Package 'MGLM'

October 12, 2022

Version 0.2.1

Title Multivariate Response Generalized Linear Models

Author Yiwen Zhang <zhangyiwen1015@gmail.com> and Hua Zhou <huazhou@ucla.edu>

Maintainer Juhyun Kim <juhkim111@ucla.edu>

Depends R (>= 3.0.0)

Imports methods, stats, parallel, stats4

Suggests ggplot2, plyr, reshape2, knitr, testthat (>= 3.0.0)

Description Provides functions that (1) fit multivariate discrete distributions, (2) generate random numbers from multivariate discrete distributions, and (3) run regression and penalized regression on the multivariate categorical response data. Implemented models include: multinomial logit model, Dirichlet multinomial model, generalized Dirichlet multinomial model, and negative multinomial model. Making the best of the minorization-maximization (MM) algorithm and Newton-Raphson method, we derive and implement stable and efficient algorithms to find the maximum likelihood estimates. On a multi-core machine, multi-threading is supported.

VignetteBuilder knitr

LazyLoad yes

LazyData yes

Repository CRAN

License GPL (>= 2)

RoxygenNote 7.1.2

NeedsCompilation no

Date/Publication 2022-04-13 23:32:32 UTC

R topics documented:

`MGLM-package` *MGLM: A package for multivariate response generalized linear models*

Description

The package provides functions that (1) fit multivariate discrete distributions, (2) generate random numbers from multivariate discrete distributions, and (3) run regression and penalized regression on the multivariate categorical response data. Implemented models include: multinomial logit model, Dirichlet multinomial model, generalized Dirichlet multinomial model, and negative multinomial model. Making the best of the minorization-maximization (MM) algorithm and Newton-Raphson method, we derive and implement stable and efficient algorithms to find the maximum likelihood estimates. On a multi-core machine, multi-threading is supported.

Details

Package:	MGLM
Type:	Package
Version:	0.0.9
Date:	2017-12-14
License:	GPL (>= 2)
Depends:	R (>= 3.0.0), methods, stats, parallel

Author(s)

Yiwen Zhang and Hua Zhou

`AIC` *Akaike's Information Criterion (AIC)*

Description

Calculates the Akaike's information criterion (AIC) for a fitted model object.

Usage

```
## S4 method for signature 'MGLMfit'
AIC(object)

## S4 method for signature 'MGLMreg'
AIC(object)

## S4 method for signature 'MGLMsparsereg'
AIC(object)

## S4 method for signature 'MGLMtune'
AIC(object)
```

Arguments

`object`	MGLM object. `"MGLMfit"`, `"MGLMreg"`, `"MGLMsparsereg"`, or `"MGLMtune"`

Value

Returns a numeric value with the corresponding AIC.

For the class `"MGLMtune"`, the function returns AIC based on the optimal tuning parameter.

Examples

```
set.seed(124)
n <- 200
d <- 4
alpha <- rep(1, d-1)
beta <- rep(1, d-1)
m <- 50
Y <- rgdirmn(n, m, alpha, beta)
gdmFit <- MGLMfit(Y, dist="GDM")
AIC(gdmFit)
```

BIC *Bayesian information criterion (BIC)*

Description

Calculates the Bayesian information criterion (BIC) for a fitted model object.

Usage

```
## S4 method for signature 'MGLMfit'
BIC(object)

## S4 method for signature 'MGLMreg'
BIC(object)

## S4 method for signature 'MGLMsparsereg'
BIC(object)

## S4 method for signature 'MGLMtune'
BIC(object)
```

Arguments

`object`	MGLM object. "`MGLMfit`", "`MGLMreg`", "`MGLMsparsereg`", or "`MGLMtune`"

Value

Returns a numeric value with the corresponding BIC.

For the class "`MGLMtune`", the function returns BIC based on the optimal tuning parameter.

Examples

```
set.seed(124)
n <- 200
d <- 4
alpha <- rep(1, d-1)
beta <- rep(1, d-1)
m <- 50
Y <- rgdirmn(n, m, alpha, beta)
gdmFit <- MGLMfit(Y, dist="GDM")
BIC(gdmFit)
```

coef — *Extract Model Coefficients*

Description

`coef` extracts estimated model coefficients of class. `coefficients` is an *alias* for it.

Usage

```
## S4 method for signature 'MGLMfit'
coef(object)

## S4 method for signature 'MGLMreg'
coef(object)

## S4 method for signature 'MGLMsparsereg'
coef(object)

## S4 method for signature 'MGLMtune'
coef(object)
```

Arguments

`object`	an object for which the extraction of model coefficients is meaningful. One of the following classes "`MGLMfit`", "`MGLMreg`", "`MGLMsparsereg`", "`MGLMtune`"

Details

Method coef.

Value

Coefficients extracted from the model object `object`.

For the class "`MGLMtune`", the function returns model coefficients based on the optimal tuning parameter.

Examples

```
library("MGLM")
data("rnaseq")
data <- rnaseq[, 1:6]
mnreg <- MGLMreg(formula = cbind(X1, X2, X3, X4, X5, X6) ~ log(totalReads) +
treatment + age + gender, data = rnaseq, dist = "MN")
coef(mnreg)
```

dist *Details of the distributions*

Description

An object that specifies the distribution to be fitted by the `MGLMfit` function, or the regression model to be fitted by the `MGLMreg` or `MGLMsparsereg` functions. Can be chosen from `"MN"`, `"DM"`, `"NegMN"`, or `"GDM"`.

Details

"MN": Multinomial distribution: A multinomial distribution models the counts of d possible outcomes. The counts of categories are negatively correlated. The density of a d category count vector y with parameter $p = (p_1, \ldots, p_d)$ is

$$P(y|p) = C^m_{y_1,\ldots,y_d} \prod_{j=1}^{d} p_j^{y_j},$$

where $m = \sum_{j=1}^{d} y_j$, $0 < p_j < 1$, and $\sum_{j=1}^{d} p_j = 1$. Here, C_k^n, often read as "n choose k", refers the number of k combinations from a set of n elements.

The `MGLMreg` function with `dist="MN"` calculates the MLE of regression coefficients β_j of the multinomial logit model, which has link function $p_j = exp(X\beta_j)/(1 + \sum_{j=1}^{d-1} exp(X\beta_j))$, $j = 1, \ldots, d-1$. The `MGLMsparsereg` function with `dist="MN"` fits regularized multinomial logit model.

"DM": Dirichlet multinomial distribution:

When the multivariate count data exhibits over-dispersion, the traditional multinomial model is insufficient. Dirichlet multinomial distribution models the probabilities of the categories by a Dirichlet distribution. The density of a d category count vector y, with parameter $\alpha = (\alpha_1, \ldots, \alpha_d)$, $\alpha_j > 0$, is

$$P(y|\alpha) = C^m_{y_1,\ldots,y_d} \prod_{j=1}^{d} \frac{\Gamma(\alpha_j + y_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\sum_{j'=1}^{d} \alpha_{j'})}{\Gamma(\sum_{j'=1}^{d} \alpha_{j'} + \sum_{j'=1}^{d} y_{j'})},$$

where $m = \sum_{j=1}^{d} y_j$. Here, C_k^n, often read as "n choose k", refers the number of k combinations from a set of n elements.

The `MGLMfit` function with `dist="DM"` calculates the maximum likelihood estimate (MLE) of $(\alpha_1, \ldots, \alpha_d)$. The `MGLMreg` function with `dist="DM"` calculates the MLE of regression coefficients β_j of the Dirichlet multinomial regression model, which has link function $\alpha_j = exp(X\beta_j)$, $j = 1, \ldots, d$. The `MGLMsparsereg` function with `dist="DM"` fits regularized Dirichlet multinomial regression model.

"GDM": Generalized Dirichlet multinomial distribution:

The more flexible Generalized Dirichlet multinomial model can be used when the counts of categories have both positive and negative correlations. The probability mass of a count vector y over

m trials with parameter $(\alpha, \beta) = (\alpha_1, \ldots, \alpha_{d-1}, \beta_1, \ldots, \beta_{d-1})$, $\alpha_j, \beta_j > 0$, is

$$P(y|\alpha, \beta) = C^m_{y_1,\ldots,y_d} \prod_{j=1}^{d-1} \frac{\Gamma(\alpha_j + y_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\beta_j + z_{j+1})}{\Gamma(\beta_j)} \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j + \beta_j + z_j)},$$

where $z_j = \sum_{k=j}^d y_k$ and $m = \sum_{j=1}^d y_j$. Here, C_k^n, often read as "n choose k", #' refers the number of k combinations from a set of n elements.

The `MGLMfit` with `dist="GDM"` calculates the MLE of $(\alpha, \beta) = (\alpha_1, \ldots, \alpha_{d-1}, \beta_1, \ldots, \beta_{d-1})$. The `MGLMreg` function with `dist="GDM"` calculates the MLE of regression coefficients α_j, β_j of the generalized Dirichlet multinomial regression model, which has link functions $\alpha_j = exp(X\alpha_j)$ and $\beta_j = exp(X\beta_j)$, $j = 1, \ldots, d-1$. The `MGLMsparsereg` function with `dist="GDM"` fits regularized generalized Dirichlet multinomial regression model.

"NegMN": Negative multinomial distribution:

Both the multinomial distribution and Dirichlet multinomial distribution are good for negatively correlated counts. When the counts of categories are positively correlated, the negative multinomial distribution is preferred. The probability mass function of a d category count vector y with parameter $(p_1, \ldots, p_{d+1}, \beta)$, $\sum_{j=1}^{d+1} p_j = 1$, $p_j > 0$, $\beta > 0$, is

$$P(y|p, \beta) = C_m^{\beta+m-1} C^m_{y_1,\ldots,y_d} \prod_{j=1}^d p_j^{y_j} p_{d+1}^\beta = \frac{\beta_m}{m!} C^m_{y_1,\ldots,y_d} \prod_{j=1}^d p_j^{y_j} p_{d+1}^\beta,$$

where $m = \sum_{j=1}^d y_j$. Here, C_k^n, often read as "n choose k", refers the number of k combinations from a set of n elements.

The `MGLMfit` function with `dist="NegMN"` calculates the MLE of $(p_1, \ldots, p_{d+1}, \beta)$. The `MGLMreg` function with `dist="NegMN"` and `regBeta=FALSE` calculates the MLE of regression coefficients $(\alpha_1, \ldots, \alpha_d, \beta)$ of the negative multinomial regression model, which has link function $p_{d+1} = 1/(1 + \sum_{j=1}^d exp(X\alpha_j))$, $p_j = exp(X\alpha_j)p_{d+1}$, $j = 1, \ldots, d$. When `dist="NegMN"` and `regBeta=TRUE`, the overdispersion parameter is linked to covariates via $\beta = exp(X\alpha_{d+1})$, and the function `MGLMreg` outputs an estimated matrix of $(\alpha_1, \ldots, \alpha_{d+1})$. The `MGLMsparsereg` function with `dist="NegMN"` fits regularized negative multinomial regression model.

Author(s)

Yiwen Zhang and Hua Zhou

See Also

`MGLMfit`, `MGLMreg`, `MGLMsparsereg`, `dmn`, `ddirmn`, `dgdirmn`, `dnegmn`

`DMD.DM.fit` *Fit multivariate discrete distributions*

Description

Fit the specified multivariate discrete distribution.

Usage

```
DMD.DM.fit(
  data,
  init,
  weight,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE
)

DMD.GDM.fit(
  data,
  init,
  weight,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE
)

DMD.NegMN.fit(
  data,
  init,
  weight,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE
)

MGLMfit(
  data,
  dist,
  init,
  weight,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE
)
```

Arguments

`data`	a data frame or matrix containing the count data. Rows of the matrix represent observations and columns are the categories. Rows and columns of all zeros are automatically removed.
`init`	an optional vector of initial value of the parameter estimates. Should have the same dimension as the estimated parameters. See `dist` for details.
`weight`	an optional vector of weights assigned to each row of the data. Should be Null or a numeric vector with the length equal to the number of rows of `data`. If `weight=NULL`, equal weights of all ones will be assigned.

`epsilon`	an optional numeric controlling the stopping criterion. The algorithm terminates when the relative change in the log-likelihoods of two successive iterates is less than `epsilon`. The default value is `epsilon=1e-8`.
`maxiters`	an optional number controlling the maximum number of iterations. The default value is `maxiters=150`.
`display`	an optional logical variable controlling the display of iterations. The default value is FALSE.
`dist`	a description of the distribution to fit. Choose from `"MN"`, `"DM"`, `"GDM"`, `"NegMN"`. See `dist` for details.

Details

See `dist` for details about model parameterization.

Value

Returns an object of S4 class `"MGLMfit"`. An object of class `"MGLMfit"` is a list containing at least the following components:

- `estimate` the vector of the distribution prameter estimates.
- `SE` the vector of standard errors of the estimates.
- `vcov` the variance-covariance matrix of the estimates.
- `logL` the loglikelihood value.
- `iter` the number of iterations used.
- `BIC` Bayesian information criterion.
- `AIC` Akaike information criterion.
- `distribution` the distribution fitted.
- `LRT` when `dist="DM"` or `"GDM"`, it is the likelihood ratio test statistic for comparing the current model to the multinomial model. No LRT provided when `dist="NegMN"`.
- `LRTpvalue` the likelihood ratio test P value.
- `gradient` the gradient at the estimated parameter values.
- `DoF` the degrees of freedom of the model.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
data(rnaseq)
Y <- as.matrix(rnaseq[, 1:6])
fit <- MGLMfit(data=Y, dist="GDM")
```

dof *Extract degrees of freedom*

Description

dof extracts the degrees of freedom of the estimated parameter from the object of class MGLMsparsereg.

Usage

```
## S4 method for signature 'MGLMsparsereg'
dof(object)
```

Arguments

object	an object of class MGLMsparsereg

Value

Returns degrees of freedom of object.

Examples

```
library("MGLM")
dist <- "DM"
n <- 100
p <- 10
d <- 5
set.seed(118)
m <- rbinom(n, 200, 0.8)
X <- matrix(rnorm(n * p), n, p)
alpha <- matrix(0, p, d)
alpha[c(1, 3, 5), ] <- 1
Alpha <- exp(X %*% alpha)
Y <- rdirmn(size = m, alpha = Alpha)
pen <- "group"
ngridpt <- 30
spmodelfit <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
                            lambda = Inf, penalty = pen)
df <- dof(spmodelfit)
```

kr *Khatri-Rao product of two matrices*

Description

Return the Khatri-Rao product of two matrices, which is a column-wise Kronecker product.

Usage

```
kr(A, B, w, byrow = TRUE)
```

Arguments

`A, B`	matrices. The two matrices `A` and `B` should have the same number of columns. We also give the user an option to do row-wise Kronecker product, to avoid transpose. When doing row-wise Kronecker product, the number of rows of A and B should be the same.
`w`	the weights vector. The length of the vector should match with the dimension of the matrices. If performing column-wise Kronecker product, the length of w should be the same as the column number of A and B. If performing row-wise Kronecker prodoct, the length of w should be the same as the row number of A and B. The default is a vector of 1 if no value provided.
`byrow`	a logical variable controlling whether to perform row/column-wise Kronecker product. The default is `byrow`=TRUE.

Details

The column/row-wise Kronecker product.

Value

A matrix of the Khatri-Rao product.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
X <- matrix(rnorm(30), 10, 3)
Y <- matrix(runif(50), 10, 5)
C <- kr(X, Y)
```

logLik *Extract log-likelihood*

Description

`logLik` extracts log-likelihood for classes `"MGLMfit"`, `"MGLMreg"`, `"MGLMsparsereg"`.

Usage

```
## S4 method for signature 'MGLMfit'
logLik(object)

## S4 method for signature 'MGLMreg'
logLik(object)

## S4 method for signature 'MGLMsparsereg'
logLik(object)
```

Arguments

`object`	an object from which a log-likelihood value can be extracted.

Value

Returns a log-likelihood value of `object`.

Examples

```
library("MGLM")
data("rnaseq")
data <- rnaseq[, 1:6]
dmFit <- MGLMfit(data, dist = "DM")
logLik(dmFit)
```

maxlambda *Extract maximum lambda*

Description

`maxlambda` extracts the maximum tuning parameter that ensures the estimated regression coefficients are not all zero for the object of class `MGLMsparsereg`.

Usage

```
## S4 method for signature 'MGLMsparsereg'
maxlambda(object)
```

Arguments

`object`	an object of class `MGLMsparsereg` from which maximum lambda value can be extracted.

Value

Returns a maximum lambda value of `object`.

Examples

```
library("MGLM")
dist <- "DM"
n <- 100
p <- 10
d <- 5
set.seed(118)
m <- rbinom(n, 200, 0.8)
X <- matrix(rnorm(n * p), n, p)
alpha <- matrix(0, p, d)
alpha[c(1, 3, 5), ] <- 1
Alpha <- exp(X %*% alpha)
Y <- rdirmn(size = m, alpha = Alpha)
pen <- "group"
ngridpt <- 30
spmodelfit <- MGLMsparsereg(formula = Y ~ 0 + X, dist = dist,
                            lambda = Inf, penalty = pen)
maxlambda <- maxlambda(spmodelfit)
```

`MGLM-deprecated` *Deprecated function(s) in the MGLM package*

Description

These functions are provided for compatibility with older version of the yourPackageName package. They may eventually be completely removed.

Usage

```
ddirm(...)

rdirm(...)

dgdirm(...)

rgdirm(...)

dneg(Y, alpha, beta)
```

Arguments

`...`	parameters to be passed to the modern version of the function
`Y, alpha, beta`	for functions dnegmn, note the change in argument order. See Details.

Details

`ddirm`	now a synonym for `ddirmn`
`dgdirm`	now a synonym for `dgdirmn`
`dneg`	now a synonym for `dnegmn`
`rdirm`	now a synonym for `rdirmn`
`rgdirm`	now a synonym for `rgdirmn`

The function dneg has been deprecated. Use dnegmn instead.

Note the change in argument order: `dneg(Y, prob, beta)` and `dnegmn(Y, alpha, beta)` from MGLM_0.0.8 have been deprecated; use `dnegmn(Y, beta, prob = alpha/(rowSums(alpha)+1), alpha=NULL)` instead.

`MGLMfit-class` *Class* `"MGLMfit"`

Description

A class containing the model fitting results from the `MGLMfit`.

Slots

`estimate` object of class `"vector"`, containing the parameter estimates.

`SE` object of class `"vector"`, containing the standard errors of the estimates.

`vcov` object of class `"matrix"`, the variance covariance matrix of the parameter estimates.

`logL` object of class `"numeric"`, the fitted log likelihood.

`BIC` object of class `"numeric"`, Bayesian information criterion.

`AIC` object of class `"numeric"`, Akaike information criterion.

`LRTpvalue` object of class `"numeric"`, likelihood ratio test p value.

`gradient` object of class `"numeric"` or `"matrix"`, containing the gradient.

`iter` object of class `"numeric"`, number of iteration used.

`distribution` object of class `"character"`, the distribution fitted.

`fitted` object of class `"vector"`, the fitted mean of each category.

`LRT` object of class `"numeric"`, the likelihood ratio test statistic.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
showClass("MGLMfit")
```

`MGLMreg` *Fit multivariate response GLM regression*

Description

`MGLMreg` fits multivariate response generalized linear models, specified by a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
MGLMreg(
  formula,
  data,
  dist,
  init = NULL,
  weight = NULL,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE,
  LRT = FALSE,
  parallel = FALSE,
  cores = NULL,
  cl = NULL,
  sys = NULL,
  regBeta = FALSE
)

MGLMreg.fit(
  Y,
  init = NULL,
  X,
  dist,
  weight = NULL,
  epsilon = 1e-08,
  maxiters = 150,
  display = FALSE,
  LRT = FALSE,
  parallel = FALSE,
  cores = NULL,
  cl = NULL,
  sys = NULL,
  regBeta = FALSE
)
```

Arguments

`formula`	an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The response has to be on the left hand side of ~.
`data`	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in `data` when using function `MGLMreg`, the variables are taken from `environment(formula)`, typically the environment from which `MGLMreg` is called.
`dist`	a description of the error distribution to fit. See `dist` for details.
`init`	an optional matrix of initial value of the parameter estimates. Should have the compatible dimension with `data`. See `dist` for details of the dimensions in each distribution.
`weight`	an optional vector of weights assigned to each row of the data. Should be `NULL` or a numeric vector. Could be a variable from `data`, or a variable from `environment(formula)` with the length equal to the number of rows of the data. If `weight=NULL`, equal weights of ones will be assigned. Default is `NULL`.
`epsilon`	an optional numeric controlling the stopping criterion. The algorithm terminates when the relative change in the loglikelihoods of two successive iterates is less than `epsilon`. The default value is `epsilon=1e-8`.
`maxiters`	an optional numeric controlling the maximum number of iterations. The default value is `maxiters=150`.
`display`	an optional logical variable controlling the display of iterations. The default value is `display=FALSE`.
`LRT`	an optional logical variable controlling whether to perform likelihood ratio test on each predictor. The default value is `LRT=FALSE`, in which case only the Wald test is performed.
`parallel`	an optional logical variable controlling whether to perform parallel computing. On a multi-core Windows machine, a cluster is created based on socket; on a multi-core Linux/Mac machine, a cluster is created based on forking. The default value is `parallel=FALSE`.
`cores`	an optional value specifying the number of cores to use. Default value is half of the logical cores.
`cl`	a cluster object, created by the package **parallel** or by package **snow**. If `parallel=TRUE`, use the registered default cluster; if `parallel=FALSE`, any given value to `cl` will be ignored.
`sys`	the operating system. Will be used when choosing parallel type.
`regBeta`	an optional logical variable. When `dist="NegMN"`, the user can decide whether to run regression on the overdispersion parameter β. The default is `regBeta=FALSE`.
`Y, X`	for `MGLMreg.fit`, `X` is a design matrix of dimension `n*(p+1)` and `Y` is the response matrix of dimension `n*d`.

Details

The formula should be in the form responses ~ covariates where the responses are the multivariate count matrix or a few columns from a data frame which is specified by `data`. The covariates are either matrices or from the data frame. The covariates can be numeric or character or factor. See `dist` for details about distributions.

Instead of using the formula, the user can directly input the design matrix and the response vector using `MGLMreg.fit` function.

Value

Returns an object of class `"MGLMreg"`. An object of class `"MGLMreg"` is a list containing the following components:

- `coefficients` the estimated regression coefficients.
- `SE` the standard errors of the estimates.
- `Hessian` the Hessian at the estimated parameter values.
- `gradient` the gradient at the estimated parameter values.
- `wald.value` the Wald statistics.
- `wald.p` the p values of Wald test.
- `test` test statistic and the corresponding p-value. If `LRT=FALSE`, only returns test resultsfrom Wald test; if `LRT=TRUE`, returns the test results from both Wald test and likelihood ratio test.
- `logL` the final loglikelihood.
- `BIC` Bayesian information criterion.
- `AIC` Akaike information criterion.
- `fitted` the fitted values from the regression model
- `iter` the number of iterations used.
- `call` the matched call.
- `distribution` the distribution fitted.
- `data` the data used to fit the model.
- `Dof` degrees of freedom.

Author(s)

Yiwen Zhang and Hua Zhou

See Also

See also `MGLMfit` for distribution fitting.

Examples

```
##----------------------------------------##
## Generate data
n <- 2000
p <- 5
d <- 4
m <- rep(20, n)
set.seed(1234)
X <- 0.1* matrix(rnorm(n*p),n, p)
alpha <- matrix(1, p, d-1)
beta <- matrix(1, p, d-1)
Alpha <- exp(X %*% alpha)
Beta <- exp(X %*% beta)
gdm.Y <- rgdirmn(n, m, Alpha, Beta)

##----------------------------------------##
## Regression
gdm.reg <- MGLMreg(gdm.Y~X, dist="GDM", LRT=FALSE)
```

`MGLMreg-class` *Class* `"MGLMreg"`

Description

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

Slots

`call` object of class `"call"`.

`data` object of class `"list"` , consists of both the predictor matrix and the response matrix.

`coefficients` object of class `"list"` or `"matrix"`, the estimated parameters.

`SE` object of class `"list"` or `"matrix"`, the standard errors of the parameters.

`test` object of class `"matrix"`, the test statistics and p-values.

`Hessian` object of class `"matrix"`, the Hessian matrix.

`logL` object of class `"numeric"`, the loglikelihood.

`BIC` object of class `"numeric"`,

`AIC` object of class `"numeric"`, Akaike information criterion.

`iter` object of class `"numeric"`, the number of iteration used.

`distribution` object of class `"character"`, the distribution fitted.

`fitted` object of class `"vector"`, the fitted value.

`gradient` object of class `"numeric"` or `"matrix"`, the gradient at the estimated parameter values.

`wald.value` object of class `"numeric"` or `"logical"`, the Wald statistics.

`wald.p` object of class `"numeric"` or `"logical"`, the p values of Wald test.

`Dof` object of class `"numeric"`, the degrees of freedom.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
showClass("MGLMreg")
```

`MGLMsparsereg` *Fit multivariate GLM sparse regression*

Description

Fit sparse regression in multivariate generalized linear models.

Usage

```
MGLMsparsereg(
  formula,
  data,
  dist,
  lambda,
  penalty,
  weight,
  init,
  penidx,
  maxiters = 150,
  ridgedelta,
  epsilon = 1e-05,
  regBeta = FALSE,
  overdisp
)

MGLMsparsereg.fit(
  Y,
  X,
  dist,
  lambda,
  penalty,
  weight,
  init,
  penidx,
  maxiters = 150,
  ridgedelta,
  epsilon = 1e-05,
  regBeta = FALSE,
  overdisp
)
```

Arguments

`formula`	an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The response has to be on the left hand side of `~`.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data` when using function `MGLMsparsereg`, the variables are taken from `environment(formula)`, typically the environment from which `MGLMsparsereg` is called.
`dist`	a description of the error distribution to fit. See `dist` for details.
`lambda`	penalty parameter.
`penalty`	penalty type for the regularization term. Can be chosen from `"sweep"`, `"group"`, or `"nuclear"`. See Details for the description of each penalty type.
`weight`	an optional vector of weights assigned to each row of the data. Should be `NULL` or a numeric vector. Could be a variable from `data`, or a variable from `environment(formula)` with the length equal to the number of rows of the data. If `weight=NULL`, equal weights of ones will be assigned.
`init`	an optional matrix of initial value of the parameter estimates. Should have the compatible dimension with the data. See `dist` for details of the dimensions in each distribution.
`penidx`	a logical vector indicating the variables to be penalized. The default value is `rep(TRUE, p)`, which means all predictors are subject to regularization. If `X` contains intercept, use `penidx=c(FALSE,rep(TRUE,p-1))`.
`maxiters`	an optional numeric controlling the maximum number of iterations. The default value is maxiters=150.
`ridgedelta`	an optional numeric controlling the behavior of the Nesterov's accelerated proximal gradient method. The default value is $\frac{1}{pd}$.
`epsilon`	an optional numeric controlling the stopping criterion. The algorithm terminates when the relative change in the objective values of two successive iterates is less then `epsilon`. The default value is `epsilon=1e-5`.
`regBeta`	an optional logical variable used when running negative multinomial regression (`dist="NegMN"`). `regBeta` controls whether to run regression on the over-dispersion parameter. The default is `regBeta=FALSE`.
`overdisp`	an optional numerical variable used only when fitting sparse negative multinomial model `dist="NegMN"` and `regBeta=FALSE`. `overdisp` gives the over dispersion value for all the observations. The default value is estimated using negative-multinomial regression. When `dist="MN", "DM", "GDM"` or `regBeta=TRUE`, the value of `overdisp` is ignored.
`Y`	a matrix containing the multivariate categorical response data. Rows of the matrix represent observations, while columns are the different categories. Rows and columns of all zeros are automatically removed when `dist="MN"`, `"DM"`, or `"GDM"`.
`X`	design matrix (including intercept). Number of rows of the matrix should match that of `Y`.

Details

In general, we consider regularization problem

$$\min_B h(B) = -l(B) + J(B),$$

where $l(B)$ is the loglikelihood function and $J(B)$ is the regularization function.

Sparsity in the individual elements of the parameter matrix B is achieved by the lasso penalty (`dist="sweep"`)

$$J(B) = \lambda \sum_{k \in penidx} \sum_{j=1}^{d} \|B_{kj}\|$$

Sparsity in the rows of the regression parameter matrix B is achieved by the group penalty (`dist="group"`)

$$J(B) = \lambda \sum_{k \in penidx} \|B_{k\cdot}\|_2,$$

where $\|v\|_2$ is the l_2 norm of a vector v. In other words, $\|B_{k\cdot}\|_2$ is the l_2 norm of the k-th row of the parameter matrix B.

Sparsity in the rank of the parameter matrix B is achieved by the nuclear norm penalty (`dist="nuclear"`)

$$J(B) = \lambda\|B\|_* = \lambda \sum_{i=1}^{min(p,d)} \sigma_i(B),$$

where $\sigma_i(B)$ are the singular values of the parameter matrix B. The nuclear norm $\|B\|_*$ is a convex relaxation of $rank(B) = \|\sigma(B)\|_0$.

See `dist` for details about distributions.

Value

Returns an object of class `"MGLMsparsereg"`. An object of class `"MGLMsparsereg"` is a list containing at least the following components:

- `coefficients` the estimated matrix of regression coefficients.
- `logL` the final loglikelihood value.
- `AIC` Akaike information criterion.
- `BIC` Bayesian information criterion.
- `Dof` degrees of freedom of the estimated parameter.
- `iter` number of iterations used.
- `maxlambda` the maxmum tuning parameter such that the estimated coefficients are not all zero. This value is returned only when the tuning parameter `lambda` given to the function is large enough such that all the parameter estimates are zero; otherwise, `maxlambda` is not computed.
- `call` a matched call.
- `data` the data used to fit the model: a list of the predictor matrix and the response matrix.
- `penalty` the penalty chosen when running the penalized regression.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
## Generate Dirichlet Multinomial data
dist <- "DM"
n <- 100
p <- 15
d <- 5
m <- runif(n, min=0, max=25) + 25
set.seed(134)
X <- matrix(rnorm(n*p),n, p)
alpha <- matrix(0, p, d)
alpha[c(1,3, 5), ] <- 1
Alpha <- exp(X%*%alpha)
Y <- rdirmn(size=m, alpha=Alpha)

## Tuning
ngridpt <- 10
p <- ncol(X)
d <- ncol(Y)
pen <- 'nuclear'
spfit <- MGLMsparsereg(formula=Y~0+X, dist=dist, lambda=Inf, penalty=pen)
```

`MGLMsparsereg-class` *Class* `"MGLMsparsereg"`

Description

A class containing the results from the `MGLMsparsereg`.

Slots

`call` object of class `"call"`.

`data` object of class `"list"` , consists of both the predictor matrix and the response matrix.

`coefficients` object of class `"matrix"`, the estimated parameters.

`logL` object of class `"numeric"`, the loglikelihood.

`BIC` object of class `"numeric"`,

`AIC` object of class `"numeric"`, Akaike information criterion.

`Dof` object of class `"numeric"`, the degrees of freedom.

`iter` object of class `"numeric"`, the number of iteration used.

`maxlambda` object of class `"numeric"`, the maximum tuning parameter that ensures the estimated regression coefficients are not all zero.

`lambda` object of class `"numeric"`, the tuning parameter used.

`distribution` object of class `"character"`, the distribution fitted.

`penalty` Object of class `"character"`, the chosen penalty when running penalized regression.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
showClass("MGLMsparsereg")
```

MGLMtune *Choose the tuning parameter value in sparse regression*

Description

Finds the tuning parameter value that yields the smallest BIC.

Usage

```
MGLMtune(
  formula,
  data,
  dist,
  penalty,
  lambdas,
  ngridpt,
  warm.start = TRUE,
  keep.path = FALSE,
  display = FALSE,
  init,
  weight,
  penidx,
  ridgedelta,
  maxiters = 150,
  epsilon = 1e-05,
  regBeta = FALSE,
  overdisp
)
```

Arguments

`formula`	an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The response has to be on the left hand side of ~.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data` when using function `MGLMtune`, the variables are taken from `environment(formula)`, typically the environment from which `MGLMtune` is called.
`dist`	a description of the distribution to fit. See `dist` for the details.

`penalty`	penalty type for the regularization term. Can be chosen from `"sweep"`, `"group"`, or `"nuclear"`. See MGLMsparsereg for the description of each penalty type.
`lambdas`	an optional vector of the penalty values to tune. If missing, the vector of penalty values will be set inside the function. `ngridpt` must be provided if `lambdas` is missing.
`ngridpt`	an optional numeric variable specifying the number of grid points to tune. If `lambdas` is given, `ngridpt` will be ignored. Otherwise, the maximum λ is determined from the data. The smallest $\lambda$is set to $1/n$, where n is the sample size.
`warm.start`	an optional logical variable to specify whether to give warm start at each tuning grid point. If `warm.start=TRUE`, the fitted sparse regression coefficients will be used as the initial value when fitting the sparseregression with the next tuning grid.
`keep.path`	an optional logical variable controling whether to output the whole solution path. The default is `keep.path=FALSE`. If `keep.path=TRUE`, the sparse regression result at each grid point will be kept, and saved in the output object `select.list`.
`display`	an optional logical variable to specify whether to show each tuning step.
`init`	an optional matrix of initial value of the parameter estimates. Should have the compatible dimension with the data. See `dist` for details of dimensions in each distribution.
`weight`	an optional vector of weights assigned to each row of the data. Should be `NULL` or a numeric vector. Could be a variable from the `data`, or a variable from `environment(formula)` with the length equal to the number of rows of the data. If `weight=NULL`, equal weights of ones will be assigned.
`penidx`	a logical vector indicating the variables to be penalized. The default value is `rep(TRUE, p)`, which means all predictors are subject to regularization. If `X` contains intercept, use `penidx=c(FALSE,rep(TRUE,p-1))`.
`ridgedelta`	an optional numeric controlling the behavior of the Nesterov's accelerated proximal gradient method. The default value is $\frac{1}{pd}$.
`maxiters`	an optional numeric controlling the maximum number of iterations. The default value is `maxiters=150`.
`epsilon`	an optional numeric controlling the stopping criterion. The algorithm terminates when the relative change in the objective values of two successive iterates is less then `epsilon`. The default value is `epsilon=1e-5`.
`regBeta`	an optional logical variable used when running negative multinomial regression (`dist="NegMN"`). `regBeta` controls whether to run regression on the over-dispersion parameter. The default is `regBeta=FALSE`.
`overdisp`	an optional numerical variable used only when fitting sparse negative multinomial model and `regBeta=FALSE`. `overdisp` gives the over-dispersion value for all the observations. The default value is estimated using negative-multinomial regression. When `dist="MN", "DM", "GDM"` or `regBeta=TRUE`, the value of `overdisp` is ignored.

Value

- `select` the final sparse regression result, using the optimal tuning parameter.
- `path` a data frame with degrees of freedom and BICs at each lambda.

Author(s)

Yiwen Zhang and Hua Zhou

See Also

`MGLMsparsereg`

Examples

```
set.seed(118)
n <- 50
p <- 10
d <- 5
m <- rbinom(n, 100, 0.8)
X <- matrix(rnorm(n * p), n, p)
alpha <- matrix(0, p, d)
alpha[c(1, 3, 5), ] <- 1
Alpha <- exp(X %*% alpha)
Y <- rdirmn(size=m, alpha=Alpha)
sweep <- MGLMtune(Y ~ 0 + X, dist="DM", penalty="sweep", ngridpt=10)
show(sweep)
```

`MGLMtune-class` *Class* `"MGLMtune"`

Description

A class containing the results from the `MGLMtune`.

Slots

`call` object of class `"call"`.

`select` object of class `"MGLMsparsereg"`, regularized regression results given by the optimal tuning parameter.

`path` object of class `"data.frame"`, the BIC, AIC, log-likelihood and degrees of freedom given each tuning parameter.

`select.list` object of class `"list"`, the regularized regression results at each tuning grid point.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
showClass("MGLMtune")
```

`path` *Extract path*

Description

`path` extracts from object of the class `MGLMtune` the path of BIC, AIC, log-likelihood and degrees of freedom given each tuning parameter.

Usage

```
## S4 method for signature 'MGLMtune'
path(object)
```

Arguments

`object`	an object of class `MGLMtune` from which path can be extracted.

Value

Returns a path of `object`.

Examples

```
library("MGLM")
dist <- "DM"
n <- 100
p <- 10
d <- 5
set.seed(118)
m <- rbinom(n, 200, 0.8)
X <- matrix(rnorm(n * p), n, p)
alpha <- matrix(0, p, d)
alpha[c(1, 3, 5), ] <- 1
Alpha <- exp(X %*% alpha)
Y <- rdirmn(size = m, alpha = Alpha)
select <- MGLMtune(Y ~ 0 + X, dist = "DM", penalty = "nuclear",
ngridpt = 10, display = FALSE)
select_path <- path(select)
```

predict *Predict method for MGLM Fits*

Description

Predict using the fitted model from `MGLMreg` when given a new set of covariates.

Usage

```
## S4 method for signature 'MGLMreg'
predict(object, newdata)
```

Arguments

`object`	model object.
`newdata`	new covariates data matrix.

Value

Outputs the probabilities of each category.

This helps answer questions such as whether certain features increase the probability of observing category j.

Examples

```
n <- 200
p <- 5
d <- 4
X <- matrix(runif(p * n), n, p)
alpha <- matrix(c(0.6, 0.8, 1), p, d - 1, byrow=TRUE)
alpha[c(1, 2),] <- 0
Alpha <- exp(X %*% alpha)
beta <- matrix(c(1.2, 1, 0.6), p, d - 1, byrow=TRUE)
beta[c(1, 2),] <- 0
Beta <- exp(X %*% beta)
m <- runif(n, min=0, max=25) + 25
Y <- rgdirmn(n, m, Alpha, Beta)
gdmReg <- MGLMreg(Y~0+X, dist="GDM")
newX <- matrix(runif(1*p), 1, p)
pred <- predict(gdmReg, newX)
```

`rdirmn` *The Dirichlet Multinomial Distribution*

Description

`ddirmn` computes the log of the Dirichlet multinomial probability mass function. `rdirmn` generates Dirichlet multinomially distributed random number vectors.

Usage

```
rdirmn(n, size, alpha)

ddirmn(Y, alpha)
```

Arguments

`n`	number of random vectors to generate. When `size` is a scalar and `alpha` is a vector, must specify `n`. When `size` is a vector and `alpha` is a matrix, `n` is optional. The default value of `n` is the length of `size`. If given, `n` should be equal to the length of `size`.
`size`	a number or vector specifying the total number of objects that are put into d categories in the Dirichlet multinomial distribution.
`alpha`	the parameter of the Dirichlet multinomial distribution. Can be a numerical positive vector or matrix. For `ddirmn`, `alpha` has to match the size of `Y`. If `alpha` is a vector, it will be replicated n times to match the dimension of `Y`. For `rdirmn`, if `alpha` is a vector, `size` must be a scalar, and all the random vectors will be drawn from the same `alpha` and `size`. If `alpha` is a matrix, the number of rows should match the length of `size`, and each random vector will be drawn from the corresponding row of `alpha` and the corresponding element in the `size` vector. See Details below.
`Y`	The multivariate count matrix with dimensions $n \times d$, where $n = 1, 2, \ldots$ is the number of observations and $d = 2, 3, \ldots$ is the number of categories.

Details

When the multivariate count data exhibits over-dispersion, the traditional multinomial model is insufficient. Dirichlet multinomial distribution models the probabilities of the categories by a Dirichlet distribution. Given the parameter vector $\alpha = (\alpha_1, \ldots, \alpha_d), \alpha_j > 0$, the probability mass of d-category count vector $Y = (y_1, \ldots, y_d)$, $d \geq 2$ under Dirichlet multinomial distribution is

$$P(y|\alpha) = C^m_{y_1,\ldots,y_d} \prod_{j=1}^{d} \frac{\Gamma(\alpha_j + y_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\sum_{j'=1}^{d} \alpha_{j'})}{\Gamma(\sum_{j'=1}^{d} \alpha_{j'} + \sum_{j'=1}^{d} y_{j'})},$$

where $m = \sum_{j=1}^{d} y_j$. Here, C^n_k, often read as "n choose k", refers the number of k combinations from a set of n elements.

The parameter α can be a vector of length d, such as the results from the distribution fitting. α can also be a matrix with n rows, such as the inverse link calculated from the regression parameter estimate $exp(X\beta)$.

Value

For each count vector and each corresponding parameter vector α, the function `ddirmn` returns the value $\log(P(y|\alpha))$. When `Y` is a matrix of n rows, `ddirmn` returns a vector of length n.

`rdirmn` returns a $n \times d$ matrix of the generated random observations.

Examples

```
m <- 20
alpha <- c(0.1, 0.2)
dm.Y <- rdirmn(n=10, m, alpha)
pdfln <- ddirmn(dm.Y, alpha)
```

`rgdirmn` *The Generalized Dirichlet Multinomial Distribution*

Description

`rgdirmn` generates random observations from the generalized Dirichlet multinomial distribution. `dgdirmn` computes the log of the generalized Dirichlet multinomial probability mass function.

Usage

```
rgdirmn(n, size, alpha, beta)

dgdirmn(Y, alpha, beta)
```

Arguments

`n`	the number of random vectors to generate. When `size` is a scalar and `alpha` is a vector, must specify `n`. When `size` is a vector and `alpha` is a matrix, `n` is optional. The default value of `n` is the length of `size`. If given, `n` should be equal to the length of `size`.
`size`	a number or vector specifying the total number of objects that are put into d categories in the generalized Dirichlet multinomial distribution.
`alpha`	the parameter of the generalized Dirichlet multinomial distribution. `alpha` is a numerical positive vector or matrix. For `gdirmn`, `alpha` should match the size of `Y`. If `alpha` is a vector, it will be replicated n times to match the dimension of `Y`. For `rdirmn`, if `alpha` is a vector, `size` must be a scalar. All the random vectors will be drawn from the same `alpha` and `size`. If `alpha` is a matrix, the number of rows should match the length of `size`. Each random vector will be drawn from the corresponding row of `alpha` and the corresponding element of `size`.

`beta`	the parameter of the generalized Dirichlet multinomial distribution. `beta` should have the same dimension as `alpha`. For `rdirm`, if `beta` is a vector, `size` must be a scalar. All the random samples will be drawn from the same `beta` and `size`. If `beta` is a matrix, the number of rows should match the length of `size`. Each random vector will be drawn from the corresponding row of `beta` and the corresponding element of `size`.
`Y`	the multivariate count matrix with dimensions $n \times d$, where $n = 1, 2, \ldots$ is the number of observations and $d = 3, 4, \ldots$ is the number of categories.

Details

$Y = (y_1, \ldots, y_d)$ are the d category count vectors. Given the parameter vector $\alpha = (\alpha_1, \ldots, \alpha_{d-1}), \alpha_j > 0$, and $\beta = (\beta_1, \ldots, \beta_{d-1}), \beta_j > 0$, the generalized Dirichlet multinomial probability mass function is

$$P(y|\alpha, \beta) = C^m_{y_1,\ldots,y_d} \prod_{j=1}^{d-1} \frac{\Gamma(\alpha_j + y_j)}{\Gamma(\alpha_j)} \frac{\Gamma(\beta_j + z_{j+1})}{\Gamma(\beta_j)} \frac{\Gamma(\alpha_j + \beta_j)}{\Gamma(\alpha_j + \beta_j + z_j)},$$

where $z_j = \sum_{k=j}^d y_k$ and $m = \sum_{j=1}^d y_j$. Here, C^n_k, often read as "n choose k", refers the number of k combinations from a set of n elements.

The α and β parameters can be vectors, like the results from the distribution fitting function, or they can be matrices with n rows, like the estimate from the regression function multiplied by the covariate matrix $exp(X\alpha)$ and $exp(X\beta)$

Value

`dgdirmn` returns the value of $\log(P(y|\alpha, \beta))$. When `Y` is a matrix of n rows, the function `dgdirmn` returns a vector of length n.

`rgdirmn` returns a $n \times d$ matrix of the generated random observations.

Examples

```
# example 1
m <- 20
alpha <- c(0.2, 0.5)
beta <- c(0.7, 0.4)
Y <- rgdirmn(10, m, alpha, beta)
dgdirmn(Y, alpha, beta)

# example 2
set.seed(100)
alpha <- matrix(abs(rnorm(40)), 10, 4)
beta <- matrix(abs(rnorm(40)), 10, 4)
size <- rbinom(10, 10, 0.5)
GDM.rdm <- rgdirmn(size=size, alpha=alpha, beta=beta)
GDM.rdm1 <- rgdirmn(n=20, size=10, alpha=abs(rnorm(4)), beta=abs(rnorm(4)))
```

rmn *The Multinomial Distribution*

Description

`rmn` generates random number vectors given `alpha`. The function `rmn(n, size, alpha)` calls `rmultinom(n, size, prob)` after converting `alpha` to probability. `dmn` computes the log of multinomial probability mass function.

Usage

```
rmn(n, size, alpha)

dmn(Y, prob)
```

Arguments

`n`	number of random vectors to generate.
`size`	a scalar or a vector.
`alpha`	a vector or a matrix.
`Y`	the multivariate count matrix with dimension $n \times d$, where $n = 1, 2, \dots$ is number of observations and $d = 2, \dots$ is number of categories.
`prob`	the probability parameter of the multinomial distribution. `prob` can be either a vector of length d or a matrix with matching size of `Y`. If `prob` is a vector, it will be replicated n times to match the dimension of `Y`. If the sum(s) of `prob` is not 1, it will be automatically scaled to have sum 1.

Details

A multinomial distribution models the counts of d possible outcomes. The counts of categories are negatively correlated. $y = (y_1, \dots, y_d)$ is a d category count vector. Given the parameter vector $p = (p_1, \dots, p_d)$, $0 < p_j < 1$, $\sum_{j=1}^d p_j = 1$, the function calculates the log of the multinomial pmf

$$P(y|p) = C_{y_1,\dots,y_d}^m \prod_{j=1}^d p_j^{y_j},$$

where $m = \sum_{j=1}^d y_j$. Here, C_k^n, often read as "n choose k", refers the number of k combinations from a set of n elements.

The parameter p can be one vector, like the result from the distribution fitting function; or, p can be a matrix with n rows, like the estimate from the regression function,

$$p_j = \frac{exp(X\beta_j)}{1 + sum_{j'=1}^{d-1} exp(X\beta_{j'})},$$

where $j = 1, \dots, d-1$ and $p_d = \frac{1}{1+\sum_{j'=1}^{d-1} exp(X\beta_{j'})}$. The d-th column of the coefficient matrix β is set to 0 to avoid the identifiability issue.

Value

The function dmn returns the value of $\log(P(y|p))$. When Y is a matrix of n rows, the function returns a vector of length n.

The function rmn returns multinomially distributed random number vectors

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
m <- 20
prob <- c(0.1, 0.2)
dm.Y <- rdirmn(n=10, m, prob)
pdfln <- dmn(dm.Y, prob)
```

---

rnaseq *RNA-seq count data*

Description

RNA-seq data simulated following the standard procedures (provided by Dr. Wei Sun, weisun@email.unc.edu).

Usage

```
rnaseq
```

Format

A data frame containing 10 columns and 100 rows. The first 6 columns are the expression counts of 6 exons of a gene; the last four columns are the covariates: age, gender, treatment, and total number of reads.

Source

Dr. Sun Wei, weisun@email.unc.edu

rnegmn	*The Negative Multinomial Distribution*

Description

`dnegmn` calculates the log of the negative multinomial probability mass function. `rnegmn` generates random observations from the negative multinomial distribution.

Usage

```
rnegmn(n, beta, prob)

dnegmn(Y, beta, prob = alpha/(rowSums(alpha) + 1), alpha = NULL)
```

Arguments

`n`	number of random vectors to generate. When `beta` is a scalar and `prob` is a vector, must specify `n`. When `beta` is a vector and `prob` is a matrix, `n` is optional. The default value of `n` is the length of `beta`. If given, `n` should be equal to the length of `beta`.
`beta`	the over dispersion parameter of the negative multinomial distribution. `beta` can be either a scalar or a vector of length n.
`prob`	the probability parameter of the negative multinomial distribution. Should be a numerical non-negative vector or matrix. For `dnegmn`, `prob` can be either a vector of length d ($d \geq 2$) or a matrix with matching size of `Y`. If `prob` is a vector, it will be replicated n times to match the dimension of `Y`. The sum of each row of `prob` should be smaller than 1. For `rnegmn`, If `prob` is a vector, `beta` must be a scalar. All the `n` random vectors will be drawn from the same `prob` and `beta`. If `prob` is a matrix, the number of rows should match the length of `beta`. Each random vector will be drawn from the corresponding row of `prob` and the corresponding element of `beta`. Each row of `prob` should have sum less than 1.
`Y`	the multivariate response matrix of dimension $n \times d$, where $n = 1, 2, \ldots$ is number of observations and $d = 2, 3, \ldots$ is number of categories.
`alpha`	an alternative way to specify the probability. Default value is `NULL`. See details.

Details

$y = (y_1, \ldots, y_d)$ is a d category vector. Given the parameter vector $p = (p_1, \ldots, p_d)$, $p_{d+1} = 1/(1 + \sum_{j'=1}^{d} p_{j'})$, and β, $\beta > 0$, the negative multinomial probability mass function is

$$P(y|p, \beta) = C_m^{\beta+m-1} C_{y_1,\ldots,y_d}^{m} \prod_{j=1}^{d} p_j^{y_j} p_{d+1}^{\beta} = \frac{\beta_m}{m!} \binom{m}{y_1, \ldots, y_d} \prod_{j=1}^{d} p_j^{y_j} p_{d+1}^{\beta},$$

where $m = \sum_{j=1}^{d} y_j$. Here, C_k^n, often read as "n choose k", refers the number of k combinations from a set of n elements.

alpha is an alternative way to specify the probability:

$$p_j = \frac{\alpha_j}{(1 + \sum_{k=1}^{d} \alpha_k)}$$

for $j = 1, \ldots, d$ and $p_{d+1} = \frac{1}{(1+\sum_{k=1}^{d} \alpha_k)}$.

The parameter prob can be a vector and beta is a scalar; prob can also be a matrix with n rows, and beta is a vector of length n like the estimate from the regression function multiplied by the covariate matrix.

Value

dnegmn returns the value of $\log(P(y|p, \beta))$. When Y is a matrix of n rows, the function returns a vector of length n.

rnegmn returns a $n \times d$ matrix of the generated random observations.

Author(s)

Yiwen Zhang and Hua Zhou

Examples

```
###-----------------------###
set.seed(128)
n <- 100
d <- 4
p <- 5
a <- -matrix(1,p,d)
X <- matrix(runif(n*p), n, p )
alpha <- exp(X%*%a)
prob <- alpha/(rowSums(alpha)+1)
beta <- exp(X%*%matrix(1,p))
Y <- rnegmn(n, beta, prob)

###-----------------------###
m <- 20
n <- 10
p <- 5
d <- 6
a <- -matrix(1,p,d)
X <- matrix(runif(n*p), n, p )
alpha <- exp(X%*%a)
prob <- alpha/(rowSums(alpha)+1)
b <- exp(X%*%rep(0.3,p))
Y <- rnegmn(prob=prob, beta=rep(10, n))
dnegmn(Y, b, prob)
```

show *Show an object*

Description

Display the object by printing its class.

Usage

```
## S4 method for signature 'MGLMfit'
show(object)

## S4 method for signature 'MGLMreg'
show(object)

## S4 method for signature 'MGLMsparsereg'
show(object)

## S4 method for signature 'MGLMtune'
show(object)
```

Arguments

`object`	an object to be printed. Should be of class `"MGLMfit"`, `"MGLMreg"`, `"MGLMsparsereg"` or `"MGLMtune"`.

Examples

```
library("MGLM")
data("rnaseq")
data <- rnaseq[, 1:6]
gdmFit <- MGLMfit(data, dist = "GDM")
show(gdmFit)
```

Index