# Package 'mclcar'

January 8, 2022

**Type** Package

**Title** Estimating Conditional Auto-Regressive (CAR) Models using Monte Carlo Likelihood Methods

**Version** 0.2-0

**Date** 2022-01-07

**Author** Zhe Sha [aut, cre]

**Maintainer** Zhe Sha <zhesha1006@gmail.com>

**Description** The likelihood of direct CAR models and Binomial and Poisson GLM with latent CAR variables are approximated by the Monte Carlo likelihood. The Maximum Monte Carlo likelihood estimator is found either by an iterative procedure of directly maximising the Monte Carlo approximation or by a response surface design method.Reference for the method can be found in the DPhil thesis in Z. Sha (2016). For application a good reference is R.Bivand et.al (2017) <doi:10.1016/j.spasta.2017.01.002>.

**License** GPL (>= 2)

**Depends** R (>= 2.10),

**LazyData** yes

**Imports** spam, rsm, fields, maxLik, nleqslv, spdep, spatialreg

**NeedsCompilation** no

**Suggests** knitr

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2022-01-08 14:12:41 UTC

## R topics documented:

---

`mclcar-package` *Estimating Conditional Auto-Regressive (CAR) Models using Monte Carlo Likelihood Methods*

---

## Description

The likelihood of direct CAR models and Binomial and Poisson GLM with latent CAR variables are approximated by the Monte Carlo likelihood. The Maximum Monte Carlo likelihood estimator is found either by an iterative procedure of directly maximising the Monte Carlo approximation or by a response surface design method.Reference for the method can be found in the DPhil thesis in Z. Sha (2016). For application a good reference is R.Bivand et.al (2017) <doi:10.1016/j.spasta.2017.01.002>.

## Details

The DESCRIPTION file:

| | |
|---|---|
| Package: | mclcar |
| Type: | Package |
| Title: | Estimating Conditional Auto-Regressive (CAR) Models using Monte Carlo Likelihood Methods |
| Version: | 0.2-0 |
| Date: | 2022-01-07 |
| Authors@R: | c(person("Zhe", "Sha", role = c("aut", "cre"), email = "zhesha1006@gmail.com")) |
| Author: | Zhe Sha [aut, cre] |
| Maintainer: | Zhe Sha <zhesha1006@gmail.com> |
| Description: | The likelihood of direct CAR models and Binomial and Poisson GLM with latent CAR variables are ap |
| License: | GPL (>= 2) |
| Depends: | R (>= 2.10), |
| LazyData: | yes |
| Imports: | spam, rsm, fields, maxLik, nleqslv, spdep, spatialreg |
| NeedsCompilation: | no |
| Suggests: | knitr |
| VignetteBuilder: | knitr |

Index of help topics:



~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

## Author(s)


Zhe Sha [aut, cre]

Maintainer: Zhe Sha <zhesha1006@gmail.com>


## References


- Box, G. E. P. and Draper, N. R. (2007) *Response Surfaces, Mixtures, and Ridge Analyses*, Wiley, New York
- Sha, Z. 2016 *Estimating conditional auto-regression models*, DPhil Thesis, Oxford.

## See Also

rsm,maxLik

## Examples

```
#### Example 1: iterative optimising the Monte Carlo likelihood
#########################################################################
system.time({
set.seed(33)
n.torus <- 8
nb <- 30
rho <- 0.2
sigma <- 1.5
beta <- c(1, 1)
pars.true <- c(rho, sigma, beta)
X0 <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
mydata2 <- CAR.simGLM(method = "binom", n = c(n.torus, n.torus), pars = pars.true,
                      Xs = as.matrix(X0), n.trial = nb)

## use a glm to find initial values for the importance sampler
data.glm <- data.frame(y=mydata2$y, mydata2$covX[,-1])
fit.glm <- glm(cbind(y, nb-mydata2$y) ~ .,data = data.glm, family=binomial)
library(spatialreg)
library(spdep)
logitp <- log((mydata2$y+0.5)/(mydata2$n.trial - mydata2$y + 0.5))
data.splm <- data.frame(y=logitp, mydata2$covX[,-1])
listW <- mat2listw(mydata2$W)
fit.splm <- spautolm(y~., data = data.splm, listw=listW, family = "CAR")
pars1 <- c(fit.splm$lambda, fit.splm$fit$s2, coef(fit.glm))

 ## Use the iterative procedure to find the MC-MLE
 iter.mcmle <- OptimMCL(data = mydata2, psi0 = pars1, family = "binom",
                        control = list(n.iter = 1, mc.var = TRUE),
                     mc.control = list(N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)), thin = 5,
                                       burns = 5e2, method = "mala", scale.fixed = TRUE))
summary(iter.mcmle, family = "binom", mc.covar=TRUE)

#### Example 2: RSM optimising the Monte Carlo likelihood
#########################################################################
mydata3 <- CAR.simGLM(method = "poisson", n = c(n.torus, n.torus),
                      pars = pars.true, Xs =as.matrix(X0))
## Fit by rsm
rsm.mcmle <- rsmMCL(data = mydata3, psi0 = c(0, 1, 2, 2), family = "poisson",
                    control = list(n.iter = 1, trace.all = TRUE),
                    mc.control = list(N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)),
                                      thin = 5, burns = 5e2,
                                      method = "mala", scale.fixed = TRUE))
summary(rsm.mcmle, family = "poisson",  mc.covar=TRUE)
})
```

CAR.simLM *Simulate samples from a CAR model.*

## Description

This help page documents several functions for simulate samples from different CAR models.

## Usage

```
CAR.simTorus(n1, n2, rho, prec)
CAR.simWmat(rho, prec, W)
CAR.simLM(pars, data)
CAR.simGLM(method = c("binom", "poisson"), W, n = NULL, pars, Xs = NULL,
n.trial = 1)
```

## Arguments

| | |
|---|---|
| n1, n2, n | n = c(n1, n2), the size of the torus |
| rho | the spatial coefficient in the CAR precision matrix (inverse of the covariance matrix) |
| prec | the precision in the CAR precision matrix |
| W | the spatial weight matrix |
| pars | parameter values of the direct CAR model |
| data | a data object same as described in loglik.dCAR |
| method | a character equal to either "binom" or "poisson" that indicate the distribution of the sample to be simulated. |
| Xs | the covariates to be used in the glm |
| n.trial | the number of trials in each unit of the binomial samples |

## Value

| | |
|---|---|
| CAR.simTorus | returns a list containing the spatial weight matrix W and the simulated CAR samples X; |
| CAR.simWmat,CAR.simLM | returns an array of the simulated CAR samples; |
| CAR.simGLM | returns a list containing:<br>**rho, sigma, beta,** the parameter values used to generate the samples<br>**y,** the simulated data<br>**covX,** the covariates<br>**W,** the spatial weight matrix<br>**Z.true,** the simulated CAR variables<br>**eta,** the simulated glm linear response<br>**Emean,** the simulated glm mean<br>**n.trial,** the number of trials in each unit for binomial samples |

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

mcl.glm, mcl.dCAR, mcl.prep.dCAR, mcl.prep.glm

## Examples

```
## Simulate CAR data on a torus
set.seed(33)
n.torus <- 10
rho <- 0.2
sigma <- 1.5
prec <- 1/sigma
beta <- c(1, 1)
XX <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
mydata1 <- CAR.simTorus(n1 = n.torus, n2 = n.torus, rho = rho, prec = prec)

## Simulate CAR data for a given spatial weight matrix
Wmat <- mydata1$W
mydata2 <- CAR.simWmat(rho = rho, prec = prec, W = Wmat)

## Simualte data from a linear model with CAR error
y <- XX %*% beta + mydata1$X
mydata1$data.vec <- data.frame(y=y, XX[,-1])
mydata3 <- CAR.simLM(pars = c(0.1, 1, 2, 0.5), data = mydata1)

## Simulate Binomial data with CAR latent variables
mydata4 <- CAR.simGLM(method="binom", n=c(10,10), pars = c(rho, sigma,
                      beta), Xs=XX, n.trial = 5)
```

---

`loglik.dCAR` *Likelihood computing and parameter estimation for a direct CAR model.*

---

## Description

This help page documents a few functions for exact evaluation of the log-likelihood and profile log-likelihood for a direct CAR model, the maximum pseudo-likelihood estimator, and least square estimators for beta and sigma given the spatial coefficient rho in the CAR covariance matrix. The exact evaluation computes the log determinant in the log-likelihood function with eigen-values of the spatial weight matrix.

## Usage

```
loglik.dCAR(pars, data, rho.cons = c(-0.249, 0.249))
ploglik.dCAR(rho, data)
```

```
get.beta.lm(rho, data)
sigmabeta(rho, data)

mple.dCAR(data, tol = 1e-06, rho0=0)
```

## Arguments

| | |
|---|---|
| pars | the parameter value to be evaluated |
| rho | the value of the spatial coefficient in the CAR covariance to be evaluated |
| data | a list containing the following objects:<br>$W$ the spatial weight matrix,<br>**lambda** the eigenvalues of $W$ (if given),<br>**data.vec** a data frame of the response variable and covariates. |
| rho.cons | rho domain interval |
| tol | tolerance for the relative difference for two consequtive iterations in finding the MPLE; default set to be = 1e-06 |
| rho0 | starting value for iteratively finding the MPLE; default value set to be 0. |

## Details

The eigen-values can be supplied in the `data` list if the likelihood for the same data is going to be evaluated many times; if not supplied the function use the function `eigen` to find the eigen-values of the weight matrix.

## Value

`loglik.dCAR` and `ploglik.dCAR` return a numeric value of the log-likelihood evaluation. `get.beta.lm`, `sigmabeta` and `mple` return a numeric array of the estimates.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

`CAR.simLM`, `get.beta.glm`, `mcl.dCAR`

## Examples

```
## Simulate data from a torus
set.seed(30)
n.torus <- 20
rho <- 0.15
sigma <- 1.5
beta <- c(1, 2)
XX <- cbind(rep(1, n.torus^2), log(1:n.torus^2))
mydata <- CAR.simTorus(n.torus, n.torus, rho, 1/sigma)
y <- XX %*% beta + mydata$X
```

```
mydata$data.vec <- data.frame(y=y, XX[,-1])

## evaluate the log-likelihood without lambda
loglik.dCAR(pars = c(0.1, 1, 0.9, 2.1), data = mydata)

## evaluate the log-likelihood with lamda
lambda <- eigen(mydata$W, symmetric = TRUE, only.values=TRUE)$values
mydata$lambda <- lambda
loglik.dCAR(pars = c(0.1, 1, 0.9, 2.1), data = mydata)

## evaluate the profile log-likelihood of rho
ploglik.dCAR(rho = 0.1, data = mydata)

## given rho = 0.1, find the least square estimates for beta and sigma
get.beta.lm(rho = 0.1, data = mydata)
sigmabeta(rho = 0.1, data = mydata)

## find the maximum pseudo-likelihood estimates
mple.dCAR(data = mydata)
```

---

`mcl.dCAR` *Monte Carlo likelihood calculation for direct CAR models.*

---

## Description

This help page contains functions for calculating the Monte Carlo likelihoods and variances for direct CAR models.

`mcl.prep.dCAR` generates the Monte Carlo samples from the importance sampling distribution to be used in the Monte Carlo likelihood functions.

`mcl.dCAR` calculates the Monte Carlo log-likelihood ratio for a given parameter value to the importance sampler value.

`mcl.profile.dCAR` calculates the Monte Carlo profile log-likelihood ratio of rho.

`vmle.dCAR` calculates the variance at the Monte Carlo MLE.

`Avar.lik.dCAR` calculates the asymptotic variance of the Monte Carlo likelihood for a direct CAR model on a given size of torus and given size of Monte Carlo samples.

## Usage

```
mcl.prep.dCAR(psi, n.samples, data)

mcl.dCAR(pars, data, simdata, rho.cons = c(-0.249, 0.249), Evar = FALSE)
mcl.profile.dCAR(rho, data, simdata, rho.cons = c(-0.249, 0.249), Evar = FALSE)

vmle.dCAR(MLE, data, simdata)
Avar.lik.dCAR(pars, psi, data, n.samples, Log = TRUE)
```

## Arguments

| | |
|---|---|
| psi | the value of the importance sampler parameters |
| n.samples | the number of Monte Carlo samples |
| pars | the parameter value for the Monte Carlo likelihood function to be evaluated at |
| rho | the value of rho for the Monte Carlo profile likelihood function to be evaluated at |
| data | a data object same as described in loglik.dCAR |
| simdata | a list object in the same format as the output of mcl.prep.dCAR |
| rho.cons | rho domain interval |
| Evar | if TRUE return the estimated variance of the Monte Carlo likelihood |
| MLE | a list object with<br>par the estimated Monte Carlo MLE<br>hessian the hessian at the Monte Carlo MLE |
| Log | if log = TRUE, then the variance is of the log-likelihood ratio; otherwise it is of the likelihood ratio |

## Details

The asymptotic and estimated variance is derived for a direct CAR model on a $n \times n$ torus with 'rook' style binary neighbourhood matrix. Details of the derivation is given in Sha (2016) ch. 3.

## Value

mcl.prep.dCAR returns a list object containing the following following elements:

- simYa matrix of the Monte Carlo samples
- t10 and t20statistics pre-calculate for the importance distribution

mcl.dCAR and mcl.proifle.dCAR return a numeric value of the Monte Carlo likelihood when Evar = FALSE; if TRUE it returns an array of the Monte Carlo likelihood and the corresponding estimated variance.

vmle.dCAR returns the estimated covariance matrix of the Monte Carlo MLE.

Avar.lik.dCAR returns a numeric value of the asymptotic variance.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## References

Sha, Z. 2016 *Estimating conditional auto-regression models*, DPhil Thesis, Oxford.

## See Also

mcl.glm, OptimMCL, rsmMCL

## Examples

```
## Simulate some data to work with
set.seed(30)
n.torus <- 10
rho <- 0.15
sigma <- 1.5
beta <- c(1, 2)
XX <- cbind(rep(1, n.torus^2), log(1:n.torus^2))
mydata1 <- CAR.simTorus(n.torus, n.torus, rho, 1/sigma)
y <- XX %*% beta + mydata1$X
mydata1$data.vec <- data.frame(y=y, XX[,-1])

## Choose the importance sampler value to the the MPLE
psi1 <- mple.dCAR(mydata1)

## Prepare the Monte Carlo samples
mcdata1 <- mcl.prep.dCAR(psi = psi1, n.samples = 100, data = mydata1)

## Calculate the Monte Carlo likelihoods
mcl.dCAR(c(rho, sigma, beta), data = mydata1, simdata = mcdata1, Evar =
TRUE)
mcl.profile.dCAR(rho, data = mydata1, simdata = mcdata1, Evar = TRUE)

## Do a direct optimization of the Monte Carlo likelihood function
## to find the MLE but it takes very long time to run and may not converge
## Not run:
opt <- optim(psi1, fn = mcl.dCAR,
             data = mydata1, simdata = mcdata1,
             hessian = TRUE, control = list(fnscale = -0.5))

## End(Not run)

## Assume the we have obtained the opt with the following values
opt<- list(par = c (0.08013547, 1.70294099, 0.01571957, 2.23203089),
           hessian =  matrix(c(  -190.8791352,  -1.5682773,    0.1863733,   -4.844151,
 -1.5682773,  -16.1605186,    0.4911009,    4.403844,
 0.1863733,   0.4911009,  -41.1496774, -156.686631,
-4.8441507,   4.4038442, -156.6866312, -634.316017), 4, 4))

## Calculate the variance of the MC-MLE
vmle.dCAR(opt, data = mydata1, simdata = mcdata1)

## Calculate the asymptotic variance of the likelihood at MC-MLE
Avar.lik.dCAR(pars = opt$par, psi = psi1, data = mydata1, n.samples =
100)
```

---

`mcl.glm` *Monte Carlo likelihood calculation for glm with latent CAR variables.*

---

## Description

This help page contains functions for calculating the Monte Carlo likelihoods and variances for generalised linear models (Binomial and Poisson) with latent CAR variables.

`mcl.prep.glm` simulates the Monte Carlo samples from the importance sampling distribution to be used in the Monte Carlo likelihood functions. The samples are from an MCMC Chain generated by the Metropolis-Hastings Algorithms. Details see `postZ`.

`mcl.glm` calculates the Monte Carlo log-likelihood ratio for a given parameter value to the importance sampler value.

`mcl.profile.glm` calculates the Monte Carlo profile log-likelihood ratio of rho and sigma.

`vmle.glm` calculates the variance at the Monte Carlo MLE.

`get.beta.glm` finds the linear coefficients of the glm with given CAR parameters by solving the Monte Carlo gradient with `nleqslv`.

## Usage

```
mcl.prep.glm(data, family, psi, Z.start = NULL, mcmc.control=list(),
             pilot.run = TRUE, pilot.plot = FALSE, plot.diag = FALSE)

mcl.glm(pars, mcdata, family, Evar = FALSE)
mcl.profile.glm(pars, beta0, mcdata, family, Evar = FALSE)

vmle.glm(MLE, mcdata, family)

get.beta.glm(beta0, rho.sig, mcdata, family)
```

## Arguments

| | |
|---|---|
| `data` | a list object given by `CAR.simGLM` |
| `family` | a character equal to either "binom" for Binomial variables or "poisson" for Poisson variables |
| `psi` | the value of the importance sampler parameters |
| `Z.start` | initial value for simulating the MCMC algorithm |
| `mcmc.control` | a list of parameter that control the MCMC algorithm, see more in the Details of `postZ`. |
| `pilot.run` | if TRUE, a pilot run is done for tuning the MCMC parameters |
| `pilot.plot` | if TRUE, the MCMC diagnostic plots are plotted for the pilot run |
| `plot.diag` | if TRUE, the MCMC diagnostic plots are plotted for the simulated chain |
| `pars` | the parameter value of the likelihood function to be evaluated at |
| `mcdata` | a list of data generated by `mcl.prep.glm` |
| `Evar` | if TRUE return the estimated variance of the Monte Carlo likelihood |
| `MLE` | a list object with |
| | `par` the estimated Monte Carlo MLE |
| | `hessian` the hessian at the Monte Carlo MLE |
| `beta0` | starting point for finding the beta |
| `rho.sig` | values of rho and sigma for the CAR covariance matrix |

## Details

The Monte Carlo samples can be simulated by different MCMC algorithms described in postZ. The log determinant is calculated directly in the evaluation of the Monte Carlo likelihood and the Monte Carlo approximation is only used to approximate the integral for the latent variables.

Given the CAR covariance matrix, the betas are found by solving the non-linear system given by letting the gradient of the Monte Carlo likelihood with respect to beta equal to zero. Due to the Monte Carlo error, the nleqslv use only a maximum of two iteration to find a local solution.

## Value

mcl.prep.glm returns a list object that contains all the output of CAR.simGLM plus the following following elements:

- Zy, a matrix of the Monte Carlo samples,
- lHZy.psi, a list of statistics pre-calculate for the importance distribution,
- mcmc.pars, a list of the mcmc tuning parameter used in the simulation.

mcl.glm and mcl.proifle.glm return a numeric value of the Monte Carlo likelihood when Evar = FALSE; if TRUE it returns an array of the Monte Carlo likelihood and the corresponding estimated variance.

vmle.glm returns the estimated covariance matrix of the Monte Carlo MLE.

get.beta.glm returns an object given by nleqslv.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

postZ, mcl.dCAR, OptimMCL, rsmMCL

## Examples

```
## Simulate some data to work with
set.seed(33)
n.torus <- 10
nb <- 30
rho <- 0.2
sigma <- 1.5
beta <- c(1, 1)
pars.true <- c(rho, sigma, beta)
X0 <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))

mydata2 <- CAR.simGLM(method = "binom", n = c(n.torus, n.torus), pars = pars.true,
                      Xs = as.matrix(X0), n.trial = nb)

## Prepare the Monte Carlo samples
## Find a suitable initial value for the importance sampler parameter, e.g:
library(spatialreg)
```

```
library(spdep)
data.glm <- data.frame(y=mydata2$y, mydata2$covX[,-1])
fit.glm <- glm(cbind(y, nb-mydata2$y) ~ .,data = data.glm, family=binomial)
logitp <- log((mydata2$y+0.5)/(mydata2$n.trial - mydata2$y + 0.5))
data.splm <- data.frame(y=logitp, mydata2$covX[,-1])
listW <- mat2listw(mydata2$W)
fit.splm <- spautolm(y~., data = data.splm, listw=listW, family = "CAR")
psi.binom <- c(fit.splm$lambda, fit.splm$fit$s2, coef(fit.glm))

## Set the parameters for the MCMC algorithm
mc.pars <- list(N.YZ =1e3, N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)), thin = 5,
                  burns = 5e2, method = "mala", scale.fixed = TRUE)

mc.data2 <- mcl.prep.glm(data = mydata2, family = "binom",
                         psi = psi.binom, mcmc.control = mc.pars,
                         pilot.plot = TRUE, plot.diag = TRUE)

## Calculate the Monte Carlo likelihoods
pars.t <- c(rho, sigma, beta)

mcl.glm(pars.t, family = "binom",  mcdata = mc.data2, Evar = TRUE)

## Do a direct optimization of the Monte Carlo likelihood function to find the MLE
## Not run:
library(maxLik)
A <- matrix(0, nrow = 3, ncol = length(pars.t))
A[,1] <- c(1, -1, 0)
A[,2] <- c(0, 0, 1)
B <- c(0.24, 0.24, -0.1)

mle.bin <- maxBFGS(fn = mcl.glm, start=as.numeric(psi.binom), print.level=1,
                  constraints=list(ineqA=A, ineqB=B),
                  mcdata = mc.data2, family = "binom")

## End(Not run)
## Assume we have obtained mle.bin
mle.bin <- list(estimate = c(0.2271854 , 1.6117040, -0.3870856,  2.6525503),
                hessian = matrix(c( -1981.34398, -71.992190,  -79.910745,  -63.451022,
                                    -71.99219, -13.985701,    2.965628,    2.216893,
                                    -79.91074,   2.965628, -251.320742, -176.993087,
                                   -63.45102,   2.216893, -176.993087, -132.575284), 4,4))

## Calculate the variance of the MC-MLE
v.mle.bin <- vmle.glm(mle.bin, mcdata = mc.data2, family = "binom")


## Find the Monte Carlo MLE of beta, given the value of rho and sigma
get.beta.glm(beta0 = psi.binom[-c(1,2)], rho.sig = psi.binom[1:2],
             mcdata = mc.data2, family = c("binom"))
```

---

mcl.HCAR *The Monte Carlo likelihood function of the HCAR model.*

---

## Description

Evaluate the Monte Carlo likelihood of a hierarchical conditional-autoregressive model at a given parameter value and importance sampler value.

## Usage

```
mcl.HCAR(pars, mcdata, data, Evar = FALSE)
```

## Arguments

| | |
|---|---|
| pars | parameter values at which the Monte Carlo likelihood function to be evaluated |
| mcdata | The Monte Carlo sample object returned by the function sim.HCAR |
| data | A list or an environment containing the same variables same as the input for sim.HCAR, |
| Evar | If true, the Monte Carlo variance is evaluated. |

## Value

| | |
|---|---|
| mc.lr | the Monte Carlo likelihood |
| v.lr | if Evar = TRUE, the Monte Carlo variance of the likelihood is also returned |

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

sim.HCAR, OptimMCL.HCAR, summary.OptimMCL.HCAR

---

# OptimMCL *Iterative procedure for maximising the Monte Carlo likelihood*

---

## Description

The function uses an iterative procedure of directly maximising the Monte Carlo likelihood and the updating step size is limited by defining an experimental region using the estimated Monte Carlo variance.

## Usage

```
OptimMCL(data, psi0, family, control = list(), mc.control = list())
```

## Arguments

| | |
|---|---|
| `data` | the data set to be estimated; for direct CAR model it is a list objects same used in `loglik.dCAR` and for glm with CAR latent variables, it is the same as the output of `CAR.simGLM`. |
| `psi0` | the initial value for the importance sampler parameter |
| `family` | a character take values in "gauss" (for direct CAR models), "binom" and "poisson" (glm Binomial/Poisson with CAR latent variable). |
| `control` | a list that controls the iterative procedures, see more in Details. |
| `mc.control` | a list that controls the MCMC algorithm, see more in Details of `postZ` |

## Details

The iterative procedure starts by using samples from the importance sampler with the parameter values given by `psi0` and directly optimise the Monte Carlo likelihood. Then the importance sampler parameter value is updated by using either the MLE found by current step if it is within the experimental region (to be defined in the following), or a weighted average between the current psi value and the current MLE value, with weight to be specified by `psi.ab` in the list of `control` argument.

The experimental region is defined to be the region where the estimated variance of the Monte Carlo likelihood is smaller than $a/\sqrt{(n.s)}$ where $n.s$ is the number of Monte Carlo samples and $a$ and is some fixed constant that can be specified through `psi.ab` in `control`.

The procedure iterate until convergence which is defined to be that the Monte Carlo likelihood is smaller than min(1, 2*sqrt(mcl.var)) where mcl.var is the variance of the estimated variance of the Monte Carlo likelihood evaluated at the MC-MLE. At this point, the procedure use a increased number of Monte Carlo samples and do the same iteration until converges again. The MC-MLE found on this second convergence is given as the final result. The Monte Carlo sample size is increased by a multiple of user specified number through the `control` argument.

The `control` argument is a list containing the following elements:

**n.iter,** maximum number of iterations; default value is 10 for direct CAR models and 20 for glm with latent CAR variables

**mc.samples,** (for direct CAR model only) an array of two with the first element being the number of Monte Carlo samples and the second being the increasing multiple of the sample size after the first convergence; default is c(500, 2)

**s.increase,** (for glm with latent CAR variables only) an numeric value for the increasing multiple of the sample size after the first convergence; default value is 2.

**rho.range,** the valid range for the parameter rho in the direct CAR model; default value is c(-0.249, 0.249)

**psi.range,** the valid range for the parameters rho and sigma in the glm with latent CAR variables; default is c(-0.249, 0.249, 0.1, 10)

**psi.ab,** an array with two entries controlling the experimental region and updating rules; the first entry controls the experimental region as $a$ defined above and the second entry controls the updating rule as defined above; default is c(1, 0.5)

**mc.var,** if true, the estimated covariance matrix for the MC-MLE is returned in every iteration; default is FALSE

**trace.all,** if true, each iteration is stored and returned in the final output, default is TRUE

**verbose,** if true, an summary message is printed after each iteration, default is TRUE

## Value

When trace.all is TRUE, the function returns a list containing the following objects:

**Psi,** a list of importance sampler parameter values from all iterations;

**MC.datas,** a list object generated by `mcl.prep.dCAR` or `mcl.prep.glm` from all iterations;

**MC.MLEs,** a list of the MC-MLE found by all iterations;

**MC.Hess,** a list of the Hessian matrix at the MC-MLE from all iterations;

**MC.Vars,** a list of the estimated covariance matrix of the MC-MLE from all iterations;

**data,** the data object supplied to the function;

**N.iter,** the total number of iterations;

**total.time,** the total time elapsed;

**convergence,** a logical value indicating whether the procedure converged or not;

**mcsamples,** an array of two entries for the initial Monte Carlo sample size and the increased Monte Carlo sample size after the first convergence.

When trace.all is FALSE, the function returns the same list with each object in the list containing the result of the final iteration only.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## References

Sha, Z. 2016 *Estimating conditional auto-regression models*, DPhil Thesis, Oxford.

## See Also

`summary.OptimMCL`, `rsmMCL`

## Examples

```
## Take long time to run
## Simulate some data to work with
set.seed(33)
n.torus <- 10
nb <- 30
rho <- 0.2
sigma <- 1.5
beta <- c(1, 1)
pars.true <- c(rho, sigma, beta)
X0 <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
mydata2 <- CAR.simGLM(method = "binom", n = c(n.torus, n.torus), pars = pars.true,
```

```
                    Xs = as.matrix(X0), n.trial = nb)

## use a glm to find initial values for the importance sampler
data.glm <- data.frame(y=mydata2$y, mydata2$covX[,-1])
fit.glm <- glm(cbind(y, nb-mydata2$y) ~ .,data = data.glm, family=binomial)
library(spatialreg)
library(spdep)
logitp <- log((mydata2$y+0.5)/(mydata2$n.trial - mydata2$y + 0.5))
data.splm <- data.frame(y=logitp, mydata2$covX[,-1])
listW <- mat2listw(mydata2$W)
fit.splm <- spautolm(y~., data = data.splm, listw=listW, family = "CAR")
pars1 <- c(fit.splm$lambda, fit.splm$fit$s2, coef(fit.glm))

## Use the iterative procedure to find the MC-MLE
## !!!NOTE: the example below is only an illustration of usage
## users should increase the number of iterations and MCMC samples
## to get convergence results.
iter.mcmle <- OptimMCL(data = mydata2, psi0 = pars1, family = "binom",
                       control = list(n.iter = 1, mc.var = TRUE),
                    mc.control = list(N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)), thin = 5,
                   burns = 5e2, method = "mala", scale.fixed = TRUE))
summary(iter.mcmle, family = "binom", mc.covar=TRUE)
```

---

`OptimMCL.HCAR` *Iterative procedure for maximising the Monte Carlo likelihood of the hierachical conditional auto-regressive models.*

---

## Description

The function uses an iterative procedure of directly maximising the Monte Carlo likelihood of a hierechical conditional auto-regressive model and the updating step size is limited by defining an experimental region using the estimated Monte Carlo variance.

## Usage

```
OptimMCL.HCAR(data, psi0,  control = list())
```

## Arguments

| | |
|---|---|
| `data` | A list or an environment contains the variables same as described in `sim.HCAR`. |
| `psi0` | Starting value for the importance sampler parameter same as described in `sim.HCAR`. |
| `control` | a list of tuning parameters to control the algorithm. Details to be found at `OptimMCL` |

## Value

Same as in `OptimMCL`.

### Author(s)

Zhe Sha <zhesha1006@gmail.com>

### See Also

mcl.HCAR, sim.HCAR, summary.OptimMCL.HCAR

---

plot.rsmMCL *Plot the fitted response surfaces*

---

### Description

This function plot the fitted response surfaces by using the result from rsmMCL.

### Usage

```
## S3 method for class 'rsmMCL'
plot(x, family, trace.all = TRUE, plain=TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | an rsmMCL object retruned by rsmMCL. |
| family | a character takes value in "guass", "binom" and "poisson". |
| trace.all | an logic value tells whether the input object given by rsmMCL contains results from all iterations or not. |
| plain | an logic value indicates whether to show plots in separate devices with a smart display. Use the defalut value TRUE if the function is used in generating documents by using **knitr**. |
| ... | For extra arguments. The Lattice parameter layout can be used to control the layout of the graphs. |

### Author(s)

Zhe Sha <zhesha1006@gmail.com>

### See Also

rsmMCL, summary.rsmMCL

### Examples

```
## See examples for rsmMCL
```

postZ *Sampling the CAR latent variables given the Binomial or Poisson observations in glm models.*

## Description

The function uses several different MCMC algorithms to sample the CAR latent variables Z from the distribution P(Z|y) given the Binomial or Poisson observations in glm models using the importance sampler parameter values psi.

## Usage

```
postZ(data, family, psi, Z.start, mcmc.control = list(), plots = FALSE)
```

## Arguments

| | |
|---|---|
| data | the data set to be estimated and the object has same as the output of CAR.simGLM |
| family | a character take values in "binom" and "poisson" for Binomial or Poisson glm with CAR latent variable). |
| psi | Parameters for the importance distribution. |
| Z.start | Initial value of the CAR variables for the Markov Chain. |
| mcmc.control | a list that controls the MCMC algorithm, see more in Details. |
| plots | if TRUE, the diagnostic plots are shown on running the function. |

## Details

The function uses two MCMC algorithms, the Metropolis Hastings algorithm with random walk proposal (rwmh) and the Metropolis adjusted Langevin algorithm (mala), in sampling the CAR latent variables. The scale parameter for the proposal distribution can be tuned to achieve the optimal acceptance rate by using an adaptive algorithm in a pilot run. The argument mcmc.control contains the following objects

- N.Zy, the number of iterations of the Markov Chain
- Scale, the scale for the proposal distribution
- thin, the thinning step
- burns, the number of burn-in samples to be discarded
- method, the name of the MCMC algorithm, either "rwmh" or "mala"
- scale.fixed, if true, the scale parameter is fixed; otherwise an adaptive MCMC algorithm is run for tuning the scale parameter to achieve the optimal acceptance rate by using an stochastic approximation algorithm
- c.ad, an array contains the two values for the tuning parameter for the stochastic approximation algorithm in the adaptive MCMC algorithm; default is set to be c(1, 0.7)

## Value

The function return a list of the following objects:

- Z a matrix of samples from from P(Z|y)
- AC.ran array of the acceptance rate updated at each iteration
- Scalesif in the list of mcmc.control scale.fixed = FALSE, an array of the adaptive scales for the MCMC proposal

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

mcl.prep.glm, OptimMCL, rsmMCL

## Examples

```
set.seed(33)
n.torus <- 10
nb <- 30
rho <- 0.2
sigma <- 1.5
beta <- c(1, 1)
pars.true <- c(rho, sigma, beta)
X0 <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
mydata2 <- CAR.simGLM(method = "binom", n = c(n.torus, n.torus), pars = pars.true,
                      Xs = as.matrix(X0), n.trial = nb)

## use a glm to find initial values for the importance sampler
library(spatialreg)
library(spdep)
data.glm <- data.frame(y=mydata2$y, mydata2$covX[,-1])
fit.glm <- glm(cbind(y, nb-mydata2$y) ~ .,data = data.glm, family=binomial)
## estimate sigma and rho, transform the binomial to Gaussian by logit
logitp <- log((mydata2$y+0.5)/(mydata2$n.trial - mydata2$y + 0.5))
data.splm <- data.frame(y=logitp, mydata2$covX[,-1])
listW <- mat2listw(mydata2$W)
fit.splm <- spautolm(y~., data = data.splm, listw=listW, family = "CAR")
pars1 <- c(fit.splm$lambda, fit.splm$fit$s2, coef(fit.glm))

## Sample form importance distribution with psi = pars1
mc.control <- list(N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)), thin = 5,
                   burns = 5e2, method = "mala", scale.fixed = TRUE)
## Binomial
Z.S0 <- CAR.simWmat(pars1[1], 1/pars1[2], mydata2$W)
simZy <- postZ(data = mydata2, Z.start = Z.S0, psi = pars1,
               family = "binom", mcmc.control = mc.control, plots =
TRUE)
```

rsmMCL *Response surface method for maximising the Monte Carlo likelihood*

## Description

The function maximise the Monte Carlo likelihood by using the response surface methodology.

## Usage

```
rsmMCL(data, psi0, family, exact0 = NULL, control = list(), mc.control = list())
```

## Arguments

| | |
|---|---|
| `data, psi0, family, mc.control` | are the same as those in `OptimMCL` |
| `control` | see more in Details |
| `exact0` | the exact log likelihood ratio values of a given range or parameters to the importance sampler parameter value. This is used for comparing the fitted response surface of the Monte Carlo likelihood to the exact if the exact evaluation is required. |

## Details

The response surface methodology finds the maximum of a function by using first and second order polynomial approximations to the target function. The target function is usually unknown or expensive to evaluate. The function is approximated by a first order model in regions that are far away from the maximum and by a second order model around the maximum region. Then the stationary point of the second order model is used as an approximation to the maximum point. By changing the design region, the number and location of the evaluation points, the approximated surface can achieve different accuracy and variance properties. Details of the methodology can be found in Box, G.E.P. and Draper, N.R.(2007) and details of the implementation of the method in this package can be found Sha (2016).

The procedure implemented in this function locates the maximum region of the target function by iteratively updating the design centre and design region. Given a starting point as the design centre, the design region is given based on the finite variance region of the importance sampling approximation as described in Sha (2016), chap. 4. Then with the design region, design points are generated by the central composite design (Box, G.E.P. and Draper, N.R. (2007)). The Monte Carlo likelihood and the corresponding estimated variance at these design points are evaluated and used to fit the first or second order polynomials.

If the stationary point is not found or found to be outside the current design region then, either a steepest ascent analysis (for a first order approximation) or a ridge analysis (for a second order approximation) is done to find the next design centre. The updating step is also limited by considering the estimated variance of the Monte Carlo likelihood.

The procedure is defined to be converged once the stationary point is found and within the design region or the difference between the design centres in two consecutive iterations is smaller than a given tolerance. The Monte Carlo sample size is then increased by a multiple of $m$ once the

procedure converges. With the new Monte Carlo sample size a few more iterations are done until the procedure achieve convergence again. The result of the second procedure is given as the final output.

The `control` argument is a list containing the following elements:

**mc.samples,** an array of numbers specifying the Monte Carlo sample size(for direct CAR models only), the increasing multiple and the maximum Monte Carlo sample size to use.

**n.iter,** maximum number of iterations; default value is 20

**time.max,** maximum total time for the computation

**K,** an array of two numbers for generating the design region; the first number K[1] decide the initial size of the design region and once the system find stationary point within the design region, K[1] becomes K[1]/K[2] to get a smaller design region for better approximation

**mc.var,** if true, the estimated covariance matrix for the MC-MLE is returned in every iteration; default is FALSE

**psi.lim,** a list that contains the range for rho and sigma in the CAR covariance matrix; default values are r.lim = c(-0.2499, 0.2499) for rho and s.lim = c(0.1, 8) for sigma

**exacts** (for direct CAR only) a list containing the setting for plotting the exact values for comparison with the fitted rsm surface; the default values are eval = TRUE (for plotting the exact values, default for glm is FALSE), rho = c(-0.25, 0.25), sigma = c(0.5, 2) define the plotting region, length = 100 define number of grid point in each coordinate in producing the plotting grid

**rsm.fit** a list of numbers that controls the fitting of the response surfaces and the elements are

- n01, number of central points in a central design in generating the design points. The central points are used to estimate the variance of the evaluation and the default is 4.
- n02, number of central points in a composite design and the default is 2.
- Rsq, R-squared in the fitted first order model in judging the goodness of fit of the first order model; the default value is 0.9.
- lof, p-value of the lack of fit test for the first order model; the default value is 0.05.
- st.diff, control the range of the approximated function; the approximated surface has less variance for values closes to zero, so this should not be too large and the default value is set to be 50.
- mcl.diff, maximum value of the approximated function evaluated at the MC-MLE; again this value should be close to zero and the default is 10.

**plotrsm,** if true the approximated response surface with steepest or ridge analysis is plotted; defualt is TRUE

**trace.all,** if true, each iteration is stored and returned in the final output, default is TRUE

**verbose,** if true, an summary message is printed after each iteration, default is TRUE

## Value

When trace.all is TRUE, the function returns a list containing the following objects:

**psi0,** a vector of the parameter values used as the initial importance sampler parameter values

**Psi,** a list of importance sampler parameter values from all iterations

**FVbox,** a list containing the finite variance regions for all iterations

**DAs,** a list containing the design region for all iterations

**DAbox,** a list containing the design region box for all iterations that can be used for plotting

**Ttime,** a list containing the computational time used in each iteration

**Expts,** a list of design points in each iteration

**Expt.Val1,** a list of all values evaluated at the central design points in each iteration

**Expt.Val2,** a list of all values evaluated at the composite design points in each iteration

**RSM.1,** a list of all the fitted first-order response model

**RSM.2,** a list of all the fitted second-order response model

**ESAs,** a list of all explorations along the steepest ascent path

**Sim.data,** a list object generated by `mcl.prep.dCAR` or `mcl.prep.glm` from all iterations;

**Exacts,** a list of exact values in each iteration for the direct CAR models

**data,** the data object supplied to the function;

**N.iter,** the total number of iterations;

**total.time,** the total time elapsed;

**convergence,** a logical value indicating whether the procedure converged or not;

**mcsamples,** an array of two entries for the initial Monte Carlo sample size and the increased Monte Carlo sample size after the first convergence.

When trace.all is FALSE, the function returns the same list with each object in the list containing the result of the final iteration only.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## References

Box, G. E. P. and Draper, N. R. (2007) *Response Surfaces, Mixtures, and Ridge Analyses*, Wiley, New York

Sha, Z. 2016 *Estimating conditional auto-regression models*, DPhil Thesis, Oxford.

## See Also

`summary.rsmMCL`, `plot.rsmMCL`, `OptimMCL`

## Examples

```
#### Poisson glm with CAR latent variables
## Simulate some data
set.seed(33)
n.torus <- 10
nb <- 30
rho <- 0.2
sigma <- 1.5
```

```
beta <- c(1, 1)
pars.true <- c(rho, sigma, beta)
X0 <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
mydata3 <- CAR.simGLM(method = "poisson", n = c(n.torus, n.torus),
                      pars = pars.true, Xs =as.matrix(X0))

## Fit by rsm
rsm.mcmle2 <- rsmMCL(data = mydata3, psi0 = c(0, 1, 2, 2), family = "poisson",
                     control = list(n.iter = 2, trace.all = TRUE),
                     mc.control = list(N.Zy = 1e3, Scale = 1.65/(n.torus^(2/6)),
                                       thin = 5, burns = 5e2,
                                       method = "mala", scale.fixed = TRUE))
summary(rsm.mcmle2, family = "poisson",  mc.covar=TRUE)
plot(rsm.mcmle2, family = "poisson")
```

---

ScotCancer *Scottish lip cancer dataset from Clayton and Kaldor (1987)*

---

## Description

The Scottish lip cancer dataset.

## Format

The data object for the Monte Carlo likelihood estiation and it follows the same structure as described in `CAR.simGLM`. The covX is the model matrix with intercept and the fixed effect Paff

## Source

Clayton D, Kaldor J 1987. Empirical Bayes Estimation of Age-standardized Relative Risk for use in Disease Mapping. *Biometrics* 43, 671–681

Ronnegard L, Shen X, and Alam M 2010 hglm: A Package for Fitting Hierarchical Generalized Linear Models. The R Journal, 2(2): 20-28. URL http://journal.r-project.org/archive/2010-2/RJournal_2010-2_Roennegaard~et~al.pdf.

## References

Clayton D, Kaldor J 1987. Empirical Bayes Estimation of Age-standardized Relative Risk for use in Disease Mapping. *Biometrics* 43, 671–681

## scotplot *Scottish lip cancer dataset from Clayton and Kaldor (1987) with geo information.*

### Description

The Scottish lip cancer dataset with geo information for plotting.

### Format

A SpatialPolygonDataFrame contains the geo information and data for plotting

### Source

Clayton D, Kaldor J 1987. Empirical Bayes Estimation of Age-standardized Relative Risk for use in Disease Mapping. *Biometrics* 43, 671–681

Lee D 2016. CARBayes: Spatial Generalised Linear Mixed Models for Areal Unit Data. R package version 4.6. https://CRAN.R-project.org/package=CARBayes

### References

Clayton D, Kaldor J 1987. Empirical Bayes Estimation of Age-standardized Relative Risk for use in Disease Mapping. *Biometrics* 43, 671–681

## sim.HCAR *Simulate samples from a HCAR model.*

### Description

Simulate samples from a hierarchical conditional-autoregressive model.

### Usage

```
sim.HCAR(psi, data, n.samples)
```

### Arguments

| | |
|---|---|
| psi | A vector of the length of all parameters used for the importance sampler. When the spatial weight matrix W for individual unit is supplied in the data, the first four entries of psi are<br>**psi[1** = rho,] the spatial coefficient parameter for individual units,<br>**psi[2** = lambda,] the spatial coefficient parameter for district level units,<br>**psi[3** = sigma.e] the marginal variance parameter for individual units,<br>**psi[4** = sigma.u] the marginal variance parameter for district level units. |

Then the rest of entries are values for the linear coefficient beta. Otherwise, the function assumes that there is no spatial correlation among the individual unit and the first three entries are used to specify lambda, sigma.e, sigma.u and the rest for beta.

`data` A list or an environment contains the following variables

**y,** the observed linear response,
**X,** the design matrix for the fixed effects,
**W,** the spatial weight matrix for individual units,
**M,** the spatial weight matrix for district level units,
**Z,** the design matrix for the random effects,
**In** an n by n identity matrix, where n is the number of individual spatial units,
**Ik** a k by a indentity matrix, where k is the number of distrct level unites.

`n.samples` the number of Monte Carlo samples

## Value

`psi` the importance sampler value used for generating the samples,

`n.samples` the number of Monte Carlo samples,

`u.y` the Monte Carlo samples of the randome effects,

`lpsi` the unnormalsied likelihood components evaluated at psi,

`const.psi` the constant in the likelihood function evaluated at psi

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

`mcl.HCAR`, `OptimMCL.HCAR`, `summary.OptimMCL.HCAR`

---

# summary.OptimMCL *Summary the output from the iterative procedure of maximising the Monte Carlo likelihood.*

---

## Description

This function summarizes the output of the output from the function `OptimMCL`.

## Usage

```
## S3 method for class 'OptimMCL'
summary(object, family, trace.all = TRUE, mc.covar =
  TRUE, ...)
```

### Arguments

| | |
|---|---|
| object | an OptimMCL object returned by OptimMCL. |
| family | a character takes value in "guass", "binom" and "poisson". |
| trace.all | an logic value tells whether the input object given by OptimMCL contains results from all iterations of not |
| mc.covar | if TRUE, the estimated covariance matrix of the MC-MLE is returned |
| ... | arguments passed to or from other methods. |

### Value

A list containing the following objects:

**MC.mle,** the final MC-MLE

**N.iter,** the total number of iterations

**total.time,** the total time elapsed

**convergence,** if TRUE the procedure converges

**hessian,** the Hessian at the MC-MLE if given; the default is NULL

**mc.covar** the estimated covariance matrix of the MC-MLE if given; the default is NULL

**mc.samples** the Monte Carlo samples size used in the initial stage and after the first convergence.

### Author(s)

Zhe Sha <zhesha1006@gmail.com>

### See Also

OptimMCL

### Examples

```
## See examples for OptimMCL
```

---

## summary.OptimMCL.HCAR *Summary the output from the iterative procedure of maximising the Monte Carlo likelihood.*

---

### Description

The summary function summarizes the output of the output from the function OptimMCL.HCAR and the ranef.HCAR calculate the empirical Bayesian estimates of the random effects given the Monte Carlo maximum likelihood estimates.

## Usage

```
## S3 method for class 'OptimMCL.HCAR'
summary(object, trace.all = TRUE, mc.covar =
  TRUE, ...)

ranef.HCAR(pars, data)
```

## Arguments

| | |
|---|---|
| object | an OptimMCL object returned by OptimMCL.HCAR. |
| trace.all | an logic value tells whether the input object given by OptimMCL.HCAR contains results from all iterations of not |
| mc.covar | if TRUE, the estimated covariance matrix of the MC-MLE is returned |
| ... | arguments passed to or from other methods. |
| pars | the paramter values for calculating the empirical Bayesian estimates of the random effects; a list or enivironment of data for example same as described in sim.HCAR |
| data | A list or an environment contains the variables same as described in sim.HCAR. |

## Value

The summary function returns a list containing the following objects:

**MC.mle,** the final MC-MLE

**N.iter,** the total number of iterations

**total.time,** the total time elapsed

**convergence,** if TRUE the procedure converges

**hessian,** the Hessian at the MC-MLE if given; the default is NULL

**mc.covar** the estimated covariance matrix of the MC-MLE if given; the default is NULL

**mc.samples** the Monte Carlo samples size used in the initial stage and after the first convergence.

The ranef.HCAR function return a dataframe object containing the estimated random effects and their corresponding standard deviations.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

mcl.HCAR, sim.HCAR, OptimMCL.HCAR

## Examples

```
## See examples for OptimMCL
```

summary.rsmMCL *Summary the output from the response surface method of maximising the Monte Carlo likelihood*

## Description

This function summarizes the output of the output from the function rsmMCL.

## Usage

```
## S3 method for class 'rsmMCL'
summary(object, family, trace.all = TRUE, mc.covar =
TRUE, ...)
```

## Arguments

| | |
|---|---|
| object | an rsmMCL object returned by rsmMCL. |
| family | a character takes value in "gauss", "binom" and "poisson". |
| trace.all | an logic value tells whether the input object given by rsmMCL contains results from all iterations of not |
| mc.covar | if TRUE, the estimated covariance matrix of the MC-MLE is returned |
| ... | arguments passed to or from other methods. |

## Value

A list containing the following objects:

**MC.mle,** the final MC-MLE

**N.iter,** the total number of iterations

**total.time,** the total time elapsed

**convergence,** if TRUE the procedure converges

**hessian,** the Hessian at the MC-MLE if given; the default is NULL

**mc.covar** the estimated covariance matrix of the MC-MLE if given; the default is NULL

**mc.samples** the Monte Carlo samples size used in the initial stage and after the first convergence.

## Author(s)

Zhe Sha <zhesha1006@gmail.com>

## See Also

rsmMCL

## Examples

```
## See examples for rsmMCL
```

# Index