Package ‘rjmcmc’

October 14, 2022

Type Package

Title Reversible-Jump MCMC Using Post-Processing

Version 0.4.5

Date 2019-07-07

Description Performs reversible-jump Markov chain Monte Carlo (Green, 1995)
<doi:10.2307/2337340>, specifically the restriction introduced by
Barker & Link (2013) <doi:10.1080/00031305.2013.791644>. By utilising
a 'universal parameter' space, RJMCMC is treated as a Gibbs sampling
problem. Previously-calculated posterior distributions are used to
quickly estimate posterior model probabilities. Jacobian matrices are
found using automatic differentiation. For a detailed description of
the package, see Gelling, Schofield & Barker (2019)
<doi:10.1111/anzs.12263>.

License GPL-3

Depends madness, R (>= 3.2.0)

Imports utils, coda, mvtnorm

Suggests FSAdata

RoxygenNote 6.1.0

LazyData TRUE

NeedsCompilation no

Author Nick Gelling [aut, cre],
Matthew R. Schofield [aut],
Richard J. Barker [aut]

Maintainer Nick Gelling <nickcjgelling@gmail.com>

Repository CRAN

Date/Publication 2019-07-09 14:20:02 UTC

R topics documented:

adiff 2
defaultpost 3

getsampler 4
rjmcmc 6
rjmcmcpost 6
rjmethods 8

Index **10**

`adiff` *Automatic Differentiation Using Madness*

Description

A wrapper function to the functionality of the `madness` package. Converts a given `x` to a madness object and then applies `func` to it, returning the result and the Jacobian for the transformation `func`. `adiff` is used by the `rjmcmcpost` function.

Usage

```
adiff(func, x, ...)
```

Arguments

`func`	The function to be differentiated (usually user-defined).
`x`	The values at which to evaluate the function `func`.
`...`	Further arguments to be passed to `func`.

Value

A numeric vector or matrix containing the result of the function evaluation `func(x, ...)`. The `"gradient"` attribute of this object contains the Jacobian matrix of the transformation `func`.

References

Pav, S. E. (2016) Madness: Automatic Differentiation of Multivariate Operations.

See Also

`madness`

Examples

```
x2x3 = function(x){
return(c(x^2, x^3))
}

y = adiff(x2x3, c(5,6))
attr(y, "gradient")
```

defaultpost *Perform Post-Processing Using Default Bijections*

Description

Performs Bayesian multimodel inference, estimating Bayes factors and posterior model probabilities for N candidate models. Unlike `rjmcmcpost`, this function uses a default bijection scheme based on approximating each posterior by a multivariate normal distribution. The result is reminiscent of the algorithm of Carlin & Chib (1995) with a multivariate normal pseudo-prior. Transformation Jacobians are computed using automatic differentiation so do not need to be specified.

Usage

```
defaultpost(posterior, likelihood, param.prior, model.prior,
  chainlength = 10000, TM.thin = chainlength/10, progress = TRUE,
  save.all = TRUE)
```

Arguments

`posterior`	A list of N matrices containing the posterior distributions under each model. Generally this will be obtained from MCMC output. Note that each parameter should be real-valued so some parameters may need to be transformed, using logarithms for example.
`likelihood`	A list of N functions specifying the log-likelihood functions for the data under each model.
`param.prior`	A list of N functions specifying the prior distributions for each model-specific parameter vector.
`model.prior`	A numeric vector of the prior model probabilities. Note that this argument is not required to sum to one as it is automatically normalised.
`chainlength`	How many iterations to run the Markov chain for.
`TM.thin`	How regularly to calculate transition matrices as the chain progresses.
`progress`	A logical determining whether a progress bar is drawn.
`save.all`	A logical determining whether to save the value of the universal parameter at each iteration, as well as the corresponding likelihoods, priors and posteriors. If `TRUE`, the output object occupies significantly more memory.

Value

Returns an object of class `rj` (see `rjmethods`). If `save.all=TRUE`, the output has named elements `result`, `densities`, `psidraws`, `progress` and `meta`. If `save.all=FALSE`, the `densities` and `psidraws` elements are omitted.

`result` contains useful point estimates, `progress` contains snapshots of these estimates over time, and `meta` contains information about the function call.

References

Carlin, B. P. and Chib, S. (1995) Bayesian Model Choice via Markov Chain Monte Carlo Methods. *Journal of the Royal Statistical Society, Series B, 473-484*.

Barker, R. J. and Link, W. A. (2013) Bayesian multimodel inference by RJMCMC: A Gibbs sampling approach. *The American Statistician, 67(3), 150-156*.

See Also

adiff rjmcmcpost

Examples

```
## Comparing two binomial models -- see Barker & Link (2013) for further details.

y=c(8,16); sumy=sum(y)
n=c(20,30); sumn=sum(n)

L1=function(p){if((all(p>=0))&&(all(p<=1))) sum(dbinom(y,n,p,log=TRUE)) else -Inf}
L2=function(p){if((p[1]>=0)&&(p[1]<=1)) sum(dbinom(y,n,p[1],log=TRUE)) else -Inf}

p.prior1=function(p){sum(dbeta(p,1,1,log=TRUE))}
p.prior2=function(p){dbeta(p[1],1,1,log=TRUE)+dbeta(p[2],17,15,log=TRUE)}

draw1=matrix(rbeta(2000,y+1,n-y+1), 1000, 2, byrow=TRUE)  ## full conditional posterior
draw2=matrix(c(rbeta(1000,sumy+1,sumn-sumy+1),rbeta(1000,17,15)), 1000, 2)

out=defaultpost(posterior=list(draw1,draw2), likelihood=list(L1,L2),
              param.prior=list(p.prior1,p.prior2), model.prior=c(1,1), chainlength=1000)
```

---

getsampler *Define Function To Sample From MCMC Output*

Description

A utility function which accepts a matrix of MCMC output and creates a function which samples from the posterior distribution for the parameters of the model.

Usage

```
getsampler(modelfit, sampler.name = "post.draw", order = "default",
  envir = .GlobalEnv)
```

Arguments

`modelfit`	A matrix of output from a previously-run MCMC algorithm, with one column per variable and one row per iteration.
`sampler.name`	A string giving the desired name for the function to be defined.
`order`	A numeric vector of indices specifying the desired parameters to extract from `modelfit`, and in which order.
`envir`	The environment in which to define the sampling function. Defaults to the global environment.

Value

A function is defined in `envir` which randomly samples from the posterior distribution for the parameters. Note that this function does not take any arguments. A function generated in this way is suitable for passing to the `rjmcmcpost` function.

References

Plummer, M. (2003) JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd international workshop on distributed statistical computing (Vol. 124, p. 125).*

See Also

`rjmcmcpost`

Examples

```
# Generate synthetic 'MCMC output' for a model with 3 parameters. There is
# one column per parameter, and 1000 iterations.
matrix_output = matrix(c(runif(1000,0,1), rnorm(1000,5,2), rgamma(1000,2,2)), 1000, 3)

getsampler(modelfit=matrix_output, sampler.name="posterior1")
set.seed(100)
posterior1()

## Alternatively
posterior1b = getsampler(modelfit=matrix_output)  # this successfully defines a function named
# posterior1b but also defines an identical function corresponding to the value
# of sampler.name, i.e. the default "post.draw" in this case.
set.seed(100)
posterior1b()
set.seed(100)
posterior1()
```

`rjmcmc` *The rjmcmc Package*

Description

Performs reversible-jump MCMC (Green, 1995), specifically the reformulation introduced by Barker & Link (2013). Using a 'universal parameter' space, RJMCMC is treated as Gibbs sampling making it simpler to implement. The required Jacobian matrices are calculated automatically, utilising the `madness` package.

Functions

`rjmcmcpost` `defaultpost` `adiff` `getsampler`

Methods

`rjmethods`

References

Green, P. J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika, 711-732.*

Barker, R. J. and Link, W. A. (2013) Bayesian multimodel inference by RJMCMC: A Gibbs sampling approach. *The American Statistician, 67(3), 150-156.*

`rjmcmcpost` *Perform Reversible-Jump MCMC Post-Processing*

Description

Performs Bayesian multimodel inference, estimating Bayes factors and posterior model probabilities for N candidate models. Using the 'universal parameter' restriction in Barker & Link (2013), RJMCMC is treated as a Gibbs sampling problem, where the algorithm alternates between updating the model and the model specific parameters. Transformation Jacobians are computed using automatic differentiation so do not need to be specified.

Usage

```
rjmcmcpost(post.draw, g, ginv, likelihood, param.prior, model.prior,
  chainlength = 10000, TM.thin = chainlength/10, save.all = FALSE,
  progress = TRUE)
```

Arguments

`post.draw`	A list of N functions that randomly draw from the posterior distribution under each model. Generally these functions sample from the output of a model fitted using MCMC.
`g`	A list of N functions specifying the bijections from the universal parameter `psi` to each model-specific parameter set.
`ginv`	A list of N functions specifying the bijections from each model-specific parameter set to `psi`. These are the inverse transformations of `g`.
`likelihood`	A list of N functions specifying the log-likelihood functions for the data under each model.
`param.prior`	A list of N functions specifying the log-prior distributions for each model-specific parameter vector.
`model.prior`	A numeric vector of the prior model probabilities. Note that this argument is not required to sum to one as it is automatically normalised.
`chainlength`	How many iterations to run the Markov chain for.
`TM.thin`	How regularly to calculate transition matrices as the chain progresses.
`save.all`	A logical determining whether to save the value of the universal parameter at each iteration, as well as the corresponding likelihoods, priors and posteriors. If `TRUE`, the output object occupies significantly more memory.
`progress`	A logical determining whether a progress bar is drawn.

Value

Returns an object of class `rj` (see `rjmethods`). If `save.all=TRUE`, the output has named elements `result`, `densities`, `psidraws`, `progress` and `meta`. If `save.all=FALSE`, the `densities` and `psidraws` elements are omitted.

`result` contains useful point estimates, `progress` contains snapshots of these estimates over time, and `meta` contains information about the function call.

References

Barker, R. J. and Link, W. A. (2013) Bayesian multimodel inference by RJMCMC: A Gibbs sampling approach. *The American Statistician, 67(3), 150-156.*

See Also

`adiff` `getsampler` `defaultpost`

Examples

```
## Comparing two binomial models -- see Barker & Link (2013) for further details.

y=c(8,16); sumy=sum(y)
n=c(20,30); sumn=sum(n)

L1=function(p){if((all(p>=0))&&(all(p<=1))) sum(dbinom(y,n,p,log=TRUE)) else -Inf}
```

```
L2=function(p){if((p[1]>=0)&&(p[1]<=1)) sum(dbinom(y,n,p[1],log=TRUE)) else -Inf}

g1=function(psi){p=psi}
g2=function(psi){w=n[1]/sum(n); p=c(w*psi[1]+(1-w)*psi[2],psi[2])}
ginv1=function(p){p}
ginv2=function(p){c(sum(n)/n[1]*p[1]-n[2]/n[1]*p[2],p[2])}

p.prior1=function(p){sum(dbeta(p,1,1,log=TRUE))}
p.prior2=function(p){dbeta(p[1],1,1,log=TRUE)+dbeta(p[2],17,15,log=TRUE)}

draw1=function(){rbeta(2,y+1,n-y+1)}
draw2=function(){c(rbeta(1,sumy+1,sumn-sumy+1),rbeta(1,17,15))}

out=rjmcmcpost(post.draw=list(draw1,draw2), g=list(g1,g2), ginv=list(ginv1,ginv2),
               likelihood=list(L1,L2), param.prior=list(p.prior1,p.prior2),
               model.prior=c(0.5,0.5), chainlength=1500)
```

rjmethods *Methods for the rj Class*

Description

An object of class `rj` is returned from the functions `rjmcmcpost` or `defaultpost`. The following methods can be applied to an object of this class. See Details for more information.

Usage

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

## S3 method for class 'rj'
plot(x, legend = TRUE, col = "maroon4", ylim = c(0, 1),
  lwd = 2, lty = c(1, 1, 1), ...)

## S3 method for class 'rj'
summary(object, quantiles = c(0.025, 0.25, 0.5, 0.75,
  0.975), ...)
```

Arguments

`x, object`	An object of class `rj`.
`...`	Further arguments to the generic method.
`legend, col, ylim, lwd, lty`	Some useful graphical arguments to the generic `plot` method.
`quantiles`	The desired density quantiles for `summary` to find.

Details

The `print` method prints the point estimates obtained from the algorithm, including the transition matrix, posterior model probabilities and Bayes factors.

The `plot` method plots how the estimates of the posterior probabilities changed as the algorithm progressed, illustrating convergence.

The `summary` method returns quantiles of the posterior densities for each model (as well as likelihoods and prior densities). The point estimates as in `print` are also returned. Note that this requires `save.all` must be `TRUE` in the `rjmcmcpost` call.

Index

adiff, 2, *4*, *6, 7*

defaultpost, 3, *6–8*

getsampler, 4, *6, 7*

madness, *2*

plot.rj *(*rjmethods*)*, 8
print.rj *(*rjmethods*)*, 8

rjmcmc, 6
rjmcmc-package *(*rjmcmc*)*, 6
rjmcmcpost, *2–6*, 6, *8*
rjmethods, *3*, *6, 7*, 8

summary.rj *(*rjmethods*)*, 8