Package ‘EMJMCMC’

May 4, 2024

Type Package

Title Evolutionary Mode Jumping Markov Chain Monte Carlo Expert Toolbox

Version 1.5.0

Date 2024-05-02

Maintainer Waldir Leoncio <`w.l.netto@medisin.uio.no`>

Description Implementation of the Mode Jumping Markov Chain Monte Carlo algorithm from Hubin, A., Storvik, G. (2018) <doi:10.1016/j.csda.2018.05.020>, Genetically Modified Mode Jumping Markov Chain Monte Carlo from Hubin, A., Storvik, G., & Frommlet, F. (2020) <doi:10.1214/18-BA1141>, Hubin, A., Storvik, G., & Frommlet, F. (2021) <doi:10.1613/jair.1.13047>, and Hubin, A., Heinze, G., & De Bin, R. (2023) <doi:10.3390/fractalfract7090641>, and Reversible Genetically Modified Mode Jumping Markov Chain Monte Carlo from Hubin, A., Frommlet, F., & Storvik, G. (2021) <doi:10.48550/arXiv.2110.05316>, which allow for estimating posterior model probabilities and Bayesian model averaging across a wide set of Bayesian models including linear, generalized linear, generalized linear mixed, generalized nonlinear, generalized nonlinear mixed, and logic regression models.

License GPL

Depends R (>= 3.4.1), bigmemory

Imports glmnet, biglm, hash, BAS, stringi, parallel, methods, speedglm, stats, withr

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0), bindata, clusterGeneration, reshape2

Config/testthat/edition 3

Encoding UTF-8

NeedsCompilation no

Author Aliaksandr Hubin [aut],
Waldir Leoncio [cre, aut],
Geir Storvik [ctb],
Florian Frommlet [ctb]

Repository CRAN

Date/Publication 2024-05-03 23:10:02 UTC

R topics documented:

do.call.emjmcmc 2
erf 3
estimate.bas.glm 4
estimate.bas.lm 5
estimate.bigm 6
estimate.elnet 8
estimate.gamma.cpen 9
estimate.gamma.cpen_2 9
estimate.glm 10
estimate.logic.glm 12
estimate.logic.lm 13
estimate.speedglm 14
LogicRegr 15
m 18
parall.gmj 18
parallelize 20
pinferunemjmcmc 21
runemjmcmc 25
sigmoid 31
simplify.formula 32
simplifyposteriors 33
truncfactorial 33

Index **35**

`do.call.emjmcmc` *A help function used by parall.gmj to run parallel chains of (R)(G)MJMCMC algorithms*

Description

A help function used by parall.gmj to run parallel chains of (R)(G)MJMCMC algorithms

Usage

```
do.call.emjmcmc(vect)
```

Arguments

`vect`	a vector of parameters of runemjmcmc as well as several additional fields that must come after runemjmcmc parameters such as: **vect$simlen** the number of parameters of runemjmcmc in vect **vect$cpu** the CPU id for to set the unique seed **vect$NM** the number of unique best models from runemjmcmc to base the output report upon

Value

a list of

post.populi	the total mass (sum of the marginal likelihoods times the priors of the visited models) from the addressed run of runemjmcmc
p.post	posterior probabilities of the covariates approximated by the addressed run of runemjmcmc
cterm	the best value of marginal likelihood times the prior from the addressed run of runemjmcmc
fparam	the final set of covariates returned by the addressed run of runemjmcmc

See Also

runemjmcmc, parall.gmj

`erf` *erf activation function*

Description

erf activation function

Usage

```
erf(x)
```

Arguments

`x`	a real number

Value

`erf(x)`, erf value

Examples

```
erf(10)
```

`estimate.bas.glm` *Obtaining Bayesian estimators of interest from a GLM model*

Description

Obtaining Bayesian estimators of interest from a GLM model

Usage

```
estimate.bas.glm(formula, data, family, prior, logn)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	either poisson() or binomial(), that are currently adopted within this function
`prior`	BAS::aic.prior(), bic.prior() or ic.prior() are allowed
`logn`	log sample size

Value

A list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

BAS::bayesglm.fit

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (((X4$V50 * X4$V19 * X4$V13 * X4$V11) > 0)) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
```

```
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
X4$Y4 <- Y4
data.example <- as.data.frame(X4)
data.example$Y4 <- as.integer(data.example$Y > mean(data.example$Y))
formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)

estimate.bas.glm(
  formula = formula1,
  data = data.example,
  prior = BAS::aic.prior(),
  logn = 47,
  family = binomial()
)
```

`estimate.bas.lm` *Obtaining Bayesian estimators of interest from a LM model*

Description

Obtaining Bayesian estimators of interest from a LM model

Usage

```
estimate.bas.lm(formula, data, prior, n, g = 0)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`prior`	integers 1, 2 or 3 are allowed corresponding to AIC, BIC or Zellner's g-prior
`n`	sample size
`g`	g

Value

a list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

BAS::bayesglm.fit

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (((X4$V50 * X4$V19 * X4$V13 * X4$V11) > 0)) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
X4$Y4 <- Y4
data.example <- as.data.frame(X4)
formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)

estimate.bas.lm(formula = formula1, data = data.example, prior = 2, n = 47)
```

`estimate.bigm` *Obtaining Bayesian estimators of interest from a GLM model*

Description

Obtaining Bayesian estimators of interest from a GLM model

Usage

```
estimate.bigm(formula, data, family, prior, n, maxit = 2, chunksize = 1e+06)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used

`family`	distribution family foe the responses
`prior`	either "AIC" or "BIC"
`n`	sample size
`maxit`	maximum number of Fisher scoring iterations
`chunksize`	size of chunks for processing the data frame

Value

a list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

n sample size

See Also

biglm::bigglm

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (((X4$V50 * X4$V19 * X4$V13 * X4$V11) > 0)) +
    9 * (X4$V37 * X4$V20 * X4$V12) + 7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
X4$Y4 <- Y4
data.example <- as.data.frame(X4)
formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)
formula1 <- as.formula(
  paste(
    colnames(data.example)[1], "~ 1 +", paste0(colnames(data.example)[-1],
    collapse = "+")
```

```
  )
)
estimate.bigm(
  formula = formula1, data = data.example, n = 47, prior = "BIC", maxit = 20,
  chunksize = 1000000, family = gaussian()
)
```

`estimate.elnet` *A test function to work with elastic networks in future, be omitted so far*

Description

A test function to work with elastic networks in future, be omitted so far

Usage

```
estimate.elnet(formula, response, data, family, alpha)
```

Arguments

`formula`	a formula object for the model to be addressed
`response`	response in a formula
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	distribution of the response family object
`alpha`	regularization parameter in [0,1]

Value

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

glmnet::glmnet

`estimate.gamma.cpen` *Estimate marginal log posterior of a single BGNLM model*

Description

Estimate marginal log posterior of a single BGNLM model

Usage

```
estimate.gamma.cpen(
  formula,
  data,
  r = 1/1000,
  logn = log(1000),
  relat = c("cos", "sigmoid", "tanh", "atan", "sin", "erf")
)
```

Arguments

`formula`	formula
`data`	dataset
`r`	prior inclusion penalty parameter
`logn`	logn
`relat`	a set of nonlinear transformations in the class of BGNLMs of interest

Value

A list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

---

`estimate.gamma.cpen_2` *Estimate marginal log posterior of a single BGNLM model with alternative defaults*

Description

Estimate marginal log posterior of a single BGNLM model with alternative defaults

Usage

```
estimate.gamma.cpen_2(
  formula,
  data,
  r = 1/223,
  logn = log(223),
  relat = c("to23", "expi", "logi", "to35", "sini", "troot", "sigmoid")
)
```

Arguments

`formula`	formula
`data`	dataset
`r`	prior inclusion penalty parameter
`logn`	logn
`relat`	a set of nonlinear transformations in the class of BGNLMs of interest

Value

A list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

---

`estimate.glm` *Obtaining Bayesian estimators of interest from a GLM model*

Description

Obtaining Bayesian estimators of interest from a GLM model

Usage

```
estimate.glm(formula, data, family, prior, n = 1, g = 0)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	distribution family for the responses
`prior`	integers 1,2 or 3 corresponding to AIC, BIC or Zellner's g-prior
`n`	sample size
`g`	g parameter of Zellner's g prior

Value

a list of

mlik	marginal likelihood of the model
waic	AIC model selection criterion
dic	BIC model selection criterion
summary.fixed$mean	a vector of posterior modes of the parameters

See Also

glm

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (((X4$V50 * X4$V19 * X4$V13 * X4$V11) > 0)) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
X4$Y4 <- Y4
data.example <- as.data.frame(X4)
formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)

formula1 <- as.formula(
  paste(
    colnames(data.example)[1], "~ 1 +", paste0(colnames(data.example)[-1],
    collapse = "+")
  )
)
estimate.glm(
  formula = formula1, data = data.example, prior = 2, family = gaussian()
)
```

`estimate.logic.glm` *Obtaining Bayesian estimators of interest from a GLM model in a logic regression context*

Description

Obtaining Bayesian estimators of interest from a GLM model in a logic regression context

Usage

```
estimate.logic.glm(formula, data, family, n, m, r = 1)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	either poisson() or binomial(), that are currently adopted within this function
`n`	sample size
`m`	total number of input binary leaves
`r`	omitted

Value

a list of

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

BAS::bayesglm.fit estimate.logic.lm

Examples

```
X1 <- as.data.frame(
  array(data = rbinom(n = 50 * 1000, size = 1, prob = 0.3), dim = c(1000, 50))
)
Y1 <- -0.7 + 1 * ((1 - X1$V1) * (X1$V4)) + 1 * (X1$V8 * X1$V11) + 1 * (X1$V5 * X1$V9)
X1$Y1 <- round(1.0 / (1.0 + exp(-Y1)))

formula1 <- as.formula(
  paste(colnames(X1)[51], "~ 1 +", paste0(colnames(X1)[-c(51)], collapse = "+"))
)
```

```
estimate.logic.glm(
  formula = formula1, data = X1, family = binomial(), n = 1000, m = 50
)
```

`estimate.logic.lm` *Obtaining Bayesian estimators of interest from an LM model for the logic regression case*

Description

Obtaining Bayesian estimators of interest from an LM model for the logic regression case

Usage

```
estimate.logic.lm(formula, data, n, m, r = 1)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`n`	sample size
`m`	total number of input binary leaves
`r`	omitted

Value

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

BAS::bayesglm.fit, estimate.logic.glm

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
```

```
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (X4$V50 * X4$V19 * X4$V13 * X4$V11) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8
  , sd = 1
)
X4$Y4 <- Y4

formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)

estimate.logic.lm(formula = formula1, data = X4, n = 1000, m = 50)
```

`estimate.speedglm` *Obtaining Bayesian estimators of interest from a GLM model*

Description

Obtaining Bayesian estimators of interest from a GLM model

Usage

```
estimate.speedglm(formula, data, family, prior, logn)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	distribution family foe the responses
`prior`	either "AIC" or "BIC"
`logn`	log sample size

Value

mlik marginal likelihood of the model

waic AIC model selection criterion

dic BIC model selection criterion

summary.fixed$mean a vector of posterior modes of the parameters

See Also

speedglm::speedglm.wfit

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (X4$V50 * X4$V19 * X4$V13 * X4$V11) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8
  , sd = 1
)
X4$Y4 <- Y4

formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)

estimate.logic.lm(formula = formula1, data = X4, n = 1000, m = 50)
```

`LogicRegr` *A wrapper for running the Bayesian logic regression based inference in a easy to use way*

Description

A wrapper for running the Bayesian logic regression based inference in a easy to use way

Usage

```
LogicRegr(
  formula,
  data,
  family = "Gaussian",
  prior = "J",
  report.level = 0.5,
  d = 20,
```

```
  cmax = 5,
  kmax = 20,
  p.and = 0.9,
  p.not = 0.05,
  p.surv = 0.1,
  ncores = -1,
  n.mods = 1000,
  print.freq = 1000L,
  advanced = list(presearch = TRUE, locstop = FALSE, estimator =
   estimate.logic.bern.tCCH, estimator.args = list(data = data.example, n = 1000, m =
   50, r = 1), recalc_margin = 250, save.beta = FALSE, interact = TRUE, relations =
   c("", "lgx2", "cos", "sigmoid", "tanh", "atan", "erf"), relations.prob = c(0.4, 0, 0,
    0, 0, 0, 0), interact.param = list(allow_offsprings = 1, mutation_rate = 300,
   last.mutation = 5000, max.tree.size = 1, Nvars.max = 100, p.allow.replace = 0.9,
    p.allow.tree = 0.2, p.nor = 0.2, p.and = 1),
     n.models = 10000, unique = TRUE,
   max.cpu = ncores, max.cpu.glob = ncores, create.table = FALSE, create.hash = TRUE,
    pseudo.paral = TRUE, burn.in = 50, outgraphs = FALSE, print.freq = print.freq,
   advanced.param = list(max.N.glob = as.integer(10), min.N.glob = as.integer(5), max.N
    = as.integer(3), min.N = as.integer(1), printable = FALSE))
)
```

Arguments

`formula`	a formula object for the model to be addressed
`data`	a data frame object containing variables and observations corresponding to the formula used
`family`	a string taking values of either "Gaussian" or "Bernoulli" corresponding to the linear or logistic Bayesian logic regression contexts
`prior`	character values "J" or "G" corresponding either to Jeffey's or robust g prior
`report.level`	a numeric value in (0,1) specifying the threshold for detections based on the marginal inclusion probabilities
`d`	population size for the GMJMCMC algorithm
`cmax`	the maximal allowed depth of logical expressions to be considered
`kmax`	the maximal number of logical expressions per model
`p.and`	probability of AND parameter of GMJMCMC algorithm
`p.not`	probability of applying logical NOT in GMJMCMC algorithm
`p.surv`	minimal survival probabilities for the features to be allowed to enter the next population
`ncores`	the maximal number of cores (and GMJMCMC threads) to be addressed in the analysis
`n.mods`	the number of the best models in the thread to calculate marginal inclusion probabilities
`print.freq`	printing frequency of the intermediate results

`advanced`	should only be addressed by experienced users to tune advanced parameters of GMJMCMC, advanced corresponds to the vector of tuning parameters of runemjmcmc function

Value

a list of

feat.stat detected logical expressions and their marginal inclusion probabilities

predictions NULL currently, since LogrRegr function is not designed for predictions at the moment, which is still possible in its expert mother function pinferunemjmcmc

allposteriors all visited by GMJMCMC logical expressions and their marginal inclusion probabilities

threads.stats a vector of detailed outputs of individual ncores threads of GMJMCMC run

See Also

runemjmcmc pinferunemjmcmc

Examples

```
set.seed(040590)
X1 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1,
    prob = runif(n = 50 * 1000, 0, 1)), dim = c(1000, 50)
  )
)
Y1 <- rnorm(
  n = 1000,
 mean = 1 + 0.7 * (X1$V1 * X1$V4) + 0.8896846 * (X1$V8 * X1$V11) + 1.434573 * (X1$V5 * X1$V9),
  sd = 1
)
X1$Y1 <- Y1

# specify the initial formula
formula1 <- as.formula(
  paste(colnames(X1)[51], "~ 1 +", paste0(colnames(X1)[-c(51)], collapse = "+"))
)
data.example <- as.data.frame(X1)


# run the inference with robust g prior
n_cores <- 1L


  res4G <- LogicRegr(
    formula = formula1, data = data.example, family = "Gaussian", prior = "G",
    report.level = 0.5, d = 15, cmax = 2, kmax = 15, p.and = 0.9, p.not = 0.01,
    p.surv = 0.2, ncores = n_cores
  )
```

```
print(res4G$feat.stat)

# run the inference with Jeffrey's prior
res4J <- LogicRegr(
  formula = formula1, data = data.example, family = "Gaussian", prior = "J",
  report.level = 0.5, d = 15, cmax = 2, kmax = 15, p.and = 0.9, p.not = 0.01,
  p.surv = 0.2, ncores = n_cores
)
print(res4J$feat.stat)
```

`m` *Product function used in the deep regression context*

Description

Product function used in the deep regression context

Usage

```
m(a, b)
```

Arguments

a	the first argument
b	the second argument

Value

`m(a,b)`, product of the arguments a*b

Examples

```
m(10,2)
```

`parall.gmj` *A function to run parallel chains of (R)(G)MJMCMC algorithms*

Description

A function to run parallel chains of (R)(G)MJMCMC algorithms

Usage

```
parall.gmj(X, M = 16, preschedule = FALSE)
```

Arguments

`X`	a vector of lists of parameters of runemjmcmc as well as several additional fields that must come after runemjmcmc parameters such as: **vect$simlen** the number of parameters of runemjmcmc in vect **vect$cpu** the CPU id for to set the unique seed **vect$NM** the number of unique best models from runemjmcmc to base the output report upon
`M`	a number of CPUs to be used (can only be equal to 1 on Windows OS currently, up to a maximal number of cores can be used on Linux-based systems)
`preschedule`	if pseudoscheduling should be used for the jobs if their number exceeds M (if TRUE) otherwise the jobs are performed sequentially w.r.t. their order

Value

a vector of lists of

post.populi the total mass (sum of the marginal likelihoods times the priors of the visited models) from the addressed run of runemjmcmc

p.post posterior probabilities of the covariates approximated by the addressed run of runemjmcmc

cterm the best value of marginal likelihood times the prior from the addressed run of runemjmcmc

fparam the final set of covariates returned by the addressed run of runemjmcmc

See Also

runemjmcmc parall.gmj

Examples

```
j <- 1
M <- 4
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (X4$V50 * X4$V19 * X4$V13 * X4$V11) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
```

```
X4$Y4 <- Y4

formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)
data.example <- as.data.frame(X4)

vect <- list(
  formula = formula1, outgraphs = FALSE, data = X4,
  estimator = estimate.logic.lm,
  estimator.args = list(data = data.example, n = 100, m = 50),
  recalc_margin = 249, save.beta = FALSE, interact = TRUE,
  relations = c("", "lgx2", "cos", "sigmoid", "tanh", "atan", "erf"),
  relations.prob = c(0.4, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0),
  interact.param = list(
    allow_offsprings = 1, mutation_rate = 250, last.mutation = 15000,
    max.tree.size = 4, Nvars.max = 40, p.allow.replace = 0.7,
    p.allow.tree = 0.2, p.nor = 0, p.and = 0.9
  ), n.models = 20000, unique = TRUE, max.cpu = 4, max.cpu.glob = 4,
  create.table = FALSE, create.hash = TRUE, pseudo.paral = TRUE,
  burn.in = 50, print.freq = 1000,
  advanced.param = list(
    max.N.glob = as.integer(10),
    min.N.glob = as.integer(5),
    max.N = as.integer(3),
    min.N = as.integer(1),
    printable = FALSE
  )
)

params <- list(vect)[rep(1, M)]

for (i in 1:M) {
  params[[i]]$cpu <- i
  params[[i]]$NM <- 1000
  params[[i]]$simlen <- 21
}

  message("begin simulation ", j)
  set.seed(363571)
  results <- parall.gmj(X = params, M = 1)
```

`parallelize`	*An example of user defined parallelization (cluster based) function for within an MJMCMC chain calculations (mclapply or lapply are used by default depending on specification and OS).*

Description

An example of user defined parallelization (cluster based) function for within an MJMCMC chain calculations (mclapply or lapply are used by default depending on specification and OS).

Usage

```
parallelize(X, FUN)
```

Arguments

`X`	a vector (atomic or list) or an expressions vector. Other objects (including classed objects) will be coerced by as.list
`FUN`	the function to be applied to each element of X or v, or in parallel to X

Details

Only allowed when working with big.memory based hash table within MJMCMC (see runemjmcmc for more details)

Value

`parallelize(X,FUN)`, a list of the same length as X and named by X

See Also

parLapply clusterMap mclapply lapply

`pinferunemjmcmc` *A wrapper for running the GLMM, BLR, or DBRM based inference and predictions in an expert but rather easy to use way*

Description

A wrapper for running the GLMM, BLR, or DBRM based inference and predictions in an expert but rather easy to use way

Usage

```
pinferunemjmcmc(
  n.cores = 4,
  mcgmj = mcgmjpse,
  report.level = 0.5,
  simplify = FALSE,
  num.mod.best = 1000,
  predict = FALSE,
  test.data = 1,
  link.function = function(z) z,
  runemjmcmc.params
)
```

Arguments

`n.cores`	the maximal number of cores (and (R)(G)MJMCMC threads) to be addressed in the analysis
`mcgmj`	an mclapply like function for performing for performing parallel computing, do not change the default unless you are using Windows
`report.level`	a numeric value in (0,1) specifying the threshold for detections based on the marginal inclusion probabilities
`simplify`	a logical value specifying in simplification of the features is to be done after the search is completed
`num.mod.best`	the number of the best models in the thread to calculate marginal inclusion probabilities
`predict`	a logical value specifying if predictions should be done by the run of pinferunemjmcmc
`test.data`	covariates data.frame to be used for predictions
`link.function`	the link functions to be used to make predictions
`runemjmcmc.params`	a vector of parameters of runemjmcmc function, see the help of runemjmcmc for details

Value

a list of

feat.stat detected features or logical expressions and their marginal inclusion probabilities

predictions predicted values if they are required, NULL otherwise

allposteriors all visited by (R)(G)MJMCMC features and logical expressions and their marginal inclusion probabilities

threads.stats a vector of detailed outputs of individual n.cores threads of (R)(G)MJMCMC run

See Also

runemjmcmc LogrRegr DeepRegr LinRegr

Examples

```
# inference

X <- read.csv(system.file("extdata", "exa1.csv", package="EMJMCMC"))
data.example <- as.data.frame(X)

# specify the initial formula
formula1 <- as.formula(
  paste(colnames(X)[5], "~ 1 +", paste0(colnames(X)[-5], collapse = "+"))
)

# define the number or cpus
```

```
M <- 1L
# define the size of the simulated samples
NM <- 1000
# define \k_{max} + 1 from the paper
compmax <- 16
# define treshold for preinclusion of the tree into the analysis
th <- (10)^(-5)
# define a final treshold on the posterior marginal probability for reporting a
# tree
thf <- 0.05
# specify tuning parameters of the algorithm for exploring DBRM of interest
# notice that allow_offsprings=3 corresponds to the GMJMCMC runs and
# allow_offsprings=4 -to the RGMJMCMC runs

  res1 <- pinferunemjmcmc(
    n.cores = M, report.level = 0.5, num.mod.best = NM, simplify = TRUE,
    runemjmcmc.params = list(
      formula = formula1, data = data.example, estimator = estimate.gamma.cpen_2,
      estimator.args = list(data = data.example), recalc_margin = 249,
      save.beta = FALSE, interact = TRUE, outgraphs = FALSE,
      relations = c("to23", "expi", "logi", "to35", "sini", "troot", "sigmoid"),
      relations.prob = c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1),
      interact.param = list(allow_offsprings = 3, mutation_rate = 250,
      last.mutation = 10000, max.tree.size = 5, Nvars.max = 15,
      p.allow.replace = 0.9, p.allow.tree = 0.01, p.nor = 0.9, p.and = 0.9),
      n.models = 10000, unique = TRUE, max.cpu = M, max.cpu.glob = M,
      create.table = FALSE, create.hash = TRUE, pseudo.paral = TRUE,
      burn.in = 100, print.freq = 1000,
      advanced.param = list(
        max.N.glob = as.integer(10),
        min.N.glob = as.integer(5),
        max.N = as.integer(3),
        min.N = as.integer(1),
        printable = FALSE
      )
    )
  )
  print(res1$feat.stat)


# prediction

compmax <- 21

# read in the train and test data sets
test <- read.csv(
  system.file("extdata", "breast_cancer_test.csv", package="EMJMCMC"),
  header = TRUE, sep = ","
)[, -1]
train <- read.csv(
  system.file("extdata", "breast_cancer_train.csv", package="EMJMCMC"),
  header = TRUE, sep = ","
)[, -1]
```

```
# transform the train data set to a data.example data.frame that EMJMCMC class
# will internally use
data.example <- as.data.frame(train)

# specify the link function that will be used in the prediction phase
g <- function(x) {
  return((x <- 1 / (1 + exp(-x))))
}

formula1 <- as.formula(
  paste(
    colnames(data.example)[31], "~ 1 +",
    paste0(colnames(data.example)[-31], collapse = "+")
  )
)


  # Defining a custom estimator function
  estimate.bas.glm.cpen <- function(
    formula, data, family, prior, logn, r = 0.1, yid=1,
    relat =c("cosi","sigmoid","tanh","atan","erf","m(")
  ) {
    #only poisson and binomial families are currently adopted
    X <- model.matrix(object = formula,data = data)
   capture.output({out <- BAS::bayesglm.fit(x = X, y = data[,yid], family=family,coefprior=prior)})
    fmla.proc<-as.character(formula)[2:3]
    fobserved <- fmla.proc[1]
   fmla.proc[2]<- stringi::stri_replace_all(str = fmla.proc[2],fixed = " ",replacement = "")
   fmla.proc[2]<- stringi::stri_replace_all(str = fmla.proc[2],fixed = "\n",replacement = "")
    sj<-2*(stringi::stri_count_fixed(str = fmla.proc[2], pattern = "*"))
    sj<-sj+1*(stringi::stri_count_fixed(str = fmla.proc[2], pattern = "+"))
    for(rel in relat) {
      sj<-sj+2*(stringi::stri_count_fixed(str = fmla.proc[2], pattern = rel))
    }
    mlik = ((-out$deviance +2*log(r)*sum(sj)))/2
    return(
      list(
        mlik = mlik, waic = -(out$deviance + 2*out$rank),
        dic = -(out$deviance + logn*out$rank),
        summary.fixed = list(mean = coefficients(out))
      )
    )
  }
  res <- pinferunemjmcmc(
    n.cores = M, report.level = 0.5, num.mod.best = NM, simplify = TRUE,
    predict = TRUE, test.data = as.data.frame(test), link.function = g,
    runemjmcmc.params = list(
      formula = formula1, data = data.example, gen.prob = c(1, 1, 1, 1, 0),
      estimator = estimate.bas.glm.cpen,
      estimator.args = list(
        data = data.example, prior = BAS::aic.prior(), family = binomial(),
        yid = 31, logn = log(143), r = exp(-0.5)
```

```
    ), recalc_margin = 95, save.beta = TRUE, interact = TRUE,
    relations = c("gauss", "tanh", "atan", "sin"),
    relations.prob = c(0.1, 0.1, 0.1, 0.1),
    interact.param = list(
      allow_offsprings = 4, mutation_rate = 100, last.mutation = 1000,
      max.tree.size = 6, Nvars.max = 20, p.allow.replace = 0.5,
      p.allow.tree = 0.4, p.nor = 0.3, p.and = 0.9
    ), n.models = 7000, unique = TRUE, max.cpu = M, max.cpu.glob = M,
    create.table = FALSE, create.hash = TRUE, pseudo.paral = TRUE,
    burn.in = 100, print.freq = 1000,
    advanced.param = list(
      max.N.glob = as.integer(10), min.N.glob = as.integer(5),
      max.N = as.integer(3), min.N = as.integer(1), printable = FALSE
    )
  )
)

for (jjjj in 1:10)
{
  resw <- as.integer(res$predictions >= 0.1 * jjjj)
  prec <- (1 - sum(abs(resw - test$X), na.rm = TRUE) / length(resw))
  print(prec)
  # FNR
  ps <- which(test$X == 1)
 fnr <- sum(abs(resw[ps] - test$X[ps])) / (sum(abs(resw[ps] - test$X[ps])) + length(ps))

  # FPR
  ns <- which(test$X == 0)
 fpr <- sum(abs(resw[ns] - test$X[ns])) / (sum(abs(resw[ns] - test$X[ns])) + length(ns))
}
```

---

runemjmcmc *Mode jumping MJMCMC or Genetically Modified Mode jumping MCMC or Reversible Genetically Modified Mode jumping MCMC for variable selection, Bayesian model averaging and feature engineering*

Description

A function that creates an EMJMCMC2016 object with specified values of some parameters and default values of other parameters.

Usage

```
runemjmcmc(
  formula,
  data,
  secondary = vector(mode = "character", length = 0),
  latnames = "",
```

```
  estimator,
  estimator.args = "list",
  n.models,
  p.add.default = 1,
  p.add = 0.5,
  unique = FALSE,
  save.beta = FALSE,
  locstop.nd = FALSE,
  latent = "",
  max.cpu = 4,
  max.cpu.glob = 2,
  create.table = TRUE,
  hash.length = 20,
  presearch = TRUE,
  locstop = FALSE,
  pseudo.paral = FALSE,
  interact = FALSE,
  deep.method = 1,
  relations = c("", "sin", "cos", "sigmoid", "tanh", "atan", "erf"),
  relations.prob = c(0.4, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1),
  gen.prob = c(1, 10, 5, 1, 0),
  pool.cross = 0.9,
  p.epsilon = 1e-04,
  del.sigma = 0.5,
  pool.cor.prob = FALSE,
  interact.param = list(allow_offsprings = 2, mutation_rate = 100, last.mutation = 2000,
    max.tree.size = 10000, Nvars.max = 100, p.allow.replace = 0.7, p.allow.tree = 0.1,
    p.nor = 0.3, p.and = 0.7),
  prand = 0.01,
  keep.origin = TRUE,
  sup.large.n = 5000,
  recalc_margin = 2^10,
  create.hash = FALSE,
  interact.order = 1,
  burn.in = 1,
  eps = 10^6,
  max.time = 120,
  max.it = 25000,
  print.freq = 100,
  outgraphs = FALSE,
  advanced.param = NULL,
  distrib_of_neighbourhoods = t(array(data = c(7.6651604, 16.773326, 14.541629,
   12.839445, 2.964227, 13.048343, 7.165434, 0.9936905, 15.94249, 11.040131, 3.200394,
   15.349051, 5.466632, 14.676458, 1.5184551, 9.285762, 6.125034, 3.627547, 13.343413,
   2.923767, 15.318774, 14.529538, 1.52196, 11.804457, 5.070282, 6.93438, 10.578945,
    12.455602, 6.0826035, 2.453729, 14.340435, 14.863495, 1.028312, 12.685017,
    13.806295), dim = c(7, 5))),
  distrib_of_proposals = c(76.9187, 71.25264, 87.68184, 60.55921, 15812.39852),
```

```
  quiet = TRUE
)
```

Arguments

`formula`	a typical formula for specifying a model with all potential covariates included
`data`	a data frame containing both covariates and response
`secondary`	a character vector of names other covariates excluded from those defined in formula (relevant for GMJMCMC only)
`latnames`	a character vector of names other covariates excluded from populations of GMJMCMC, for example for continuous covariates to be combined with BLR (relevant for GMJMCMC only) or the names of latent Gaussian variables to be selected in BGNLMM
`estimator`	a function returning a list with marginal likelihood, waic, dic and coefficients of the addressed model. The list should be of a format: list(mlik = mlik,waic = waic , dic = dic,summary.fixed =list(mean = coefficients))
`estimator.args`	a list of arguments of estimator functions to be used (formula parameter has to be omitted, see the example)
`n.models`	maximal number of models to be estimated during the search
`p.add.default`	a parameter defining sparsity after filtrations in GMJMCMC as initial marginal inclusion probabilities vector for parameters in the current pool
`p.add`	a default marginal inclusion probability parameter to be changed during the search to the true value
`unique`	defines whether n.models allows repetitions of the same models (unique=FALSE) or not (unique=TRUE)
`save.beta`	a boolean parameter defining if beta coefficients for the models should be stored (must be set to TRUE if one is interested in predictions)
`locstop.nd`	Defines whether local greedy optimizers stop at the first local optima found (locstop.nd=TRUE) or not (locstop.nd=FALSE)
`latent`	a latent random field to be addressed (to be specifically used when estimator = INLA, currently unsupported)
`max.cpu`	maximal number of CPUs in MJMCMC when within chain parallelization is allowed pseudo.paral = FALSE
`max.cpu.glob`	maximal number of CPUs in global moves in MJMCMC when within chain parallelization is allowed pseudo.paral = FALSE
`create.table`	a Boolean variable defining if a big.memory based hash table (only available for MJMCMC with no feature engineering, allows data sharing between CPUs) or the original R hash data structure (available for all algorithm, does not allow data sharing between CPUs) is used for storing of the results
`hash.length`	a parameter defining hash size for the big.memory based hash table as 2^hash.length (only relevant when create.table = TRUE)
`presearch`	a boolean parameter defining if greedy forward and backward regression steps are used for initialization of initial approximations of marginal inclusion probabilities

`locstop`	a boolean parameter defining if the presearch is stopped at the first local extremum visited
`pseudo.paral`	defines if lapply or mclapply is used for local vectorized computations within the chain (can only be TRUE if create.table= TRUE)
`interact`	a boolean parameter defining if feature engineering is allowed in the search
`deep.method`	an integer in {1, 2, 3, 4} defining the method of estimating the alpha parameters of BGNLM, details to be found in https://www.jair.org/index.php/jair/article/view/13047
`relations`	a vector of allowed modification functions (only relevant when feature engineering is enabled by means of interact = TRUE)
`relations.prob`	probability distribution of addressing modifications defined in relations parameter (both vectors must be of the same length)
`gen.prob`	a vector of probabilities for different operators in GMJMCMC or RGMJMCMC in the deep regression context (hence only relevant if `interact.param$allow_offsprings` is either 3 or 4)
`pool.cross`	a parameter defining the probability of addressing covariates from the current pool of covariates in GMJMCMC (covariates from the set of filtered covariates can be addressed with probability 1-pool.cross) (only relevant when interact = TRUE)
`p.epsilon`	a parameter to define minimal deviations from 0 and 1 probabilities when allowing adaptive MCMC based on marginal inclusion probabilities
`del.sigma`	a parameter describing probability of deleting each of the function from the selected feature in the reduction operator(only relevant for the deep regression models context)
`pool.cor.prob`	a boolean parameter indicating if inclusion of the filtered covariates during mutations are based on probabilities proportional to the absolute values of correlations of these parameters and the observations (should not be addressed for multivariate observations, e.g. survival studies with Cox regression)
`interact.param`	a list of parameters for GMJMCMC, where allow_offsprings is 1 for logic regression context, 2 for the old version of GMJMCMC for deep regressions, 3 for the new version of GMJMCMC for deep regressions and 4 for the RGMJMCMC for the deep regressions; mutation_rate defines how often changes of the search space are allowed in terms of the number of MJMCMC iterations per search space; last.mutation defines the iteration after which changes of search space are no longer allowed; max.tree.size is a parameter defining maximal depth of features; Nvars.max is a parameter defining maximal number of covariates in the search space after the first filtration; p.allow.replace is a parameter defining the upper bound on the probability allowing the replacement of corresponding features with marginal inclusion probabilities below it; p.allow.tree is a lower bound for the probability of not being filtered out after initializing steps of MJMCMC in GMJMCMC; p.nor is a parameter for not operator in the logic regression context (allow_offsprings==1); p.and = is the probability of & crossover in the logic regression context (allow_offsprings==1)
`prand`	probability of changes of components in randomization kernels of RGMJMCMC

`keep.origin`	a boolean parameter defining if the initially unfiltered covariates can leave the search space afterwards (TRUE) or not (FALSE)
`sup.large.n`	omitted currently
`recalc_margin`	a parameter defining how often marginal inclusion probabilities would be recalculated
`create.hash`	a parameter defining if by default the results are stored in a hash table
`interact.order`	omitted currently
`burn.in`	number of burn-in steps for (R)(G)MJMCMC
`eps`	omitted, not to be changed
`max.time`	maximal time for the run of (R)(G)MJMCMC algorithm in minutes
`max.it`	maximal number of (R)(G)MJMCMC iterations
`print.freq`	printing frequency of the intermediate results
`outgraphs`	a boolean variable defining if the graphics on the marginal inclusion probabilities should be drawn (must not be used inside mclapply wrapper of runemjmcmc since otherwise errors can occur)
`advanced.param`	omitted currently
`distrib_of_neighbourhoods`	a matrix defining probability distribution on 7 types of neighbourhoods within 4 possible local search strategies as well as within global moves
`distrib_of_proposals`	probability distribution up to a constant of proportionality for addressing different local search strategies after large jumps or no large jumps (5th component)
`quiet`	defaults to `FALSE`. If `TRUE`, prints intermediate messages

Details

The algorithm is an extended Metropolis-Hastings algorithm (or its Genetically modified version) mixing single site changes with occasionally large jumps. The models are described through the gamma vector, a binary vector indicating which variables that are included in the model.

See Hubin & Storvik (2016),Hubin, Storvik & Frommlet (2017), Hubin & Storvik (2017) details. The local optimization is performed through stepwise search within a neighborhood in the current gamma vector, allowing one component to be changed at a time.

Value

a list containing

p.post a vector of posterior probabilities of the final vector of active covariates (features)

m.post a vector of posterior probabilities of the models from the search space induced by the final vector of active covariates (features)

s.mass sum of marginal likelihoods times the priors from the explored part of the search space induced by the final vector of active covariates (features)

Author(s)

Aliaksandr Hubin

References

Hubin & Storvik (2016),Hubin, Storvik & Frommlet (2017), Hubin & Storvik (2017)

See Also

global objects statistics1 (if create.table== TRUE) or hashStat (if create.table== FALSE) contain all marginal likelihoods and two other model selection criteria as well as all of the beta coefficients for the models (if save.beta== TRUE)

Examples

```
X4 <- as.data.frame(
  array(
    data = rbinom(n = 50 * 1000, size = 1, prob = runif(n = 50 * 1000, 0, 1)),
    dim = c(1000, 50)
  )
)
Y4 <- rnorm(
  n = 1000,
  mean = 1 +
    7 * (X4$V4 * X4$V17 * X4$V30 * X4$V10) +
    7 * (((X4$V50 * X4$V19 * X4$V13 * X4$V11) > 0)) +
    9 * (X4$V37 * X4$V20 * X4$V12) +
    7 * (X4$V1 * X4$V27 * X4$V3) +
    3.5 * (X4$V9 * X4$V2) +
    6.6 * (X4$V21 * X4$V18) +
    1.5 * X4$V7 +
    1.5 * X4$V8,
  sd = 1
)
X4$Y4 <- Y4
data.example <- as.data.frame(X4)

# specify the initial formula
formula1 <- as.formula(
  paste(colnames(X4)[51], "~ 1 +", paste0(colnames(X4)[-c(51)], collapse = "+"))
)


# specify tuning parameters of the algorithm for exploring DBRM of interest
# notice that allow_offsprings=3 corresponds to the GMJMCMC runs and
# allow_offsprings=4 -to the RGMJMCMC runs

  res <- runemjmcmc(
    formula = formula1, outgraphs = FALSE, data = X4,
    estimator = estimate.gamma.cpen, estimator.args = list(data = data.example),
    recalc_margin = 249, save.beta = FALSE, interact = TRUE,
    relations = c("cos", "sigmoid", "tanh", "atan", "sin", "erf"),
```

```
    relations.prob = c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1),
    interact.param = list(
      allow_offsprings = 4, mutation_rate = 250, last.mutation = 15000,
      max.tree.size = 4, Nvars.max = 40, p.allow.replace = 0.7,
      p.allow.tree = 0.2, p.nor = 0, p.and = 0.9
    ), n.models = 20000, unique = TRUE, max.cpu = 4, max.cpu.glob = 4,
    create.table = FALSE, create.hash = TRUE, pseudo.paral = TRUE, burn.in = 50,
    print.freq = 1000,
    advanced.param = list(
      max.N.glob = as.integer(10),
      min.N.glob = as.integer(5),
      max.N = as.integer(3),
      min.N = as.integer(1),
      printable = FALSE
    )
  )
```

`sigmoid` *sigmoid activation function*

Description

sigmoid activation function

Usage

```
sigmoid(x)
```

Arguments

`x`	a real number

Value

sigmoid value

Examples

```
sigmoid(10)
```

`simplify.formula` *A function parsing the formula into the vectors of character arrays of responses and covariates*

Description

A function parsing the formula into the vectors of character arrays of responses and covariates

Usage

```
simplify.formula(fmla, names)
```

Arguments

`fmla`	an R formula object
`names`	all column names from the data.frame to be used with the formula

Value

a list of

fobserved a vector of character arrays corresponding to the observations

fparam a vector of character arrays corresponding to the covariates

See Also

formula data.frame

Examples

```
X1 <- as.data.frame(
  array(data = rbinom(n = 50 * 1000, size = 1, prob = 0.3), dim = c(1000, 50))
)
Y1 <- -0.7 + 1 * ((1 - X1$V1) * (X1$V4)) + 1 * (X1$V8 * X1$V11) + 1 * (X1$V5 * X1$V9)
X1$Y1 <- round(1.0 / (1.0 + exp(-Y1)))

formula1 <- as.formula(
  paste(colnames(X1)[51], "~ 1 +", paste0(colnames(X1)[-c(51)], collapse = "+"))
)
names <- colnames(X1)
simplify.formula(fmla = formula1, names = names)
```

`simplifyposteriors` *A function that ads up posteriors for the same expression written in different character form in different parallel runs of the algorithm (mainly for Logic Regression and Deep Regression contexts)*

Description

A function that ads up posteriors for the same expression written in different character form in different parallel runs of the algorithm (mainly for Logic Regression and Deep Regression contexts)

Usage

```
simplifyposteriors(X, posteriors, th = 1e-04, thf = 0.2, resp)
```

Arguments

`X`	a data.frame containing the data on the covariates
`posteriors`	a data.frame with expressions in the first column and their posteriors in the second column from all of the runs
`th`	initial filtering before summary threshold
`thf`	threshold for final filtering after summary
`resp`	the response to be addressed

Value

res, a data.frame with the summarized across runs expressions and their posteriors

See Also

runemjmcmc

`truncfactorial` *Truncated factorial to avoid stack overflow for huge values*

Description

truncated factorial to avoid stack overflow for huge values

Usage

```
truncfactorial(x)
```

Arguments

`x`	a non-negative integer number

Value

`truncfactorial(x)`, truncated factorial as min(x!,171!)

Examples

```
truncfactorial(10)
```

Index

∗ **methods**
  - do.call.emjmcmc, 2
  - erf, 3
  - estimate.bas.glm, 4
  - estimate.bas.lm, 5
  - estimate.bigm, 6
  - estimate.elnet, 8
  - estimate.glm, 10
  - estimate.logic.glm, 12
  - estimate.logic.lm, 13
  - estimate.speedglm, 14
  - LogicRegr, 15
  - m, 18
  - parall.gmj, 18
  - parallelize, 20
  - pinferunemjmcmc, 21
  - runemjmcmc, 25
  - sigmoid, 31
  - simplify.formula, 32
  - simplifyposteriors, 33
  - truncfactorial, 33

∗ **models**
  - do.call.emjmcmc, 2
  - erf, 3
  - estimate.bas.glm, 4
  - estimate.bas.lm, 5
  - estimate.bigm, 6
  - estimate.elnet, 8
  - estimate.glm, 10
  - estimate.logic.glm, 12
  - estimate.logic.lm, 13
  - estimate.speedglm, 14
  - LogicRegr, 15
  - m, 18
  - parall.gmj, 18
  - parallelize, 20
  - pinferunemjmcmc, 21
  - runemjmcmc, 25
  - sigmoid, 31
  - simplify.formula, 32
  - simplifyposteriors, 33
  - truncfactorial, 33

do.call.emjmcmc, 2

erf, 3
estimate.bas.glm, 4
estimate.bas.lm, 5
estimate.bigm, 6
estimate.elnet, 8
estimate.gamma.cpen, 9
estimate.gamma.cpen_2, 9
estimate.glm, 10
estimate.logic.glm, 12
estimate.logic.lm, 13
estimate.speedglm, 14

LogicRegr, 15

m, 18

parall.gmj, 18
parallelize, 20
pinferunemjmcmc, 21

runemjmcmc, 25

sigmoid, 31
simplify.formula, 32
simplifyposteriors, 33

truncfactorial, 33