Package ‘MfUSampler’

January 20, 2025

Type Package

Title Multivariate-from-Univariate (MfU) MCMC Sampler

Version 1.1.0

Date 2022-12-08

Author Alireza S. Mahani, Mansour T.A. Sharabiani

Maintainer Alireza S. Mahani <alireza.s.mahani@gmail.com>

Description Convenience functions for multivariate MCMC using univariate samplers including: slice sampler with stepout and shrinkage (Neal (2003) <DOI:10.1214/aos/1056562461>), adaptive rejection sampler (Gilks and Wild (1992) <DOI:10.2307/2347565>), adaptive rejection Metropolis (Gilks et al (1995) <DOI:10.2307/2986138>), and univariate Metropolis with Gaussian proposal.

License GPL (>= 2)

Imports ars, coda, dlm

Suggests sns, RcppArmadillo, inline, mvtnorm

NeedsCompilation no

Repository CRAN

Date/Publication 2022-12-08 08:02:34 UTC

Contents

MfU.Control *Constructing Control List for MfU.Sample*

Description

Returns a list of all control parameters needed for univariate samplers. Parameter names (after removing the prefixes) are identical to those used in original packages / source code. To be used with multivariate distributions, all control parameters must have the same length as the dimensionality of state space, either as vectors or lists.

Usage

```
MfU.Control(n, slice.w=1, slice.m=Inf, slice.lower=-Inf, slice.upper=+Inf
  , ars.x=c(-4,1,4), ars.ns=100, ars.m=3, ars.emax=64, ars.lb=FALSE, ars.xlb=0
  , ars.ub=FALSE, ars.xub=0, arms.indFunc = function(x) TRUE
  , unimet.sigma = 1.0)
```

Arguments

`n` Dimensionality of state space, corresponding to `length(x)` in `MfU.Sample`.

`slice.w` Size of the steps for creating slice sampler interval.

`slice.m` Limit on stepout steps.

`slice.lower` Lower bound on support of the distribution.

`slice.upper` Upper bound on support of the distribution.

`ars.x` A vector of starting points for each coordinate, over which log-density is defined.

`ars.ns` Maximum number of points defining the hulls.

`ars.m` Number of starting points.

`ars.emax` Large value for which it is possible to compute an exponential.

`ars.lb` Boolean indicating if there is a lower bound to the domain.

`ars.xlb` Value of the lower bound.

`ars.ub` Boolean indicating if there is an upper bound to the domain.

`ars.xub` Value of the upper bound.

`arms.indFunc` Indicator function of the convex support of the target density.

`unimet.sigma` Standard deviation of Gaussian proposal.

Details

All arguments (aside from `n`) supplied to `MfU.Control` can be vectors (or in the case of `ars.x` a list) of length `n`, in which case they are kept unmodified. Alternatively, a single parameter can be passed into `MfU.Control`, which is then expanded by the function into a vector/list of length `n` by simple replication. Each element of the resulting vector/list is used for one of the `n` visited coordinates during the univariate sampling cycles. Naming and description of arguments for each univariare sampler is kept in maximal consistency with original source codes / libraries.

Value

A list with 4 elements, `slice`, `ars`, `arms`, and `unimet`, each containing elements of the same name as their corresponding arguments in the function call.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

References

Gilks WR and Wild P (1992). Adaptive Rejection Sampling. *Applied Statistics*, **41**, 337-348.

Gilks WR, Best NG, and Tan KKC (1995). Adaptive rejection Metropolis sampling within Gibbs sampling. *Applied Statistics*, **44**, 455-472.

Mahani A.S and Sharabiani M.T.A. (2017). Multivariate-From-Univariate MCMC Sampler: The R Package MfUSampler. Journal of Statistical Software, Code Snippets, 78(1), 1-22. doi:10.18637/jss.v078.c01

Neal R.M. (2003). Slice Sampling. *Annals of Statistics*, **31**, 705-767.

Examples

```
# default control a for 10-dimensional space
mycontrol <- MfU.Control(10)
# setting a lower bound of 0 for last coordinate
mycontrol <- MfU.Control(10, slice.lower=c(rep(-Inf,9),0.0))
```

`MfU.Sample` *Drawing MCMC Samples from a Multivariate Distribution Using a Univariate Sampler*

Description

This function is an extended Gibbs wrapper around univariate samplers to allow for drawing samples from multivariate distributions. Four univariate samplers are currently available: 1) slice sample with stepout and shrinkage (Neal 2003, using Radford Neal's `R` code from his homepage), and 2) adaptive rejection sampling (Gilks and Wild 1992, using `ars` function from **ars** package), 3) adaptive rejection Metropolis (Gilks et al 1995, using `arms` function from **HI** package), and 4) univariate Metropolis with Gaussian proposal. The wrapper performs a full cycle of univariate sampling steps, one coordinate at a time. In each step, the latest sample values obtained for other coordinates are used to form the conditional distributions.

Usage

```
MfU.Sample(x, f, uni.sampler = "slice", ...
  , control = MfU.Control(length(x)))
MfU.Sample.Run(x, f, uni.sampler = c("slice", "ars", "arms", "unimet"), ...
  , control = MfU.Control(length(x)), nsmp = 10)
```

Arguments

`x` Initial value for the multivariate distribution. It must be a numeric vector.

`f` The multivariate log-density to be sampled. For any of `{"slice", "arms", "unimet"}`, the function must return the log-density (up to a constant). For `"ars"`, the function must accept a boolean flag `grad` and return the log-density (`grad=FALSE`) or its gradient (`grad=TRUE`).

`uni.sampler` Name of univariate sampler to be used. Default is `"slice"`, standing for the univariate Slice Sampler with stepout and shrinkage, as described in Neal (2003). Other options are `"ars"`, referring to adaptive rejection sampling algorithm of Gilks and Wild (1992), `"arms"`, referring to adaptive rejection Metropolis algorithm of Gilks et al (1995), and `"unimet"`, referring to univariate Metropolis with Gaussian proposal.

`...` Other arguments to be passed to `f`.

`control` List of parameters controlling the execution of univariate samplers. See `MfU.Control`.

`nsmp` Number of MCMC samples to generate in `MfU.Sample.Run`.

Details

In the case of ARS, the wrapper is an exact implementation of Gibbs sampling (Geman and Geman 1984), while for the other 3 samplers the wrapper can be considered a generalization of Gibbs sampling, where instead of drawing a sample from each conditional distribution, we perform a state transition for which the conditional probability is an invariant distribution. The wrapper takes advantage of the fact that conditional distributions for each coordinate are simply proportional to the full joint distribution, with all other variables held constant, at their most recent sampled values. Note that ARS requires log-concavity of the conditional distributions. Log-concavity of the full multivariate distribution is sufficient but not necessary for univariate conditionals to be log-concave. Slice sampler (default option) is derivative-free, robust with respect to choice of tuning parameters, and can be applied to a wider collection of multivariate distributions as a drop-in method with good results. Multivariate samplers such as Metropolis (Bishop 2006) or Stochastic Newton Sampler (Mahani et al 2014) do not require our wrapper.

Value

For `MfU.Sample`, a vector of length `length(x)`, representing a sample from the multivariate log-density `f`; for `MfU.Sample.Run`, an object of class `"MfU"`, which is a matrix of sampled values, one sampler per row (`nsmp` rows), with sampling time attached as attribute `"t"`.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

References

Bishop C.M. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

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

Gilks W.R. and Wild P. (1992). Adaptive Rejection Sampling. *Applied Statistics*, **41**, 337-348.

Gilks W.R., Best N.G., and Tan K.K.C. (1995) Adaptive rejection Metropolis sampling within Gibbs sampling. *Applied Statistics*, **44**, 455-472.

Mahani A.S., Hasan A., Jiang M. and Sharabiani M.T.A. (2016). Stochastic Newton Sampler: The R Package sns. Journal of Statistical Software, Code Snippets, 74(2), 1-33. doi:10.18637/jss.v074.c02

Mahani A.S and Sharabiani M.T.A. (2017). Multivariate-From-Univariate MCMC Sampler: The R Package MfUSampler. Journal of Statistical Software, Code Snippets, 78(1), 1-22. doi:10.18637/jss.v078.c01

Neal R.M. (2003). Slice Sampling. *Annals of Statistics*, **31**, 705-767.

Examples

```
z <- c(1, 4, 7, 10, 13, 16, 19, 24)
m1.prior <- c(17, 26, 39, 27, 35, 37, 26, 23)
m2.prior <- c(215, 218, 137, 62, 36, 16, 13, 15)
m1.current <- c(46, 52, 44, 54, 38, 39, 23, 52)
m2.current <- c(290, 211, 134, 91, 53, 42, 23, 32)

m1.total <- m1.prior + m1.current
m2.total <- m2.prior + m2.current

logpost.retin <- function(beta, z, m1, m2
  , beta0 = rep(0.0, 3), W = diag(1e+6, nrow = 3)) {
  X <- cbind(1, z, z^2)

  beta <- as.numeric(beta)
  Xbeta <- X %*% beta
  log.prior <- -0.5 * t(beta - beta0) %*% solve(W) %*% (beta - beta0)
  log.like <- -sum((m1 + m2) * log(1 + exp(-Xbeta)) + m2 * Xbeta)
  log.post <- log.prior + log.like

  return (log.post)
}

nsmp <- 1000
beta.ini <- c(0.0, 0.0, 0.0)
beta.smp <- MfU.Sample.Run(beta.ini, logpost.retin, nsmp = nsmp
  , z = z, m1 = m1.total, m2 = m2.total)
summary(beta.smp)
```

predict.MfU *Sample-based prediction using "MfU" Objects*

Description

Method for sample-based prediction using the output of MfU.Sample.Run.

Usage

```
## S3 method for class 'MfU'
predict(object, fpred, ...)
## S3 method for class 'predict.MfU'
summary(object, start = round(nrow(object)/2) + 1
  , end = nrow(object), thin = 1
  , quantiles = c(0.025, 0.5, 0.975), ...)
## S3 method for class 'summary.predict.MfU'
print(x, n = 6L, ...)
```

Arguments

`object`	Object of class "MfU" (output of `MfU.Sample.Run`) or "predict.MfU" (output of `predict.MfU`).
`fpred`	Prediction function, accepting a single value for the state vector and producing a vector of outputs.
`start`	Which iteration to start from for calculating sample statistics.
`end`	Last iteration to use for calculating sample statistics. Defaults to last iteration.
`thin`	One out of `thin` samples are kept for calculating sample statistics. Default is `1`, using all samples within specified range.
`quantiles`	Values for which sample-based quantiles are calculated.
`x`	An object of class "summary.predict.MfU".
`n`	Number of rows of prediction matrix to print.
`...`	Arguments passed to/from other functions.

Value

`predict.MfU` produces a matrix with number of rows equal to the length of prediction vector produces by `fpred`. Its numnber of columns is equal to the number of samples used within the user-specified range, and after thinning (if any). `summary.predict.MfU` produces sample-based prediction mean, standard deviation, quantiles, and effective sample size.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

References

Mahani A.S and Sharabiani M.T.A. (2017). Multivariate-From-Univariate MCMC Sampler: The R Package MfUSampler. Journal of Statistical Software, Code Snippets, 78(1), 1-22. doi:10.18637/jss.v078.c01

See Also

`MfU.Sample.Run`

summary.MfU *Summarizing and plotting "MfU" Objects*

Description

Methods for summarizing and plotting the output of `MfU.Sample.Run`.

Usage

```
## S3 method for class 'MfU'
summary(object, start = round(nrow(object)/2) + 1
  , end = nrow(object), thin = 1
  , quantiles = c(0.025, 0.5, 0.975), ...)
## S3 method for class 'summary.MfU'
print(x, ...)
## S3 method for class 'MfU'
plot(x, start = round(nrow(x)/2) + 1
  , end = nrow(x), thin = 1, ...)
```

Arguments

`object` An object of class "MfU", typically the output of `MfU.Sample.Run`.

`start` Which iteration to start from for calculating sample statistics.

`end` Last iteration to use for calculating sample statistics. Defaults to last iteration.

`thin` One out of `thin` samples are kept for calculating sample statistics. Default is `1`, using all samples within specified range.

`quantiles` Values for which sample-based quantiles are calculated.

`...` Arguments passed to `summary.mcmc` and `plot.mcmc` functions in **coda** package.

`x` For `plot.MfU`, an object of class "MfU", typically the output of `MfU.Sample.Run`; for `print.summary.MfU`, an object of class `summary.MfU`, typically the output of `summary.MfU` function.

Value

These functions are thin wrappers around `summary.mcmc` and `plot.mcmc`. See **coda** package documentation for details.

Author(s)

Alireza S. Mahani, Mansour T.A. Sharabiani

References

Mahani A.S and Sharabiani M.T.A. (2017). Multivariate-From-Univariate MCMC Sampler: The R Package MfUSampler. Journal of Statistical Software, Code Snippets, 78(1), 1-22. doi:10.18637/jss.v078.c01

See Also

`MfU.Sample.Run`

Index