

Package: dcmle (via r-universe)

August 19, 2024

Type Package

Title Hierarchical Models Made Easy with Data Cloning

Version 0.4-1

Date 2023-07-04

Author Peter Solymos [cre, aut]

(<https://orcid.org/0000-0001-7337-1740>)

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

Repository <https://datacloning.r-universe.dev>

RemoteUrl <https://github.com/datacloning/dcmle>

RemoteRef HEAD

RemoteSha b639b2c9f3df6ca5d1607304a8d7370e654cb8c2

Contents

dcmle-package	2
chanames	4
codaMCMC-class	5

crosscorr.plot	7
cumuplot	8
custommodel-class	8
dcArgs-class	9
dcCodaMCMC-class	10
dcDiag-class	11
dcdiag-class	12
dcFit-class	12
dcFunction-class	13
dcInits-class	14
dcmle	15
dcmle-class	17
dcModel-class	19
dcParams-class	20
dcTable-class	20
dctable-class	21
diagnostics	21
gsFit-class	22
makeDcFit	23
mcmc-class	24
mcmc.list-class	25
mcmc.list.dc-class	26
MCMClist-class	26
nClones-class	28
summary.codaMCMC-class	29
summary.dcCodaMCMC-class	29
summary.dcmle-class	30
Index	32

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>

ReferencesForum: <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
bevholFit <-
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 'bevholMle' extends the 'dcmle' class
```

```

## it can have additional slots
setClass("bevholtMle",
  representation(y="numeric", title="character"),
  contains = "dcmlc")
## Function to fit the Beverton-Holt model to data
bevholt <- function(y, n.clones, ...) {
  new("bevholtMle",
    dcmlc(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, "dcmlc"), 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"): ...
lamdamax.diag signature(x = "codaMCMC"): ...
mcpair 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

<code>x</code>	MCMC object.
<code>...</code>	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	<i>Class "dcArgs"</i>
--------------	-----------------------

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))
c1 <- makePSOCKcluster(3)
(m4 <- dcmle(paramecium, n.clones=2, n.iter=1000, cl=c1))
(m5 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=c1))
(m6 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=c1,
             partype="parchains"))
(m7 <- dcmle(paramecium, n.clones=1:3, n.iter=1000, cl=c1,
             partype="both"))
stopCluster(c1)

## End(Not run)
```

dcmle-class	<i>Class "dcmle"</i>
-------------	----------------------

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"): ...
dcspd 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"): ...
mcpair 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	<i>Class "dcModel"</i>
---------------	------------------------

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	<i>Diagnostic functions set as generic</i>
-------------	--

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](#) [heidell.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 **dcmlc** 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")
```

<code>mcmc.list-class</code>	<i>Class "mcmc.list"</i>
------------------------------	--------------------------

Description

Stands for the 'mcmc.list' S3 class from **cod**a 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"): ...
mcpair 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

* classes

- [codaMCMC-class](#), 5
- [custommodel-class](#), 8
- [dcArgs-class](#), 9
- [dcCodaMCMC-class](#), 10
- [dcDiag-class](#), 11
- [dcdiag-class](#), 12
- [dcFit-class](#), 12
- [dcFunction-class](#), 13
- [dcInits-class](#), 14
- [dcmle-class](#), 17
- [dcModel-class](#), 19
- [dcParams-class](#), 20
- [dcTable-class](#), 20
- [dctable-class](#), 21
- [gsFit-class](#), 22
- [makeDcFit](#), 23
- [mcmc-class](#), 24
- [mcmc.list-class](#), 25
- [mcmc.list.dc-class](#), 26
- [MCMClist-class](#), 26
- [nClones-class](#), 28
- [summary.codaMCMC-class](#), 29
- [summary.dcCodaMCMC-class](#), 29
- [summary.dcmle-class](#), 30

* htest

- [dcmle](#), 15

* models

- [dcmle](#), 15

* package

- [dcmle-package](#), 2

* utils

- [chanames](#), 4
- [crosscorr.plot](#), 7
- [cumuplot](#), 8
- [diagnostics](#), 21
- [makeDcFit](#), 23

- [\[, codaMCMC, ANY, ANY, ANY-method \(dcCodaMCMC-class\)](#), 10

- [\[, codaMCMC-method \(codaMCMC-class\)](#), 5
- [\[, dcCodaMCMC, ANY, ANY, ANY-method \(dcCodaMCMC-class\)](#), 10
- [\[, dcCodaMCMC-method \(dcCodaMCMC-class\)](#), 10
- [\[, dcmle, ANY, ANY, ANY-method \(dcCodaMCMC-class\)](#), 10
- [\[, dcmle-method \(dcmle-class\)](#), 17
- [\[\[, codaMCMC-method \(codaMCMC-class\)](#), 5
- [\[\[, dcCodaMCMC-method \(dcCodaMCMC-class\)](#), 10
- [\[\[, dcmle-method \(dcmle-class\)](#), 17

- [acfplot, codaMCMC-method \(codaMCMC-class\)](#), 5
- [acfplot, dcmle-method \(dcmle-class\)](#), 17
- [acfplot, MCMClist-method \(MCMClist-class\)](#), 26
- [as.array, codaMCMC-method \(codaMCMC-class\)](#), 5
- [as.array, dcmle-method \(dcmle-class\)](#), 17
- [as.matrix, codaMCMC-method \(codaMCMC-class\)](#), 5
- [as.matrix, dcmle-method \(dcmle-class\)](#), 17
- [as.mcmc.list, codaMCMC-method \(codaMCMC-class\)](#), 5
- [as.mcmc.list, dcmle-method \(dcmle-class\)](#), 17
- [autocorr.diag, codaMCMC-method \(codaMCMC-class\)](#), 5
- [autocorr.diag, dcmle-method \(dcmle-class\)](#), 17
- [autocorr.diag, MCMClist-method \(MCMClist-class\)](#), 26

- [bugs.fit](#), 13, 15, 16, 22, 23
- [bugs.parfit](#), 15, 16

- [chanames](#), 4, 4

- chanames, codaMCMC-method
(codaMCMC-class), 5
- chanames, dcmle-method (dcmle-class), 17
- chanames, MCMClist-method
(MCMClist-class), 26
- chisq.diag, codaMCMC-method
(codaMCMC-class), 5
- chisq.diag, dcmle-method (dcmle-class),
17
- chisq.diag, MCMClist-method
(MCMClist-class), 26
- codaMCMC, 3, 10
- codaMCMC-class, 5
- coef, codaMCMC-method (codaMCMC-class), 5
- coef, dcmle-method (dcmle-class), 17
- coerce, codaMCMC, dcmle-method
(codaMCMC-class), 5
- coerce, codaMCMC, MCMClist-method
(codaMCMC-class), 5
- coerce, dcCodaMCMC, dcmle-method
(dcCodaMCMC-class), 10
- coerce, dcCodaMCMC, MCMClist-method
(dcCodaMCMC-class), 10
- coerce, dcmle, codaMCMC-method
(dcmle-class), 17
- coerce, dcmle, dcCodaMCMC-method
(dcmle-class), 17
- coerce, dcmle, MCMClist-method
(dcmle-class), 17
- coerce, MCMClist, codaMCMC-method
(codaMCMC-class), 5
- coerce, MCMClist, dcCodaMCMC-method
(dcCodaMCMC-class), 10
- coerce, MCMClist, dcmle-method
(dcmle-class), 17
- confint, codaMCMC-method
(codaMCMC-class), 5
- confint, dcCodaMCMC-method
(dcCodaMCMC-class), 10
- confint, dcmle-method (dcmle-class), 17
- confint, MCMClist-method
(MCMClist-class), 26
- crosscorr, codaMCMC-method
(codaMCMC-class), 5
- crosscorr, dcmle-method (dcmle-class), 17
- crosscorr, MCMClist-method
(MCMClist-class), 26
- crosscorr.plot, 7, 7
- crosscorr.plot, codaMCMC-method
(codaMCMC-class), 5
- crosscorr.plot, dcmle-method
(dcmle-class), 17
- crosscorr.plot, MCMClist-method
(MCMClist-class), 26
- cumuplot, 8, 8
- cumuplot, codaMCMC-method
(codaMCMC-class), 5
- cumuplot, dcmle-method (dcmle-class), 17
- cumuplot, MCMClist-method
(MCMClist-class), 26
- custommodel, 9, 23
- custommodel-class, 8
- dc.fit, 13, 15, 16, 22, 23
- dc.parfit, 15, 16
- dcArgs-class, 9
- dcCodaMCMC, 3
- dcCodaMCMC-class, 10
- dcDiag, 12
- dcdiag, 10–12, 30, 31
- dcdiag, codaMCMC-method
(codaMCMC-class), 5
- dcdiag, dcCodaMCMC-method
(dcCodaMCMC-class), 10
- dcdiag, dcmle-method (dcmle-class), 17
- dcDiag-class, 11
- dcdiag-class, 12
- dcFit, 2, 15
- dcFit-class, 12, 24
- dcFunction-class, 13
- dcInits-class, 14
- dcmle, 2, 3, 15, 15, 16, 17, 19, 24, 31
- dcmle-class, 17
- dcmle-package, 2
- dcModel, 8
- dcModel-class, 19
- dcParams-class, 20
- dcsd, codaMCMC-method (codaMCMC-class), 5
- dcsd, dcmle-method (dcmle-class), 17
- dcTable, 21
- dctable, 10, 21
- dctable, codaMCMC-method
(codaMCMC-class), 5
- dctable, dcCodaMCMC-method
(dcCodaMCMC-class), 10
- dctable, dcmle-method (dcmle-class), 17
- dcTable-class, 20

- dctable-class, [21](#)
- densityplot, codaMCMC-method (codaMCMC-class), [5](#)
- densityplot, dcmle-method (dcmle-class), [17](#)
- densityplot, MCMClist-method (MCMClist-class), [26](#)
- densplot, codaMCMC-method (codaMCMC-class), [5](#)
- densplot, dcmle-method (dcmle-class), [17](#)
- densplot, MCMClist-method (MCMClist-class), [26](#)
- diagnostics, [21](#)
- end, codaMCMC-method (codaMCMC-class), [5](#)
- end, dcmle-method (dcmle-class), [17](#)
- frequency, codaMCMC-method (codaMCMC-class), [5](#)
- frequency, dcmle-method (dcmle-class), [17](#)
- frequency, MCMClist-method (MCMClist-class), [26](#)
- gelman.diag, [22](#)
- gelman.diag (diagnostics), [21](#)
- gelman.diag, codaMCMC-method (codaMCMC-class), [5](#)
- gelman.diag, dcmle-method (dcmle-class), [17](#)
- gelman.diag, MCMClist-method (MCMClist-class), [26](#)
- gelman.plot, [22](#)
- gelman.plot (diagnostics), [21](#)
- gelman.plot, codaMCMC-method (codaMCMC-class), [5](#)
- gelman.plot, dcmle-method (dcmle-class), [17](#)
- gelman.plot, MCMClist-method (MCMClist-class), [26](#)
- geweke.diag, [22](#)
- geweke.diag (diagnostics), [21](#)
- geweke.diag, codaMCMC-method (codaMCMC-class), [5](#)
- geweke.diag, dcmle-method (dcmle-class), [17](#)
- geweke.diag, MCMClist-method (MCMClist-class), [26](#)
- gsFit, [2](#), [13](#), [15](#)
- gsFit-class, [22](#), [24](#)
- head, codaMCMC-method (codaMCMC-class), [5](#)
- head, dcmle-method (dcmle-class), [17](#)
- heidel.diag, [22](#)
- heidel.diag (diagnostics), [21](#)
- heidel.diag, codaMCMC-method (codaMCMC-class), [5](#)
- heidel.diag, dcmle-method (dcmle-class), [17](#)
- heidel.diag, MCMClist-method (MCMClist-class), [26](#)
- jags.fit, [11](#), [15](#), [16](#), [22](#), [23](#), [26](#), [30](#), [31](#)
- jags.parfit, [15](#), [16](#)
- lambdamax.diag, codaMCMC-method (codaMCMC-class), [5](#)
- lambdamax.diag, dcmle-method (dcmle-class), [17](#)
- lambdamax.diag, MCMClist-method (MCMClist-class), [26](#)
- makeDcFit, [3](#), [13](#), [16](#), [23](#)
- makeGsFit, [3](#), [16](#), [23](#)
- makeGsFit (makeDcFit), [23](#)
- mcmc, [25](#)
- mcmc-class, [24](#)
- mcmc.list, [7](#), [25](#), [26](#), [28](#), [29](#)
- mcmc.list-class, [25](#)
- mcmc.list.dc-class, [26](#)
- MCMClist, [24–26](#)
- MCMClist-class, [26](#)
- mcpair, codaMCMC-method (codaMCMC-class), [5](#)
- mcpair, dcmle-method (dcmle-class), [17](#)
- mcpair, MCMClist-method (MCMClist-class), [26](#)
- nchain, codaMCMC-method (codaMCMC-class), [5](#)
- nchain, dcmle-method (dcmle-class), [17](#)
- nchain, MCMClist-method (MCMClist-class), [26](#)
- nclones, codaMCMC-method (codaMCMC-class), [5](#)
- nclones, dcCodaMCMC-method (dcCodaMCMC-class), [10](#)
- nclones, dcmle-method (dcmle-class), [17](#)
- nClones-class, [28](#)
- niter, codaMCMC-method (codaMCMC-class), [5](#)

- niter, dcmle-method (dcmle-class), 17
- niter, MCMClist-method (MCMClist-class), 26
- nvar, codaMCMC-method (codaMCMC-class), 5
- nvar, dcmle-method (dcmle-class), 17
- nvar, MCMClist-method (MCMClist-class), 26

- pairs, codaMCMC-method (codaMCMC-class), 5
- pairs, dcmle-method (dcmle-class), 17
- pairs, MCMClist-method (MCMClist-class), 26
- plot, codaMCMC, missing-method (codaMCMC-class), 5
- plot, dcmle, missing-method (dcmle-class), 17
- plot, MCMClist, missing-method (MCMClist-class), 26

- qqmath, codaMCMC-method (codaMCMC-class), 5
- qqmath, dcmle-method (dcmle-class), 17
- qqmath, MCMClist-method (MCMClist-class), 26
- quantile, codaMCMC-method (codaMCMC-class), 5
- quantile, dcmle-method (dcmle-class), 17
- quantile, MCMClist-method (MCMClist-class), 26

- raftery.diag, 22
- raftery.diag (diagnostics), 21
- raftery.diag, codaMCMC-method (codaMCMC-class), 5
- raftery.diag, dcmle-method (dcmle-class), 17
- raftery.diag, MCMClist-method (MCMClist-class), 26

- show, codaMCMC-method (codaMCMC-class), 5
- show, dcFit-method (dcFit-class), 12
- show, dcmle-method (dcmle-class), 17
- show, gsFit-method (gsFit-class), 22
- show, summary.codaMCMC-method (summary.codaMCMC-class), 29
- show, summary.dcCodaMCMC-method (summary.dcCodaMCMC-class), 29
- show, summary.dcmle-method (summary.dcmle-class), 30

- stack, codaMCMC-method (codaMCMC-class), 5
- stack, dcmle-method (dcmle-class), 17
- stan.fit, 15, 16
- stan.parfit, 15, 16
- start, codaMCMC-method (codaMCMC-class), 5
- start, dcmle-method (dcmle-class), 17
- str, dcCodaMCMC-method (dcCodaMCMC-class), 10
- str, dcmle-method (dcmle-class), 17
- summary, codaMCMC-method (codaMCMC-class), 5
- summary, dcCodaMCMC-method (dcCodaMCMC-class), 10
- summary, dcmle-method (dcmle-class), 17
- summary.codaMCMC, 30, 31
- summary.codaMCMC-class, 29
- summary.dcCodaMCMC, 31
- summary.dcCodaMCMC-class, 29
- summary.dcmle-class, 30

- tail, codaMCMC-method (codaMCMC-class), 5
- tail, dcmle-method (dcmle-class), 17
- thin, codaMCMC-method (codaMCMC-class), 5
- thin, dcmle-method (dcmle-class), 17
- thin, MCMClist-method (MCMClist-class), 26

- time, codaMCMC-method (codaMCMC-class), 5
- time, dcmle-method (dcmle-class), 17
- traceplot, codaMCMC-method (codaMCMC-class), 5
- traceplot, dcmle-method (dcmle-class), 17
- traceplot, MCMClist-method (MCMClist-class), 26

- update, dcmle-method (dcmle-class), 17

- varnames, 4
- varnames (chanames), 4
- varnames, codaMCMC-method (codaMCMC-class), 5
- varnames, dcmle-method (dcmle-class), 17
- varnames, MCMClist-method (MCMClist-class), 26
- vcov, codaMCMC-method (codaMCMC-class), 5
- vcov, dcmle-method (dcmle-class), 17

- window, codaMCMC-method (codaMCMC-class), 5

window, dcmle-method (dcmle-class), [17](#)

xyplot, codaMCMC-method
(codaMCMC-class), [5](#)

xyplot, dcmle-method (dcmle-class), [17](#)

xyplot, MCMClist-method
(MCMClist-class), [26](#)