
draws <- numeric(10) + 0.5 # set to numeric(1e3) + 0.5 for more complete illustration
nEvaluations <- 0L
for (i in seq.int(2, length(draws))) {
  out <- slice_genelliptical(draws[i - 1], log_target = lf,
                                     mu = 0.5, sigma = 1, df = 5)
  draws[i] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (length(draws) - 1)
plot(density(draws), xlim = c(0, 1))
curve(exp(lf(x)), 0, 1, col = "blue", add = TRUE)
```

slice_genelliptical_mv *Generalized Elliptical Slice Sampler (Multivariate)*

Description

Generalized Elliptical Slice Sampler, Algorithm 2 of Nishihara et al. (2014)

Usage

```
slice_genelliptical_mv(
  x,
  log_target,
  mu,
  Sig,
  df,
  is_chol = FALSE,
  diagnostics = NULL
)
```

Arguments

x	The current state (as a numeric vector).
log_target	A function taking numeric vector that evaluates the natural logarithm of the unnormalized target density. Returns a numeric scalar.
mu	A numeric vector with the location parameter of the Student t pseudo-target.
Sig	Positive definite scale matrix of the Student t pseudo-target. Alternatively, a lower-triangular matrix with the Cholesky factor of the scale matrix (for faster computation).
df	Degrees of freedom of Student t pseudo-target.
is_chol	Logical, is the supplied Sig in Cholesky (lower triangular) format? Default is false.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing two elements: x is the new state and nEvaluations is the number of evaluations of the target function used to obtain the new state.

References

Nishihara, R., Murray, I., and Adams, R. P. (2014), "Parallel MCMC with Generalized Elliptical Slice Sampling," *Journal of Machine Learning Research*, 15, 2087-2112. https://jmlr.org/papers/v15/nishihara14a.html

Examples

```
lf <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
n_iter <- 10 # set to 1e4 for more complete illustration
draws <- matrix(0.3, nrow = n_iter, ncol = 2)
nEvaluations <- 0L
for (i in seq.int(2, n_iter)) {
  out <- slice_genelliptical_mv(draws[i - 1,], log_target = lf,
              mu = c(0.5, 0.5), Sig = matrix(c(0.5, 0.25, 0.25, 0.5), nrow = 2),
              df = 5)
  draws[i,] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (n_iter - 1)
plot(draws[,1], draws[,2], xlim = c(0, 1))
hist(draws[,1], freq = FALSE); curve(dbeta(x, 3, 4), col = "blue", add = TRUE)
hist(draws[,2], freq = FALSE); curve(dbeta(x, 5, 3), col = "blue", add = TRUE)
```

---

slice_hyperrect *Multivariate Slice Sampler with Shrinking Hyperrectangle*

Description

Multivariate slice sampler in Algorithm 8 of Neal (2003) using the "shrinkage" procedure.

Usage

```
slice_hyperrect(
  x,
  log_target,
  w = NULL,
  L = NULL,
  R = NULL,
  diagnostics = NULL
)
```

Arguments

x	The current state (as a numeric vector).
log_target	A function taking numeric vector that evaluates the log-target density, returning a numeric scalar.
w	A numeric vector tuning the algorithm which gives the typical slice width in each dimension. This is a main tuning parameter of the algorithm. If NULL, the sampler begins shrinking from the supplied boundaries L and R (which should correspond with the support).
L	Optional numeric vector giving the lower boundary of support in each dimension.

R — Optional numeric vector giving the upper boundary of support in each dimension. Will be used if w is null. If all of L, R, and w are null, then the boundaries default to those of the unit hypercube.

diagnostics — Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing two elements: "x" is the new state and "nEvaluations" is the number of evaluations of the target function used to obtain the new state.

References

Neal, R. M. (2003), "Slice sampling," *The Annals of Statistics*, 31, 705-767. doi:10.1214/aos/1056562461

Examples

```
lf <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
n_iter <- 10 # set to 1e4 for more complete illustration
draws <- matrix(0.2, nrow = n_iter, ncol = 2)
nEvaluations <- 0L
for (i in seq.int(2, n_iter)) {
 out <- slice_hyperrect(draws[i - 1, ], log_target = lf, w = c(0.5, 0.5))
 draws[i,] <- out$x
 nEvaluations <- nEvaluations + out$nEvaluations
 cat(i, '\r')
}
nEvaluations / (nrow(draws) - 1)
plot(draws[,1], draws[,2], xlim = c(0, 1))
hist(draws[,1], freq = FALSE); curve(dbeta(x, 3, 4), col = "blue", add = TRUE)
hist(draws[,2], freq = FALSE); curve(dbeta(x, 5, 3), col = "blue", add = TRUE)
```

---

slice_latent *Latent Slice Sampler*

Description

Single update using the latent slice sampler of Li and Walker (2023).

Usage

```
slice_latent(x, s, log_target, rate, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric scalar).
s	A random variable that determines the length of the initial shrinking interval.
log_target	A function taking numeric scalar that evaluates the (potentially unnormalized) log-target density, returning a numeric scalar.
rate	The rate parameter for the distribution of s.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing three elements:

x is the new state of the target variable.

s is the new state of the latent scale variable.

nEvaluations is the number of evaluations of the target function used to obtain the new state.

References

Li, Y. and Walker, S. G. (2023), "A latent slice sampling algorithm," *Computational Statistics and Data Analysis*, 179, 107652. doi:10.1016/j.csda.2022.107652

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
draws <- numeric(10) + 0.5 # set to numeric(1e3) + 0.5 for more complete illustration
nEvaluations <- 0L
s <- 0.5
for (i in seq.int(2, length(draws))) {
  out <- slice_latent(draws[i - 1], s, log_target = lf, rate = 0.3)
  draws[i] <- out$x
  s <- out$s
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (length(draws) - 1)
plot(density(draws), xlim = c(0, 1))
curve(exp(lf(x)), 0, 1, col = "blue", add = TRUE)
```

`slice_quantile` *Quantile Slice Sampler*

Description

Single update using a quantile slice sampler of Heiner et al. (2026+).

Usage

```
slice_quantile(x, log_target, pseudo, diagnostics = NULL)
```

Arguments

`x`	The current state (as a numeric scalar).
`log_target`	A function taking numeric scalar that evaluates the (potentially unnormalized) log-target density, returning a numeric scalar.
`pseudo`	List containing two functions specifying the pseudo-target distribution: `ld` evaluates the log density for a scalar input, and `q` evaluates the quantile (inverse-CDF) function for an input in (0,1).
`diagnostics`	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When `diagnostics` is `NULL`, it is reset to match the value of `getOption("qslice.diagnostics")` which defaults to `FALSE` if the option is not set. Default for `diagnostics` is `NULL`.

Details

The sampler works with ψ, the pseudo-target CDF of the state, in double precision. Because floating-point numbers are dense near zero but not near one, states whose pseudo-target upper-tail probability is below about `1e-16` cannot be represented. Use a pseudo-target whose right tail is at least as heavy as that of the target (e.g., Student t).

Value

A list containing three elements:

`x` is the new state.

`u` is the value of the CDF of the pseudo-target associated with the returned value (also referred to as psi).

`nEvaluations` is the number of evaluations of the target function used to obtain the new state.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
pseu <- pseudo_list(family = "beta", params = list(shape1 = 2.0, shape2 = 2.0))
draws <- numeric(10) + 0.5 # set to numeric(1e3) + 0.5 for more complete illustration
nEvaluations <- 0L
for (i in seq.int(2, length(draws))) {
  out <- slice_quantile(draws[i - 1], log_target = lf, pseudo = pseu)
  draws[i] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (length(draws) - 1)
plot(density(draws), xlim = c(0, 1))
curve(exp(lf(x)), 0, 1, col = "blue", add = TRUE)
```

---

slice_quantile_mv *Multivariate Quantile Slice Sampler*

Description

Quantile slice sampler for a random vector (Heiner et al., 2026+). The pseudo-target is specified through independent univariate distributions.

Usage

```
slice_quantile_mv(x, log_target, pseudo, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric vector).
log_target	A function taking numeric vector that evaluates the log-target density, returning a numeric scalar.
pseudo	List of length equal to the number of dimensions in x. Each element is itself a list that specifies the pseudo-target for the corresponding dimension with functions ld that evaluates the log density, p that evaluates the CDF, and q that evaluates the quantile (inverse-CDF) function.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing three elements: "x" is the new state, "u" is the vector of values of the pseudo-target CDFs associated with the returned value, and "nEvaluations" is the number of evaluations of the target function used to obtain the new state.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
lf <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
ps_shsc <- list(c(2, 2), c(2, 1))
ps <- list(
  list(ld = function(x) dbeta(x, ps_shsc[[1]][1], ps_shsc[[1]][2], log = TRUE),
       p = function(x) pbeta(x, ps_shsc[[1]][1], ps_shsc[[1]][2]),
       q = function(x) qbeta(x, ps_shsc[[1]][1], ps_shsc[[1]][2]) ),
  list(ld = function(x) dbeta(x, ps_shsc[[2]][1], ps_shsc[[2]][2], log = TRUE),
       p = function(x) pbeta(x, ps_shsc[[2]][1], ps_shsc[[2]][2]),
       q = function(x) qbeta(x, ps_shsc[[2]][1], ps_shsc[[2]][2]) )
  )
n_iter <- 10 # set to 1e4 for more complete illustration
draws <- matrix(0.2, nrow = n_iter, ncol = 2)
draws_u <- draws
draws_u[1,] <- sapply(1:length(ps), function(k) ps[[k]]$p(draws[1,k]))
nEvaluations <- 0L
for (i in seq.int(2, n_iter)) {
  out <- slice_quantile_mv(draws[i - 1, ], log_target = lf, pseudo = ps)
  draws[i,] <- out$x
  draws_u[i,] <- out$u
  nEvaluations <- nEvaluations + out$nEvaluations
  cat(i, '\r')
}
nEvaluations / (nrow(draws) - 1)
plot(draws[,1], draws[,2], xlim = c(0, 1))
hist(draws[,1], freq = FALSE); curve(dbeta(x, 3, 4), col = "blue", add = TRUE)
hist(draws[,2], freq = FALSE); curve(dbeta(x, 5, 3), col = "blue", add = TRUE)
plot(draws_u[,1], draws_u[,2], xlim = c(0, 1))
utility_shrinkslice(u = draws_u[,1], nbins = 30, type = "samples",
plot = TRUE, utility_type = "AUC")
utility_shrinkslice(u = draws_u[,2], nbins = 30, type = "samples",
plot = TRUE, utility_type = "AUC")
```

---

slice_quantile_mv_seq *Multivariate Quantile Slice Sampler from a sequence of conditional pseudo-targets*

Description

Quantile slice sampler for a random vector (Heiner et al., 2026+). The pseudo-target is specified as a sequence of growing conditional distributions.

Usage

```
slice_quantile_mv_seq(x, log_target, pseudo_control, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric vector).
log_target	A function taking numeric vector that evaluates the log-target density, returning a numeric scalar.
pseudo_control	A list with pseudo_init, a list output from pseudo_list describing the marginal pseudo-target for x[1]. Attributes of pseudo_init will be used in subsequent pseudo-targets, except for location and scale parameters. loc_fn, a function that specifies the location of a conditional pseudo-target given the elements in x that precede it. sc_fn, a function that specifies the scale of a conditional pseudo-target given the elements in x that precede it. lb, a numeric vector (same length as x) specifying the lower bound of support for each conditional pseudo-target. ub, a numeric vector (same length as x) specifying the upper bound of support for each conditional pseudo-target.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list containing three elements: "x" is the new state, "u" is a vector of values of the sequence of conditional CDFs of the pseudo-targets associated with the returned value, and "nEvaluations" is the number of evaluations of the target function used to obtain the new state.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
# Funnel distribution from Neal (2003).
K <- 10
n_iter <- 50 # MCMC iterations; set to 10e3 for more complete illustration
n <- 100 # number of iid samples from the target; set to 10e3 for more complete illustration
Y <- matrix(NA, nrow = n, ncol = K) # iid samples from the target
Y[, 1] <- rnorm(n, 0.0, 3.0)
for (i in 1:n) {
  Y[i, 2:K] <- rnorm(K - 1, 0.0, exp(0.5 * Y[i, 1]))
}
ltarget <- function(x) {
  dnorm(x[1], 0.0, 3.0, log = TRUE) +
    sum(dnorm(x[2:K], 0.0, exp(0.5 * x[1]), log = TRUE))
}
pseudo_control <- list(
```

```
  loc_fn = function(x) {
    0.0
  },
  sc_fn = function(x) {
    if (is.null(x)) {
      out <- 3.0
    } else {
      out <- exp(0.5 * x[1])
    }
    out
  },
  pseudo_init = pseudo_list(
    family = "t",
    params = list(loc = 0.0, sc = 3.0, df = 20),
    lb = -Inf,
    ub = Inf
  ),
  lb = rep(-Inf, K),
  ub = rep(Inf, K)
)
x0 <- runif(K)
draws <- matrix(rep(x0, n_iter + 1), nrow = n_iter + 1, byrow = TRUE)
draws_u <- matrix(rep(x0, n_iter), nrow = n_iter, byrow = TRUE)
n_eval <- 0
for (i in 2:(n_iter + 1)) {
  tmp <- slice_quantile_mv_seq(
    draws[i - 1, ],
    log_target = ltarget,
    pseudo_control = pseudo_control
  )
  draws[i, ] <- tmp$x
  draws_u[i - 1, ] <- tmp$u
  n_eval <- n_eval + tmp$nEvaluations
}
# (es <- coda::effectiveSize(coda::as.mcmc(draws)))
# mean(es)
n_eval / n_iter
sapply(1:K, function(k) {
  utility_shrinkslice(u = draws_u[, k], utility_type = "AUC")
})
hist(draws_u[, 1])
plot(draws[, 1], draws[, 2])
points(Y[, 1], Y[, 2], col = "blue", cex = 0.5)
```

`slice_stepping_out` *Slice sampler using the Stepping Out and Shrinkage Procedures*

Description

Single update for the univariate slice sampler of Neal (2003) using the "stepping out" procedure, followed by the "shrinkage" procedure.

Usage

```
slice_stepping_out(x, log_target, w, max = Inf, diagnostics = NULL)
```

Arguments

x	The current state (as a numeric scalar).
log_target	A function taking numeric scalar that evaluates the (potentially unnormalized) log-target density, returning a numeric scalar.
w	A numeric scalar tuning the algorithm which gives the typical slice width. This is a main tuning parameter of the algorithm.
max	The limit m of Neal (2003) on the size of the initial interval, in units of w; at most max - 1 steps out are taken in total (max of zero or one gives no stepping out). Setting max to zero avoids some evaluations of log_target, but may lead to relatively high autocorrelation if w is too small. If w is too small, setting max to a large value (even Inf) should lead to low autocorrelation at the cost of more evaluations for log_target.
diagnostics	Logical, indicating whether the sampler should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Value

A list with two elements:

x is the new state.

nEvaluations is the number of evaluations of the target function used to obtain the new state.

References

Neal, R. M. (2003), "Slice sampling," *The Annals of Statistics*, 31, 705-767. doi:10.1214/aos/1056562461

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
draws <- numeric(10) + 0.5 # set to numeric(1e3) for more complete illustration
nEvaluations <- 0L
for (i in seq.int(2, length(draws))) {
  out <- slice_stepping_out(draws[i - 1], log_target = lf, w = 0.7, max = Inf)
  draws[i] <- out$x
  nEvaluations <- nEvaluations + out$nEvaluations
}
nEvaluations / (length(draws) - 1)
plot(density(draws), xlim = c(0, 1))
curve(exp(lf(x)), 0, 1, col = "blue", add = TRUE)
```

`utility_pseudo` *Utility for a given univariate target and pseudo-target*

Description

Takes a pseudo-target and target (or samples from the target) and evaluates the utility function for the transformed target, which can be one of Area Under the Curve (AUC) and Mean Slice Width (MSW). See Heiner et al. (2026+).

Usage

```
utility_pseudo(
  pseudo,
  log_target = NULL,
  samples = NULL,
  type = "samples",
  x = NULL,
  nbins = 30,
  plot = TRUE,
  utility_type = "AUC",
  options = list(tol_int = 0.001, n_opt = 10, interval_opt = c(1e-06, 0.999999),
    diagnostics = NULL)
)
```

Arguments

`pseudo`	List containing the following functions relating to the univariate pseudo-target, each with scalar input: `ld`: function to evaluate the log density `q`: function to evaluate the quantile function `p`: function to evaluate the distribution function
`log_target`	Function to evaluate the log density of the unnormalized univariate target.
`samples`	Numeric vector of samples from the target distribution.
`type`	String specifying the input type. One of "function", "samples", or "grid". Default is to use "samples". Use of "function" requires specification of `log_target`. Utility is calculated using numerical integration (integrate) and, additionally for AUC, optimization (optim). Use of "samples" requires specification of `samples`. Use of "grid" requires specification of `log_target` and optionally `x`. Grid method is useful if numerical optimization or integration fails when employed with `type = "function"`.
`x`	Optional numeric vector specifying grid (on (0,1)) over which to evaluate the transformed target. Defaults to `NULL`.

nbins	Number of histogram bins to use (defaults to 30). Overridden by the length of x if x is supplied.
plot	Logical for whether to generate two plots: 1. direct comparison of the target and pseudo-target densities, and 2. transformed target density. Defaults to TRUE.
utility_type	String identifying utility type, either AUC (default) or MSW.
options	List containing the following. tol_int: Positive numeric scalar that passes to abs.tol in the call to integrate used for type = "function". n_opt: Positive integer giving the number of equally spaced initialization points for numerical optimization (optim) in AUC with type = "function". interval_opt: Numerical vector of length two giving the interval for numerical optimization (optim) in AUC with type = "function". diagnostics: Logical, indicating whether the target log-density and pseudo-target functions should be run in diagnostic mode for informative error handling. When diagnostics is NULL, it is reset to match the value of getOption("qslice.diagnostics") which defaults to FALSE if the option is not set. Default for diagnostics is NULL.

Details

Optionally plot the target and pseudo-target densities as well as the transformed target.

Value

Scalar value of the utility function evaluation.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
pseu <- pseudo_list(family = "t", params = list(loc = 0.25, sc = 1.0, df = 1))
ltargdens <- function(x) dnorm(x, log = TRUE)
oldpar <- par(mfrow = c(1,2))
utility_pseudo(pseudo = pseu, log_target = ltargdens, type = "function", utility_type = "MSW")
utility_pseudo(pseudo = pseu, log_target = ltargdens, type = "function", utility_type = "AUC")
samp <- rnorm(10e3)
utility_pseudo(pseudo = pseu, samples = samp, nbins = 30, type = "samples", utility_type = "MSW")
utility_pseudo(pseudo = pseu, samples = samp, nbins = 30, type = "samples", utility_type = "AUC")
par(oldpar)
pseu <- pseudo_list(family = "beta", params = list(shape1 = 1.5, shape2 = 1.5))
ltargdens <- function(x) dbeta(x, shape1 = 2.0, shape2 = 3.0, log = TRUE)
oldpar <- par(mfrow = c(1,2))
utility_pseudo(pseudo = pseu, log_target = ltargdens, type = "function", utility_type = "MSW")
utility_pseudo(pseudo = pseu, log_target = ltargdens, type = "function", utility_type = "AUC")
samp <- rbeta(10e3, shape1 = 2.0, shape2 = 3.0)
utility_pseudo(pseudo = pseu, samples = samp, nbins = 30, type = "samples", utility_type = "MSW")
```

```
utility_pseudo(pseudo = pseu, samples = samp, nbins = 30, type = "samples", utility_type = "AUC")
par(oldpar)
```

`utility_shrinkslice` *Utility for a Transformed Target*

Description

Evaluates the utility function for a transformed target, which can be one of Area Under the Curve (AUC) and Mean Slice Width (MSW).

Usage

```
utility_shrinkslice(
  h = NULL,
  x = NULL,
  u = NULL,
  type = "samples",
  nbins = 30,
  plot = FALSE,
  utility_type = "AUC",
  options = list(tol_int = 0.001, n_opt = 10, interval_opt = c(1e-06, 0.999999))
)
```

Arguments

`h`	Function to evaluate the unnormalized transformed target $h(\psi) = g(\hat{\Pi}^{-1}(\psi))/\hat{\pi}(\hat{\Pi}^{-1}(\psi))$ with argument $\psi \in (0, 1)$. Use `h = NULL` if type = "samples".
`x`	Numeric vector of histogram locations. Not used if `u` is supplied or type = "integration". If used but not supplied, a default grid is chosen.
`u`	Numeric vector of samples supported on unit interval (ψ) with which to create histogram (use `u = NULL` if type = "integration" or "grid").
`type`	String specifying the utility calculation method. One of "samples", "integration", or "grid". Use of "integration" requires `h`. Use of "samples" requires `u`. Use of "grid" requires `h` and optionally `x`. Default is to use "samples".
`nbins`	Number of histogram bins to use (defaults to 30). Overridden by the length of `x`.
`plot`	Logical for whether to plot a visualization of the transformed target. Defaults to `FALSE`.
`utility_type`	String identifying utility type, either "AUC" (default) or "MSW"
`options`	List containing the following. `tol_int`: Positive numeric scalar that passes to `abs.tol` in the call to `integrate()` used for type = "integration". `n_opt`: Positive integer giving the number of initialization points for numerical optimization in AUC with type = "integration". `interval_opt`: Numerical vector of length two giving the interval for numerical optimization in AUC with type = "integration".

Details

Calculating the utility with a function and `type = 'integration'` employs numerical optimization (for AUC) and numerical integration (for both AUC and MSW), which may not succeed for highly multimodal or narrow target densities.

Value

Scalar value of the utility function evaluation.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2026+), "Quantile Slice Sampling," *arXiv preprint arXiv:2407.12608* doi:10.48550/arXiv.2407.12608

Examples

```
a <- 4
b <- 2
u <- rbeta(10e3, a, b)
h <- function(x) dbeta(x, a, b)
x <- sort(runif(20))
type <- "AUC"
type <- "MSW"

utility_shrinkslice(
  h = h,
  x = NULL,
  u = NULL,
  type = "integration",
  plot = TRUE,
  utility_type = type
)

utility_shrinkslice(
  h = h,
  x = NULL,
  u = NULL,
  nbins = 50,
  type = "grid",
  plot = TRUE,
  utility_type = type
)

utility_shrinkslice(
  h = h,
  x = x,
  u = NULL,
  type = "grid",
  plot = TRUE,
  utility_type = type
)
```

```
utility_shrinkslice(
  h = NULL,
  x = NULL,
  u = u,
  nbins = 30,
  type = "samples",
  plot = TRUE,
  utility_type = type
)

utility_shrinkslice(
  h = NULL,
  x = x,
  u = u,
  type = "samples",
  plot = TRUE,
  utility_type = type
)
```

Index

auc, 2

imh_pseudo, 3
integrate, *12*, *29, 30*

lapprox, 4

optim, *5*, *12, 13*, *29, 30*

pseudo_condseq, 5
pseudo_condseq_XfromU, 7
pseudo_list, *4–8*, 9, *12, 13*, *26*
pseudo_opt, 11

rwm_norm, 14

slice_elliptical, 15
slice_elliptical_mv, 16
slice_genelliptical, 17
slice_genelliptical_mv, 19
slice_hyperrect, 20
slice_latent, 21
slice_quantile, 23
slice_quantile_mv, 24
slice_quantile_mv_seq, *6*, *8*, 25
slice_stepping_out, 27

utility_pseudo, *13*, 29
utility_shrinkslice, *2*, 31