Package: mmcc (via r-universe)

July 21, 2024

Type Package

Title tidy mcmc.list using data.table

Version 0.0.9.9000

Description Tidy up, diagnose, and visualise your mcmc samples quickly and easily so you can get on with your analysis.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports data.table, magrittr, purrr, coda, generics, graphics, stats, glue

RoxygenNote 7.1.2

Suggests knitr, rmarkdown, ggplot2, covr, greta, bayesplot, reticulate, rjags, testthat, rstan (>= 2.17.0), spelling

VignetteBuilder knitr

Depends R (>= 3.1.2)

Roxygen list(markdown = TRUE)

Language en-US

Repository https://njtierney.r-universe.dev

RemoteUrl https://github.com/njtierney/mmcc

RemoteRef HEAD

RemoteSha f1f271bf565edfaa21c2f4a1979dc478e450a9d9

Contents

`diag_autocorr` *Diagnostic: Calculate the autocorrelation for each chain and parameter*

Description

Diagnostic: Calculate the autocorrelation for each chain and parameter

Usage

```
diag_autocorr(x, lags = NULL)
```

Arguments

`x`	mcmc.list object or dataframe creatd by `mcmc_to_dt`
`lags`	integer the lag value that you want to use

Value

a data.table data frame with the autocorrelation at each lag for each chain and parameter columns `lag`, `acf`, `chain`, and `parameter`

Examples

```
library(coda)
data(line)
line_acf <- diag_autocorr(line)
line_acf
```

diag_ess *Diagnostic: Calculate the effective sample size for each chain and parameter*

Description

Diagnostic: Calculate the effective sample size for each chain and parameter

Usage

```
diag_ess(x)
```

Arguments

x	mcmc.list object or dataframe creatd by mcmc_to_dt

Value

a data.table data frame with the effective sample size for each chain and parameter

Examples

```
library(coda)
data(line)
line_ess <- diag_ess(line)
line_ess
```

diag_mc_stderr *Batch Means Standard Errors*

Description

Calculate Monte Carlo Standard Errors Using Batch Means

Usage

```
diag_mc_stderr(x, b_size = "sqroot", warn = TRUE)
```

Arguments

x	'mcmc.list' or 'data.table' object
b_size	character, method for determining size of batch (see Details)
warn	logical, give a warning of there are too few samples in the MCMC output

Details

For the batch size, the options are 'sqroot', for using the using the square root of the MCMC output length, or 'cuberoot', for using the cube root of the output length. The default is 'sqroot'.

References

Galin L. Jones, Murali Haran, Brian S. Caffo, and Ronald Neath (2006). "Fixed-Width Output Analysis for Markov Chain Monte Carlo," Journal of the American Statistical Association, 101, 1537–1547

Examples

```
library(coda)
data(line)
r <- diag_mc_stderr(line)
r
```

`example_jags_model` *Example JAGS Model*

Description

This is an example JAGS model to use for examples in the `mmcc` package. The model is fit as a basic linear regression, with uniform priors on beta0 and beta1, where y is assumed to be normal, with mean mu and precision tau. The model is drawn from the vignette, "Model summaries for a Bayesian linear regression", and the code to generate it can be found in the `data-raw` folder.

Usage

```
data(example_jags_model)
```

Format

An object of class `jags` of length 8.

Examples

```
library(rjags)
example_jags_model$recompile()
model_dic <- dic.samples(example_jags_model, n.iter = 1000)
glance(model_dic)
```

example_stan_model *Example STAN Model*

Description

This is an example STAN model to use for examples in the mmcc package. It was created with the example code given below in examples.

Usage

```
data(example_stan_model)
```

Format

An object of class stanfit of dimension 5 x 4 x 3.

Examples

```
mcmc_to_dt(example_stan_model)
```

glance.dic *Glance upon your dic samples to get the penalised deviance*

Description

This provides a one-row dataframe (data.table) with information on the overall deviance, effective number of parameters (when type was pD) or optimism (when type was popt), and resulting penalised deviance.

Usage

```
## S3 method for class 'dic'
glance(x, ...)
```

Arguments

x	a dic object generated by rjags::dic.samples
...	(optional) additional arguments to pass

Value

a one-row dataframe of summary information of the dic samples

Author(s)

Sam Clifford, <sj.clifford@gmail.com>

Examples

```
library(rjags)
example_jags_model$recompile()
model_dic <- dic.samples(example_jags_model, n.iter = 1000)
glance(model_dic)
```

`glance.mcmc.list` *Glance upon your mcmclist to get summary information*

Description

This provides a one-row dataframe with information on number of chains, the number of variables, the number of iterations, and the lower and upper values for effective sample size (ess), and rhat.

Usage

```
## S3 method for class 'mcmc.list'
glance(x, ...)
```

Arguments

`x`	an mcmc.list object
`...`	(optional) additional arguments to pass

Value

a one-row dataframe of summary information of the mcmc model

Examples

```
library(coda)
data(line)
glance(line)
```

`mcmc-dims` *Dimensions of MCMC objects*

Description

Retrieve dimensions of an MCMC object:

- `n_chain(x)` the number of chains
- `n_var(x)` the number of variables
- `n_iter(x)` the number of iterations

Usage

```
n_chain(x)

n_iter(x)

n_var(x)
```

Arguments

x	an mcmc object, see "Details" for a list of supported mcmc objects.

Details

This is similar to `coda::nchain()` but more general, working for the following classes: * `mcmc` * `mcmc.list` * `data.table` (generated by `mcmc_to_dt`) * `stanfit` * `jags`

---

mcmc_to_dt *Convert* `mcmc.list` *to a tidy data.table object*

Description

`mcmc_to_dt` use data.table to return a tidy dataframe from an "mcmc.list", or "stan" object.

Usage

```
mcmc_to_dt(mcmc_object, ...)
```

Arguments

mcmc_object	an object of class "mcmc.list", as you would find with fitting a model using `jags.model()`, and `coda.samples`, or "stan", from fitting a stan model.
...	additional arguments

Value

a tidy data.table dataframe of MCMC sample

mcmc_to_dt.mcmc.list *Convert* mcmc.list *to a tidy data.table object*

Description

mcmc_to_dt use data.table to return a tidy dataframe from an "mcmc.list", or "stan" object.

Usage

```
## S3 method for class 'mcmc.list'
mcmc_to_dt(mcmc_object, ..., colnames = NULL)
```

Arguments

mcmc_object	an object of class "mcmc.list", as you would find with fitting a model using jags.model(), and coda.samples, or "stan", from fitting a stan model.
...	additional arguments
colnames	which parameters we want from mcmc_object, if NULL then all columns get selected

Examples

```
library(coda)
data(line)
mcmc_to_dt(line)
```

mcmc_to_dt.stanfit *MCMC tidiers (draft) for STAN*

Description

MCMC tidiers (draft) for STAN

Usage

```
## S3 method for class 'stanfit'
mcmc_to_dt(mcmc_object, ...)
```

Arguments

mcmc_object	an object of class "mcmc.list", as you would find with fitting a model using jags.model(), and coda.samples, or "stan", from fitting a stan model.
...	additional arguments

Examples

```
mcmc_to_dt(example_stan_model)
```

`n_sims` *Return the number of simulations*

Description

Return the number of simulations

Usage

```
n_sims(x)
```

Arguments

`x`	mcmc list

Value

integer of number of simulations

Author(s)

Nicholas Tierney

`thin_dt` *thin_dt*

Description

post-hoc thinning of MCMC chains which have been converted to a data.table

Usage

```
thin_dt(dt, thin = 1)
```

Arguments

`dt`	an object of class "data.table" returned from `mcmc_to_dt`
`thin`	thinning interval

Value

a data.table dataframe

Examples

```
library(coda)
data(line)
mcmc_dt <- mcmc_to_dt(line)
thin_dt(mcmc_dt, thin = 10)
thin_dt(mcmc_dt, thin = 2)
thin_dt(mcmc_dt, thin = 20)
```

`tidy.mcmc.list` *Return a tidy data summary of an MCMC object*

Description

`tidy.mcmc.list` is a function that behaves like those from `broom`. It takes an mcmc.list object from `coda.samples` and return a data frame that summarises each parameters with its mean and quantiles and returns the output as a data.table object. This can be called as `tidy`. Currently summarises over all chains.

Usage

```
## S3 method for class 'mcmc.list'
tidy(x, conf_level = c(0.95), chain = FALSE, colnames = NULL, ...)
```

Arguments

`x`	object of class "mcmc.list", as you would find with fitting a model using `jags.model()`, and `coda.samples`.
`conf_level`	level of the credible interval to be calculated. Can be multiple values.
`chain`	whether or not to summarise each parameter for each chain
`colnames`	which parameters we want from `mcmc_object`, if `NULL` then all columns get selected
`...`	extra arguments

Value

a data.table containing parameter summaries

Author(s)

Sam Clifford, <sj.clifford@gmail.com>

Examples

```
library(coda)
data(line)
tidy(line)
# Optionally ask for a subset of parameters with a vector of `colnames`,
# and summarise for each chain:
tidy(line,
     chain = TRUE,
     colnames=c("alpha"))
# can provide two levels of confidence:
tidy(line, conf_level = c(0.95, 0.50))
tidy(line, conf_level = c(0.95))
tidy(line, conf_level = c(0.89, 0.25))
```

Index