Package 'dcmle'

July 4, 2023

Type Package

Title Hierarchical Models Made Easy with Data Cloning

Version 0.4-1

Date 2023-07-04

Author Peter Solymos

Maintainer Peter Solymos <`psolymos@gmail.com`>

Description S4 classes around infrastructure provided by the
'coda' and 'dclone' packages to make package development easy as a breeze
with data cloning for hierarchical models.

License GPL-2

Depends R (>= 2.15.0), dclone (>= 2.0-0)

Imports coda, methods, stats4, lattice

Suggests MASS, parallel, rjags

SystemRequirements JAGS (>= 3.0.0)

URL `https://groups.google.com/forum/#!forum/dclone-users`,
`https://datacloning.org`, `https://github.com/datacloning/dcmle`

BugReports `https://github.com/datacloning/dcmle/issues`

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2023-07-04 21:13:01 UTC

R topics documented:

`dcmle-package` *Hierarchical Models Made Easy with Data Cloning*

Description

S4 classes around infrastructure provided by the dclone package to make package development with data cloning for hierarchical models easy as a breeze.

Details

The package defines S4 object classes for plain BUGS models (`"gsFit"`, after BU*GS*/JA*GS*), and BUGS models made ready for data cloning (`"dcFit"`). It also defines virtual classes for S3 object classes defined in the **dclone** and **coda** packages.

The S4 class `"dcmle"` is a fitted model object containing MCMC results as returned by the `dcmle` function. These object classes are easily extensible to allow inclusion into functions fitting specific models to the data (see Examples).

Author(s)

Peter Solymos

Maintainer: Peter Solymos <psolymos@gmail.com>

References

Forum: `https://groups.google.com/forum/#!forum/dclone-users`

Issues: `https://github.com/datacloning/dcmle/issues`

Data cloning website: `https://datacloning.org`

See Also

Fitting wrapper function: `dcmle`

Object classes: `"dcmle"`, `"codaMCMC"`, `"dcCodaMCMC"`

Creator functions `makeGsFit` and `makeDcFit`

Examples

```
## Data and model taken from Ponciano et al. 2009
## Ecology 90, 356-362.

## Function to create template object for the Beverton-Holt model
## R CMD check will not choke on character representation of model
## the convenient makeDcFit creator function is used here
bevholtFit <-
function(y) {
makeDcFit(
  data = list(ncl=1, n=length(y), Y=dcdim(data.matrix(y))),
  model = structure(
    c("model {",
      "  for (k in 1:ncl) {",
      "    for(i in 2:(n+1)) {",
      "      Y[(i-1), k] ~ dpois(exp(X[i, k]))",
      "      X[i, k] ~ dnorm(mu[i, k], 1 / sigma^2)",
      "      mu[i,k] <- X[(i-1),k]+log(lambda)-log(1+beta*exp(X[(i-1),k]))",
      "    }",
      "    X[1, k] ~ dnorm(mu0, 1 / sigma^2)",
      "  }",
      "  beta ~ dlnorm(-1, 1)",
      "  sigma ~ dlnorm(0, 1)",
      "  tmp ~ dlnorm(0, 1)",
      "  lambda <- tmp + 1",
      "  mu0 <- log(2)  + log(lambda) - log(1 + beta * 2)",
      "}"),
      class = "custommodel"),
  multiply = "ncl",
  unchanged = "n",
  params <- c("lambda","beta","sigma"))
}
## S4 class 'bevholtMle' extends the 'dcmle' class
```

```
## it can have additional slots
setClass("bevholtMle",
    representation(y="numeric", title="character"),
    contains = "dcmle")
## Function to fit the Beverton-Holt model to data
bevholt <- function(y, n.clones, ...) {
    new("bevholtMle",
        dcmle(bevholtFit(y), n.clones=n.clones, ...),
        y = y,
        title = "Beverton-Holt Model")
}
## Show method with appropriate heading
setMethod("show", "bevholtMle", function(object)
    show(summary(as(object, "dcmle"), object@title)))
paurelia <- c(17,29,39,63,185,258,267,392,510,
    570,650,560,575,650,550,480,520,500)
## Not run:
(m <- bevholt(paurelia, n.clones=2, n.iter=1000))
vcov(m)
m@y

## End(Not run)
```

`chanames` *coda package related generic functions*

Description

coda package related generic functions.

Usage

```
chanames(x, ...)
varnames(x, ...)
```

Arguments

`x`	MCMC object.
`...`	Other arguments.

Value

See corresponding help pages.

Author(s)

Peter Solymos

See Also

`chanames` `varnames`

`codaMCMC-class` *Class* `"codaMCMC"`

Description

An S4 representation of an mcmc.lits object of the **coda** package.

Objects from the Class

Objects can be created by calls of the form `new("codaMCMC", ...)`.

Slots

`values`**:** Object of class `"numeric"`, values from the posterior sample of length `niter * nvar * nchains`.

`varnames`**:** Object of class `"character"`, variable names.

`start`**:** Object of class `"integer"`, start of iterations.

`end`**:** Object of class `"integer"`, end of iterations.

`thin`**:** Object of class `"integer"`, thinning value.

`nchains`**:** Object of class `"integer"`, number of chains.

`niter`**:** Object of class `"integer"`, number of iterations.

`nvar`**:** Object of class `"integer"`, number of variables

Methods

[`signature(x = "codaMCMC")`: ...

[[`signature(x = "codaMCMC")`: ...

acfplot `signature(x = "codaMCMC")`: ...

as.array `signature(x = "codaMCMC")`: ...

as.matrix `signature(x = "codaMCMC")`: ...

as.mcmc.list `signature(x = "codaMCMC")`: ...

autocorr.diag `signature(mcmc.obj = "codaMCMC")`: ...

chanames `signature(x = "codaMCMC")`: ...

chisq.diag `signature(x = "codaMCMC")`: ...

coef `signature(object = "codaMCMC")`: ...

coerce `signature(from = "codaMCMC", to = "dcmle")`: ...

coerce `signature(from = "codaMCMC", to = "MCMClist")`: ...

coerce `signature(from = "dcmle", to = "codaMCMC")`: ...

coerce `signature(from = "MCMClist", to = "codaMCMC")`: ...

confint `signature(object = "codaMCMC")`: ...

crosscorr.plot `signature(x = "codaMCMC")`: ...

crosscorr `signature(x = "codaMCMC")`: ...

cumuplot `signature(x = "codaMCMC")`: ...

dcdiag `signature(x = "codaMCMC")`: ...

dcsd `signature(object = "codaMCMC")`: ...

dctable `signature(x = "codaMCMC")`: ...

densityplot `signature(x = "codaMCMC")`: ...

densplot `signature(x = "codaMCMC")`: ...

end `signature(x = "codaMCMC")`: ...

frequency `signature(x = "codaMCMC")`: ...

gelman.diag `signature(x = "codaMCMC")`: ...

gelman.plot `signature(x = "codaMCMC")`: ...

geweke.diag `signature(x = "codaMCMC")`: ...

head `signature(x = "codaMCMC")`: ...

heidel.diag `signature(x = "codaMCMC")`: ...

lambdamax.diag `signature(x = "codaMCMC")`: ...

mcpar `signature(x = "codaMCMC")`: ...

nchain `signature(x = "codaMCMC")`: ...

nclones `signature(x = "codaMCMC")`: ...

niter `signature(x = "codaMCMC")`: ...

nvar `signature(x = "codaMCMC")`: ...

pairs `signature(x = "codaMCMC")`: ...

plot `signature(x = "codaMCMC", y = "missing")`: ...

qqmath `signature(x = "codaMCMC")`: ...

quantile `signature(x = "codaMCMC")`: ...

raftery.diag `signature(x = "codaMCMC")`: ...

show `signature(object = "codaMCMC")`: ...

stack `signature(x = "codaMCMC")`: ...

start `signature(x = "codaMCMC")`: ...

summary `signature(object = "codaMCMC")`: ...

tail `signature(x = "codaMCMC")`: ...

thin `signature(x = "codaMCMC")`: ...

time `signature(x = "codaMCMC")`: ...

traceplot `signature(x = "codaMCMC")`: ...

varnames `signature(x = "codaMCMC")`: ...

vcov `signature(object = "codaMCMC")`: ...

window `signature(x = "codaMCMC")`: ...

xyplot `signature(x = "codaMCMC")`: ...

Author(s)

Peter Solymos

See Also

`mcmc.list`

Examples

```
showClass("codaMCMC")
```

`crosscorr.plot` *Generic after similar coda function*

Description

Generic after similar coda function

Usage

```
crosscorr.plot(x, ...)
```

Arguments

`x`	MCMC object.
`...`	Other arguments.

Value

See corresponding help page

Author(s)

Peter Solymos

See Also

`crosscorr.plot`

`cumuplot` *Generic after similar coda function*

Description

Generic after similar coda function

Usage

```
cumuplot(x, ...)
```

Arguments

`x`	MCMC object.
`...`	Other arguments.

Value

See corresponding help page

Author(s)

Peter Solymos

See Also

`cumuplot`

`custommodel-class` *Class* `"custommodel"`

Description

Stands for the ’custommodel’ S3 class from **dclone** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"dcModel"`, directly.

Methods

No methods defined with class "custommodel" in the signature.

Author(s)

Peter Solymos

See Also

`custommodel`

Examples

```
showClass("custommodel")
```

`dcArgs-class` *Class* `"dcArgs"`

Description

A class union for `NULL` and `"character"`.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcArgs" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcArgs")
```

`dcCodaMCMC-class` *Class* `"dcCodaMCMC"`

Description

An S4 representation of an mcmc.list object of the **coda** package, with data cloning attributes from **dclone** package (the mcmc.list.dc class).

Objects from the Class

Objects can be created by calls of the form `new("dcCodaMCMC", ...)`.

Slots

`dctable`**:** Object of class `"dcTable"`, data cloning based iterative posterior statistics based on `dctable`.

`dcdiag`**:** Object of class `"dcDiag"`, data cloning convergence diagnostics based on `dcdiag`.

`nclones`**:** Object of class `"nClones"`, number of clones.

`values`**:** Object of class `"numeric"`, same as in `"codaMCMC"` class.

`varnames`**:** Object of class `"character"`, same as in `"codaMCMC"` class.

`start`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

`end`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

`thin`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

`nchains`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

`niter`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

`nvar`**:** Object of class `"integer"`, same as in `"codaMCMC"` class.

Extends

Class `"codaMCMC"`, directly.

Methods

[`signature(x = "dcCodaMCMC")`: ...

[[`signature(x = "dcCodaMCMC")`: ...

coerce `signature(from = "dcCodaMCMC", to = "dcmle")`: ...

coerce `signature(from = "dcCodaMCMC", to = "MCMClist")`: ...

coerce `signature(from = "dcmle", to = "dcCodaMCMC")`: ...

coerce `signature(from = "MCMClist", to = "dcCodaMCMC")`: ...

confint `signature(object = "dcCodaMCMC")`: ...

dcdiag `signature(x = "dcCodaMCMC")`: ...

dctable `signature(x = "dcCodaMCMC")`: ...

nclones `signature(x = "dcCodaMCMC")`: ...

str `signature(object = "dcCodaMCMC")`: ...

summary `signature(object = "dcCodaMCMC")`: ...

Author(s)

Peter Solymos

See Also

`jags.fit`

Examples

```
showClass("dcCodaMCMC")
```

`dcDiag-class` *Class* `"dcDiag"`

Description

Virtual class for data cloning convergence diagnostics.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcDiag" in the signature.

Author(s)

Peter Solymos

See Also

`dcdiag`

Examples

```
showClass("dcDiag")
```

`dcdiag-class` *Class* `"dcdiag"`

Description

Stands for the ’dcdiag’ S3 class from **dclone** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"dcDiag"`, directly.

Methods

No methods defined with class "dcdiag" in the signature.

Author(s)

Peter Solymos

See Also

`dcdiag`

Examples

```
showClass("dcdiag")
```

`dcFit-class` *Class* `"dcFit"`

Description

Compendium for data cloning

Objects from the Class

Objects can be created by calls of the form `new("dcFit", ...)`.

Slots

`multiply`: Object of class `"dcArgs"`, same as corresponding `dc.fit` argument.

`unchanged`: Object of class `"dcArgs"`, same as corresponding `dc.fit` argument.

`update`: Object of class `"dcArgs"`, same as corresponding `dc.fit` argument.

`updatefun`: Object of class `"dcFunction"`, same as corresponding `dc.fit` argument.

`initsfun`: Object of class `"dcFunction"`, same as corresponding `dc.fit` argument.

`flavour`: Object of class `"character"`, same as corresponding `dc.fit` argument, default is `"jags"`. It can also be `"winbugs"`, `"openbugs"`, or `"brugs"` referring to the `argument` of `bugs.fit`, in which case `flavour` will be treated as `"bugs"`.

`data`: Object of class `"list"`, same as corresponding `dc.fit` argument.

`model`: Object of class `"dcModel"`, same as corresponding `dc.fit` argument.

`params`: Object of class `"dcParams"`, same as corresponding `dc.fit` argument.

`inits`: Object of class `"dcInits"`, same as corresponding `dc.fit` argument.

Extends

Class `"gsFit"`, directly.

Methods

show `signature(object = "dcFit")`: ...

Author(s)

Peter Solymos

See Also

`dc.fit`, `makeDcFit`

Examples

```
showClass("dcFit")
```

`dcFunction-class` *Class* `"dcFunction"`

Description

Virtual class for BUGS/JAGS models defined as functions.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcFunction" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcFunction")
```

`dcInits-class` *Class* `"dcInits"`

Description

Virtual class for initial values.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcInits" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcInits")
```

`dcmle` *Asymptotic maximum likelihood estimation with data cloning*

Description

This function is a wrapper to fit the model to the data and obtain MLE point estimates and asymptotic standard errors based on the estimate of the Fisher information matrix (theory given by Lele et al. 2007, 2010, software implementation is given in Solymos 2010).

Usage

```
dcmle(x, params, n.clones = 1, cl = NULL, nobs, ...)
```

Arguments

`x`	an object of class "`gsFit`" or "`dcFit`".
`params`	character, vector of model parameters to monitor.
`n.clones`	integer, vector for the number of clones used in fitting.
`cl`	cluster object (snow type cluster) or number of cores (multicore type forking), optional.
`nobs`	number of observations, optional.
`...`	other arguments passed to underlying functions (see Details).

Details

The function uses slots of the input object and passes them as arguments to underlying functions (`jags.fit`, `jags.parfit`, `bugs.fit`, `bugs.parfit`, `stan.fit`, `stan.parfit` `dc.fit`, `dc.parfit`).

Value

An object of class "`dcmle`".

Author(s)

Peter Solymos

References

Solymos, P., 2010. dclone: Data Cloning in R. *The R Journal* **2(2)**, 29–37. URL: `https://journal.r-project.org/archive/2010-2/RJournal_2010-2_Solymos.pdf`

Lele, S.R., B. Dennis and F. Lutscher, 2007. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. *Ecology Letters* **10**, 551–563.

Lele, S. R., K. Nadeem and B. Schmuland, 2010. Estimability and likelihood inference for generalized linear mixed models using data cloning. *Journal of the American Statistical Association* **105**, 1617–1625.

See Also

For additional arguments: `jags.fit`, `jags.parfit`, `bugs.fit`, `bugs.parfit`, `stan.fit`, `stan.parfit` `dc.fit`, `dc.parfit`.

Object classes: "`dcmle`"

Creator functions `makeGsFit` and `makeDcFit`

Examples

```
## Data and model taken from Ponciano et al. 2009
## Ecology 90, 356-362.
paurelia <- c(17,29,39,63,185,258,267,392,510,
    570,650,560,575,650,550,480,520,500)
paramecium <- new("dcFit")
paramecium@data <- list(
    ncl=1,
    n=length(paurelia),
    Y=dcdim(data.matrix(paurelia)))
paramecium@model <- function() {
    for (k in 1:ncl) {
        for(i in 2:(n+1)){
            Y[(i-1), k] ~ dpois(exp(X[i, k])) # observations
            X[i, k] ~ dnorm(mu[i, k], 1 / sigma^2) # state
            mu[i, k] <- X[(i-1), k] + log(lambda) - log(1 + beta * exp(X[(i-1), k]))
        }
        X[1, k] ~ dnorm(mu0, 1 / sigma^2) # state at t0
    }
    beta ~ dlnorm(-1, 1) # Priors on model parameters
    sigma ~ dlnorm(0, 1)
    tmp ~ dlnorm(0, 1)
    lambda <- tmp + 1
    mu0 <- log(2)  + log(lambda) - log(1 + beta * 2)
}
paramecium@multiply <- "ncl"
paramecium@unchanged <- "n"
paramecium@params <- c("lambda","beta","sigma")
## Not run:
(m1 <- dcmle(paramecium, n.clones=1, n.iter=1000))
(m2 <- dcmle(paramecium, n.clones=2, n.iter=1000))
(m3 <- dcmle(paramecium, n.clones=1:3, n.iter=1000))
cl <- makePSOCKcluster(3)
(m4 <- dcmle(paramecium, n.clones=2, n.iter=1000, cl=cl))
(m5 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=cl))
(m6 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=cl,
    partype="parchains"))
(m7 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=cl,
    partype="both"))
stopCluster(cl)

## End(Not run)
```

`dcmle-class` *Class* `"dcmle"`

Description

Fitted model object from `dcmle`.

Objects from the Class

Objects can be created by calls of the form `new("dcmle", ...)`.

Slots

`call`**:** Object of class `"language"`, the call.

`coef`**:** Object of class `"numeric"`, coefficients (posterior means).

`fullcoef`**:** Object of class `"numeric"`, full coefficients, possibly with fixed values.

`vcov`**:** Object of class `"matrix"`, variance covariance matrix.

`details`**:** Object of class `"dcCodaMCMC"`, the fitted model object.

`nobs`**:** Object of class `"integer"`, number of observations, optional.

`method`**:** Object of class `"character"`.

Methods

[`signature(x = "dcmle")`: ...

[[`signature(x = "dcmle")`: ...

acfplot `signature(x = "dcmle")`: ...

as.array `signature(x = "dcmle")`: ...

as.matrix `signature(x = "dcmle")`: ...

as.mcmc.list `signature(x = "dcmle")`: ...

autocorr.diag `signature(mcmc.obj = "dcmle")`: ...

chanames `signature(x = "dcmle")`: ...

chisq.diag `signature(x = "dcmle")`: ...

coef `signature(object = "dcmle")`: ...

coerce `signature(from = "codaMCMC", to = "dcmle")`: ...

coerce `signature(from = "dcCodaMCMC", to = "dcmle")`: ...

coerce `signature(from = "dcmle", to = "codaMCMC")`: ...

coerce `signature(from = "dcmle", to = "dcCodaMCMC")`: ...

coerce `signature(from = "dcmle", to = "MCMClist")`: ...

coerce `signature(from = "MCMClist", to = "dcmle")`: ...

confint `signature(object = "dcmle")`: ...

crosscorr.plot `signature(x = "dcmle")`: ...

crosscorr `signature(x = "dcmle")`: ...

cumuplot `signature(x = "dcmle")`: ...

dcdiag `signature(x = "dcmle")`: ...

dcsd `signature(object = "dcmle")`: ...

dctable `signature(x = "dcmle")`: ...

densityplot `signature(x = "dcmle")`: ...

densplot `signature(x = "dcmle")`: ...

end `signature(x = "dcmle")`: ...

frequency `signature(x = "dcmle")`: ...

gelman.diag `signature(x = "dcmle")`: ...

gelman.plot `signature(x = "dcmle")`: ...

geweke.diag `signature(x = "dcmle")`: ...

head `signature(x = "dcmle")`: ...

heidel.diag `signature(x = "dcmle")`: ...

lambdamax.diag `signature(x = "dcmle")`: ...

mcpar `signature(x = "dcmle")`: ...

nchain `signature(x = "dcmle")`: ...

nclones `signature(x = "dcmle")`: ...

niter `signature(x = "dcmle")`: ...

nvar `signature(x = "dcmle")`: ...

pairs `signature(x = "dcmle")`: ...

plot `signature(x = "dcmle", y = "missing")`: ...

qqmath `signature(x = "dcmle")`: ...

quantile `signature(x = "dcmle")`: ...

raftery.diag `signature(x = "dcmle")`: ...

show `signature(object = "dcmle")`: ...

stack `signature(x = "dcmle")`: ...

start `signature(x = "dcmle")`: ...

str `signature(object = "dcmle")`: ...

summary `signature(object = "dcmle")`: ...

tail `signature(x = "dcmle")`: ...

thin `signature(x = "dcmle")`: ...

time `signature(x = "dcmle")`: ...

traceplot `signature(x = "dcmle")`: ...

update `signature(object = "dcmle")`: ...

varnames `signature(x = "dcmle")`: ...

vcov `signature(object = "dcmle")`: ...

window `signature(x = "dcmle")`: ...

xyplot `signature(x = "dcmle")`: ...

Author(s)

Peter Solymos

See Also

`dcmle`

Examples

```
showClass("dcmle")
```

`dcModel-class` *Class* `"dcModel"`

Description

Virtual class for BUGS/JAGS models.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcModel" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcModel")
```

`dcParams-class` *Class* `"dcParams"`

Description

Virtual class for model parameters to monitor.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcParams" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcParams")
```

`dcTable-class` *Class* `"dcTable"`

Description

Posterior statistics from iterative fit, virtual class.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "dcTable" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("dcTable")
```

`dctable-class` *Class* `"dctable"`

Description

Stands for the 'dctable' S3 class from **dclone** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"dcTable"`, directly.

Methods

No methods defined with class "dctable" in the signature.

Author(s)

Peter Solymos

See Also

`dctable`

Examples

```
showClass("dctable")
```

`diagnostics` *Diagnostic functions set as generic*

Description

Diagnostic functions set as generic.

Usage

```
gelman.diag(x, ...)
geweke.diag(x, ...)
heidel.diag(x, ...)
raftery.diag(x, ...)

gelman.plot(x, ...)
```

Arguments

`x`	MCMC objects.
`...`	Other arguments.

Details

Diagnostic functions from **coda** package are defined as generics for extensibility.

Value

Diagnostics summaries, and plot.

Author(s)

Peter Solymos

References

See relevant help pages.

See Also

`gelman.diag` `geweke.diag` `heidel.diag` `raftery.diag`

`gelman.plot`

`gsFit-class` *Class* `"gsFit"`

Description

BUGS/JAGS compendium

Objects from the Class

Objects can be created by calls of the form `new("gsFit", ...)`.

Slots

`data`**:** Object of class `"list"`, same as corresponding `jags.fit` `bugs.fit` or argument.

`model`**:** Object of class `"dcModel"`, same as corresponding `jags.fit` `bugs.fit` or argument.

`params`**:** Object of class `"dcParams"`, same as corresponding `jags.fit` `bugs.fit` or argument.

`inits`**:** Object of class `"dcInits"`, same as corresponding `jags.fit` `bugs.fit` or argument.

`flavour`**:** Object of class `"character"`, same as corresponding `dc.fit` argument, default is `"jags"`. It can also be `"winbugs"`, `"openbugs"`, or `"brugs"` referring to the `argument` of `bugs.fit`, in which case `flavour` will be treated as `"bugs"`.

Methods

show `signature(object = "gsFit")`: ...

Author(s)

Peter Solymos

See Also

`jags.fit`, `bugs.fit`, `makeGsFit`

Examples

```
showClass("gsFit")
```

`makeDcFit` *Data object creators*

Description

Creator functions for data types used in the **dcmle** package.

Usage

```
makeGsFit(data, model, params = NULL, inits = NULL, flavour)

makeDcFit(data, model, params=NULL, inits = NULL,
    multiply = NULL, unchanged = NULL, update = NULL,
    updatefun = NULL, initsfun = NULL, flavour)
```

Arguments

`data`	usually a named list with data.
`model`	BUGS model (function, character vector or a `custommodel` object). The argument is coerced into a `custommodel` object.
`params`	optional, character vector for model parameters to monitor.
`inits`	initial values (`NULL`, list or function).
`multiply`	optional, argument passed to `dc.fit`.
`unchanged`	optional, argument passed to `dc.fit`.
`update`	optional, argument passed to `dc.fit`.
`updatefun`	optional, argument passed to `dc.fit`.
`initsfun`	optional, argument passed to `dc.fit`.
`flavour`	optional, argument passed to `dc.fit`.

Details

'gsFit' (after BU*GS*/JA*GS*) is a basic object class representing requirements for the Bayesian MCMC model fitting. The 'dcFit' object class extends 'gsFit' by additional slots that are used to fine tune how data cloning is done during fitting process. Both 'gsFit' and 'dcFit' represent prerequisites for model fitting, but do not containing any fitted parts. Creator functions `makeGsFit` and `makeDcFit` are available for these classes. See `dcmle-package` help page for usage of creator functions.

The default `flavour` is stored in `getOption("dcmle.flavour")` with value `"jags"`. It can be changed as `options("dcmle.flavour"="bugs")` if required.

Value

`makeGsFit` returns a 'gsFit' object (gsFit-class).

`makeDcFit` returns a 'dcFit' object (dcFit-class).

Author(s)

Peter Solymos

See Also

gsFit-class, dcFit-class, `dcmle`

Examples

```
showClass("gsFit")
new("gsFit")
showClass("dcFit")
new("dcFit")
```

`mcmc-class` *Class* `"mcmc"`

Description

Stands for the 'mcmc' S3 class from **coda** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class "`MCMClist`", directly.

Methods

No methods defined with class "mcmc" in the signature.

Author(s)

Peter Solymos

See Also

`mcmc`

Examples

```
showClass("mcmc")
```

`mcmc.list-class` *Class* `"mcmc.list"`

Description

Stands for the 'mcmc.list' S3 class from **coda** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"MCMClist"`, directly.

Methods

No methods defined with class "mcmc.list" in the signature.

Author(s)

Peter Solymos

See Also

`mcmc.list`

Examples

```
showClass("mcmc.list")
```

`mcmc.list.dc-class` *Class* `"mcmc.list.dc"`

Description

Stands for the 'mcmc.list.dc' S3 class from **dclone** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Extends

Class `"MCMClist"`, directly.

Methods

No methods defined with class "mcmc.list.dc" in the signature.

Author(s)

Peter Solymos

See Also

`mcmc.list`, `jags.fit`

Examples

```
showClass("mcmc.list.dc")
```

`MCMClist-class` *Class* `"MCMClist"`

Description

Virtual class for S3 mcmc.list object from **coda** package.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

acfplot `signature(x = "MCMClist")`: ...

autocorr.diag `signature(mcmc.obj = "MCMClist")`: ...

chanames `signature(x = "MCMClist")`: ...

chisq.diag `signature(x = "MCMClist")`: ...

coerce `signature(from = "codaMCMC", to = "MCMClist")`: ...

coerce `signature(from = "dcCodaMCMC", to = "MCMClist")`: ...

coerce `signature(from = "dcmle", to = "MCMClist")`: ...

coerce `signature(from = "MCMClist", to = "codaMCMC")`: ...

coerce `signature(from = "MCMClist", to = "dcCodaMCMC")`: ...

coerce `signature(from = "MCMClist", to = "dcmle")`: ...

confint `signature(object = "MCMClist")`: ...

crosscorr.plot `signature(x = "MCMClist")`: ...

crosscorr `signature(x = "MCMClist")`: ...

cumuplot `signature(x = "MCMClist")`: ...

densityplot `signature(x = "MCMClist")`: ...

densplot `signature(x = "MCMClist")`: ...

frequency `signature(x = "MCMClist")`: ...

gelman.diag `signature(x = "MCMClist")`: ...

gelman.plot `signature(x = "MCMClist")`: ...

geweke.diag `signature(x = "MCMClist")`: ...

heidel.diag `signature(x = "MCMClist")`: ...

lambdamax.diag `signature(x = "MCMClist")`: ...

mcpar `signature(x = "MCMClist")`: ...

nchain `signature(x = "MCMClist")`: ...

niter `signature(x = "MCMClist")`: ...

nvar `signature(x = "MCMClist")`: ...

pairs `signature(x = "MCMClist")`: ...

plot `signature(x = "MCMClist", y = "missing")`: ...

qqmath `signature(x = "MCMClist")`: ...

quantile `signature(x = "MCMClist")`: ...

raftery.diag `signature(x = "MCMClist")`: ...

thin `signature(x = "MCMClist")`: ...

traceplot `signature(x = "MCMClist")`: ...

varnames `signature(x = "MCMClist")`: ...

xyplot `signature(x = "MCMClist")`: ...

Author(s)

Peter Solymos

See Also

`mcmc.list`

Examples

```
showClass("MCMClist")
```

`nClones-class` *Class* `"nClones"`

Description

Number of clones, virtual class.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "nClones" in the signature.

Author(s)

Peter Solymos

Examples

```
showClass("nClones")
```

`summary.codaMCMC-class`

Class `"summary.codaMCMC"`

Description

Summary object.

Objects from the Class

Objects can be created by calls of the form `new("summary.codaMCMC", ...)`.

Slots

`settings`**:** Object of class `"integer"`, MCMC settings.

`coef`**:** Object of class `"matrix"`, posterior statistics.

Methods

show `signature(object = "summary.codaMCMC")`: ...

Author(s)

Peter Solymos

See Also

`mcmc.list`.

Examples

```
showClass("summary.codaMCMC")
```

`summary.dcCodaMCMC-class`

Class `"summary.dcCodaMCMC"`

Description

Summary object.

Objects from the Class

Objects can be created by calls of the form `new("summary.dcCodaMCMC", ...)`.

Slots

`settings`: Object of class `"integer"`, MCMC settings.

`coef`: Object of class `"matrix"`, coefficients (posterior means).

`convergence`: Object of class `"dcDiag"`, data cloning convergence diagnostics.

Extends

Class `"summary.codaMCMC"`, directly.

Methods

show `signature(object = "summary.dcCodaMCMC")`: ...

Author(s)

Peter Solymos

See Also

`jags.fit`, `dcdiag`

Examples

```
showClass("summary.dcCodaMCMC")
```

`summary.dcmle-class` *Class* `"summary.dcmle"`

Description

Summary object.

Objects from the Class

Objects can be created by calls of the form `new("summary.dcmle", ...)`.

Slots

`title`: Object of class `"character"`, title to print, optional.

`call`: Object of class `"language"`, the call.

`settings`: Object of class `"integer"`, MCMC settings.

`coef`: Object of class `"matrix"`, coefficients (posterior means).

`convergence`: Object of class `"dcDiag"`, data cloning convergence diagnostics.

Extends

Class "`summary.dcCodaMCMC`", directly. Class "`summary.codaMCMC`", by class "summary.dcCodaMCMC", distance 2.

Methods

show `signature(object = "summary.dcmle")`: ...

Author(s)

Peter Solymos

See Also

`jags.fit`, `dcdiag`, `dcmle`

Examples

```
showClass("summary.dcmle")
```

Index