Package 'qslice'

May 30, 2024

Title Implementations of Various Slice Samplers

Version 0.3.1

Description Implementations of the quantile slice sampler of Heiner et al. (2024+, in preparation) 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>,
and independence Metropolis-Hastings sampler.

License MIT + file LICENSE | Apache License 2.0

Depends R (>= 4.1.0)

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2024-05-30 08:00:02 UTC

R topics documented:

---

auc *Area Under the Curve (histogram)*

Description

Calculate the histogram approximation to the area under the curve after restricting the curve to fit within the unit square. Specifically, the highest histogram bar reaches 1 and the support is the unit interval. See Heiner et al. (2024+).

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).

Details

Accepts either samples u or a function y representing a (possibly unnormalized) probability density supported on the unit interval.

Value

The (approximate) area under the curve as a numeric value of length one.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*.

Examples

```
u_samples <- rbeta(10e3, 2, 2)
auc(u = u_samples)
auc(u = u_samples, nbins = 50)
auc(y = function(x) {dbeta(x, 2, 2)}, nbins = 30)
auc(y = function(x) {dbeta(x, 2, 2)}, nbins = 300)
xx <- seq(0.001, 0.999, length = 1000)
auc(x = xx, y = function(x) {dbeta(x, 2, 2)})
auc(x = xx, y = dbeta(xx, 2, 2))
```

---

imh_pseudo *Independence Metropolis-Hastings*

Description

Independence Metropolis-Hastings

Usage

```
imh_pseudo(x, log_target, pseudo)
```

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 for the 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 r function that takes no arguments and generates a single multivariate draw from the proposal distribution.

Value

A list containing the new state, x, and whether the proposed value was accepted, logical accpt.

Examples

```
lf <- function(x) dbeta(x[1], 3, 4, log = TRUE) + dbeta(x[2], 5, 3, log = TRUE)
n_iter <- 100 # 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, degf = 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))
```

---

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

Description

Given a realization of a random vector, generate a the corresponding sequence of conditional pseudo-target inverse CDFs (Heiner et al., 2024+). 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 of values between 0 and 1.
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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*

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, degf = 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) auc(u = draws_u[,k]))
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 of from a 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)
```

Arguments

`u`	A numeric vector of values 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 `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 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, degf = 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) auc(u = draws_u[,k]))
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. (2024+).

Usage

```
pseudo_list(family, params, lb = -Inf, ub = Inf, log_p = FALSE, 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 `degf`
	`cauchy`: location `loc` and scale `sc`
	`norm`: location `loc` and scale `sc`
	`logistic`: location `loc` and scale `sc`
	`beta`: scale `scale1` and scale `scale2`

`lb`	Numeric scalar giving the value of left truncation. Defaults to `-Inf`. Not operative in family `beta`.
`ub`	Numeric scalar giving the value of right truncation. Defaults to `Inf`. Not operative in family `beta`.
`log_p`	(Not implemented) Logical: evaluate distribution and quantile functions using the log probability.
`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

ld: function to evaluate the log density

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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*

Examples

```
pseu <- pseudo_list(family = "t", params = list(loc = 0.0, sc = 1.0, degf = 5),
                    lb = 0.0, ub = Inf) # half t
str(pseu)
pseu$d(1.5)
pseu$ld(1.5)
pseu$p(1.5)
pseu$q(0.8060963)
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 = 2.0, shape2 = 1.0))
str(pseu)
```

```
pseu$d(0.5)
pseu$ld(0.5)
pseu$p(0.5)
pseu$q(0.25)
```

`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., 2024+). Optimize over the selected utility function.

Usage

```
pseudo_opt(
  log_target = NULL,
  samples = NULL,
  type = "samples",
  family = "t",
  degf = c(1, 5, 20),
  lb = -Inf,
  ub = Inf,
  utility_type = "AUC",
  nbins = 100,
  tol_opt = 1e-06,
  tol_int = 0.001,
  plot = TRUE,
  verbose = FALSE
)
```

Arguments

`log_target`	Function to evaluate the log density of the unnormalized 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`.
`family`	String specifying the family of distributions for the pseudo-target. Can be any of the families accepted by pseudo_list.
`degf`	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.
tol_int	Positive numeric scalar that passes to abs.tol in the call to integrate. Defaults to 1.0e-3.
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.

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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*

Examples

```
(pseu <- pseudo_opt(samples = rnorm(1e3), type = "samples",
               family = "t", utility_type = "AUC",
               nbins = 10, plot = TRUE,
               verbose = FALSE))
oldpar <- par(mfrow = c(1,2))
(pseu <- pseudo_opt(log_target = function(x) dnorm(x, log = TRUE),
                type = "function",
                family = "logistic", utility_type = "AUC",
                nbins = 100, 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",
                nbins = 30, plot = TRUE,
                verbose = FALSE))
par(oldpar)
```

`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_target, mu, sigma)
```

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 mean of the supporting normal distribution.
`sigma`	A numeric scalar with the standard deviation of the supporting normal distribution.

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

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

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
draws <- numeric(10) # set to numeric(1e3) for more complete illustration
nEvaluations <- 0L
for (i in seq.int(2, length(draws))) {
  out <- slice_elliptical(draws[i - 1], log_target = lf, mu = 0.5, sigma = 1)
  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_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_target, mu, Sig, is_chol = FALSE)
```

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	Numeric vector with the mean of the supporting normal distribution.
Sig	Positive definite covariance matrix. 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.

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

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

```
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(0.3, nrow = n_iter, ncol = 2)
nEvaluations <- 0L
for (i in seq.int(2, n_iter)) {
  out <- slice_elliptical_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))
  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)
```

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 mean of the supporting normal distribution.
sigma	A numeric scalar with the standard deviation of the supporting normal distribution.
df	Degrees of freedom of Student t pseudo-target.

Value

A list contains 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) # set to numeric(1e3) 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)
```

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`	Numeric vector with the mean of the supporting normal distribution.
`Sig`	Positive definite covariance matrix. Alternatively, a lower-triangular matrix with the Cholesky factor of the covariance 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.

Value

A list contains 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)
```

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 (should, correspond with the support).
L	Numeric vector giving the lower boundary of support in each dimension.
R	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.

Value

A list contains 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)
```

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.

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) # set to numeric(1e3) 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. (2024+).

Usage

```
slice_quantile(x, log_target, pseudo)
```

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).

Value

A list containing three elements:

`x` is the new state.

`u` is the value of the CDF of the psuedo-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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*.

Examples

```
lf <- function(x) dbeta(x, 3, 4, log = TRUE)
pseu <- list(ld = function(x) dbeta(x, shape1 = 1, shape2 = 1, log = TRUE),
             q = function(u) qbeta(u, shape1 = 1, shape2 = 1))
draws <- numeric(10) # set to numeric(1e3) 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., 2024+). The pseudo-target is specified through independent univariate distributions.

Usage

```
slice_quantile_mv(x, log_target, pseudo)
```

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.

Value

A list containing three elements: "x" is the new state, "u" is the value of the CDF of the psuedo-target associated with the returned value, inverse CDF method, 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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*

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))
hist(draws_u[,1], freq = FALSE)
hist(draws_u[,2], freq = FALSE)
auc(u = draws_u[,1])
auc(u = draws_u[,2])
```

`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., 2024+). The pseudo-target is specified as a sequence of growing conditional distributions.

Usage

```
slice_quantile_mv_seq(x, log_target, pseudo_control)
```

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.

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 psuedo-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. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*

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, degf = 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) auc(u = draws_u[,k]))
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)
```

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 maximum number of times to step 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`.

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 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. (2024+).

Usage

```
utility_pseudo(
  pseudo,
  log_target = NULL,
  samples = NULL,
  type = "samples",
  x = NULL,
  nbins = 30,
  plot = TRUE,
  utility_type = "AUC",
  tol_int = 0.001
)
```

Arguments

`pseudo`	List containing the following functions 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 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. Use of "samples" requires specification of samples. Use of "grid" requires specification of x.
x	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). Must match 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.
tol_int	Positive numeric scalar that passes to abs.tol in the call to integrate. Defaults to 1.0e-3.

Details

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

Value

Scalar value of the utility function evaluation.

References

Heiner, M. J., Johnson, S. B., Christensen, J. R., and Dahl, D. B. (2024+), "Quantile Slice Sampling," *arXiv preprint arXiv:###*.

Examples

```
pseu <- pseudo_list(family = "logistic", params = list(loc = 0.0, sc = 0.66))
ltarg <- list(ld = function(x) dnorm(x, log = TRUE))
oldpar <- par(mfrow = c(1,2))
utility_pseudo(pseudo = pseu, log_target = ltarg$ld, type = "function",
               nbins = 100, utility_type = "MSW")
samp <- rnorm(10e3)
utility_pseudo(pseudo = pseu, samples = samp, type = "samples", utility_type = "AUC")
utility_pseudo(pseudo = pseu, samples = samp, type = "samples", utility_type = "MSW")
par(oldpar)
```

Index