Package ‘automultinomial’

October 5, 2016

Title Autologistic and Automultinomial Spatial Regression and Variable Selection

Version 1.0.0

Author Stephen Berg [aut, cre], Rao Fu [aut], Jun Zhu [ths]

Maintainer Stephen Berg `<saberg2@wisc.edu>`

Description Contains functions for autologistic variable selection and parameter estimation for spatially correlated categorical data (including k>2). The main function is MPLE. Capable of fitting the centered autologistic model described in Caragea and Kaiser (2009)<doi:10.1198/jabes.2009.07032>, as well as the traditional autologistic model of Besag (1974)<http://www.jstor.org/stable/2984812>.

Depends R (>= 3.2.0)

Imports graphics, Hmisc, igraph, Matrix, MASS, stats, utils

License GPL-2 | GPL-3

LazyData true

Suggests testthat, numDeriv, knitr, rmarkdown

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-10-05 09:29:49

R topics documented:

`adjMat` *Spatial adjacency matrix*

Description

Takes as input a matrix gridMat with the following format: 0 means a site is included in the study area, and nonzero values mean the site is not included. Returns an adjacency matrix for the study sites, where sites at a Manhattan distance of exactly kNN are considered neighbors.

Usage

```
adjMat(m, n, boundary = FALSE)
```

Arguments

`m`	height of the grid
`n`	width of the grid
`boundary`	surround grid by an external boundary, defaults to FALSE

Value

a list containing a sparse adjacency matrix for a square lattice, and the vector of the points not on the edge

Examples

```
mat=adjMat(10,10)
```

centeredLogLikelihood *Centered model log pseudolikelihood*

Description

Computes a log pseudolikelihood for the centered model, taking coefficients, a design matrix, a response matrix, and a list of adjacency matrices

Usage

```
centeredLogLikelihood(theta, X, z, A, constraint, thetaInds, innerIndices)
```

Arguments

`theta`	a matrix of coefficients with dimension (p+(k-1)*length(A)) by k; frequently A will have length 1
`X`	nxp design matrix
`z`	nxk response matrix
`A`	a list of adjacency matrices
`constraint`	constraint on eta
`thetaInds`	relevant indices
`innerIndices`	internal grid indices

Value

a log (pseudo-)likelihood

Examples

```
#g=centeredLogLikelihood(theta,X,z,A)
```

centeredLogLikGrad *Centered model gradient*

Description

Computes a gradient for the centered model, taking coefficients, a design matrix, a response matrix, and a list of adjacency matrices

Usage

```
centeredLogLikGrad(theta, X, z, A, constraint, thetaInds, innerIndices = NULL)
```

Arguments

`theta`	a matrix of coefficients with dimension (p+(k-1)*length(A)) by k; frequently A will have length 1
`X`	nxp design matrix
`z`	nxk response matrix
`A`	a list of adjacency matrices
`constraint`	constraint on eta
`thetaInds`	relevant indices
`innerIndices`	only here to match argument list with uncentered case

Value

a gradient vector

Examples

```
#g=centeredLogLikGrad(theta,X,z,A)
```

`centeredLogLikHess` *Centered model hessian*

Description

Computes a hessian for the centered model, taking coefficients, a design matrix, a response matrix, and a list of adjacency matrices

Usage

```
centeredLogLikHess(theta, X, z, A, constraint, thetaInds)
```

Arguments

`theta`	a matrix of coefficients with dimension (p+(k-1)*length(A)) by k; frequently A will have length 1
`X`	nxp design matrix
`z`	nxk response matrix
`A`	a list of adjacency matrices
`constraint`	constraint on eta
`thetaInds`	relevant indices

Value

a gradient vector

Examples

```
#g=centeredLogLikHess(theta,X,z,A)
```

chol_psd *Cholesky decomposition for positive semidefinite matrix*

Description

Takes a square symmetric positive definite or positive semidefinite matrix and returns a Cholesky factor.

Usage

```
chol_psd(a)
```

Arguments

a	A matrix

Value

The lower triangular Cholesky factor

Examples

```
X=replicate(100,rnorm(99))

#a is positive semidefinite but not positive definite
a_0=t(X)%*%X
b=chol_psd(a_0)

#recover a
a_1=b%*%t(b)
diff=max(abs(a_1-a_0))
```

criterionFit *Minimize criterion (AIC or BIC)*

Description

Takes a list of coefficient values and finds the values which minimize AIC or BIC

Usage

```
criterionFit(thetaList, X, z, A, BIC, logLike, constraint, thetaInds,
  innerIndices)
```

Arguments

`thetaList`	a list of coefficient values (probably from a LASSO path)
`X`	a design matrix
`z`	response matrix (for multinomial or logistic data)
`A`	a list of adjacency matrices
`BIC`	logical, use BIC if true or AIC if false
`logLike`	function to compute the loglikelihood: either centered or uncentered
`constraint`	constraint on eta
`thetaInds`	relevant indices
`innerIndices`	internal indices

Value

List containing a minimum criterion coefficient vector and the criterion for that vector

Examples

```
#criterionFit(thetaList,X,z,A,BIC,logLike,constraint,thetaInds,innerIndices)
```

`indsFunction` *Get indices of relevant component of theta matrix*

Description

Gets indices of beta, eta, diagonal of eta, and off diagonal elements of eta

Usage

```
indsFunction(theta, X, z, A)
```

Arguments

`theta`	matrix of coefficients
`X`	design matrix
`z`	matrix of multinomial responses
`A`	list of adjacency matrices

Value

list of index vectors

Examples

```
#
```

JMatrixCentered *J matrix for centered variance estimate*

Description

Computes J matrix for the centered variance estimate

Usage

```
JMatrixCentered(theta, X, z, A, constraint, thetaInds)
```

Arguments

theta	coefficients
X	design matrix
z	response matrix
A	list of adjacency matrices
constraint	"symmetric" or "diagonal"
thetaInds	list of relevant indices in theta: betaInds, etaDiag, ...

Value

A sparse adjacency matrix

Examples

```
#JMatrix=JMatrixCentered(theta,X,z,A)
```

JMatrixUncentered *Uncentered J matrix*

Description

Used for a variance estimate accounting for correlation between sites or for sandwich estimate for independent sites.

Usage

```
JMatrixUncentered(theta, X, z, A, constraint, thetaInds, innerIndices)
```

Arguments

`theta`	a matrix of coefficients
`X`	a design matrix
`z`	a matrix of responses
`A`	a list of adjacency matrices
`constraint`	"symmetric" or "diagonal"
`thetaInds`	list of useful indices
`innerIndices`	the 1-dimensional indices of points within the grid

Value

The J matrix for the sandwich variance estimator.

Examples

```
#JMatrix=JMatrixUnCentered(theta,X,z,A,innerIndices)
```

`logLikelihood` *Compute log likelihood*

Description

Compute log likelihood for uncentered or independent case models

Usage

```
logLikelihood(theta, X, z, A, constraint, thetaInds, innerIndices)
```

Arguments

`theta`	a matrix of coefficinets
`X`	design matrix
`z`	response matrix
`A`	list of adjacency matrices/list containing adjacency matrix
`constraint`	constraint on eta
`thetaInds`	relevant indices
`innerIndices`	internal grid indices

Value

A sparse adjacency matrix

Examples

```
#logLikelihood(theta,X,z,A,constraint,thetaInds)
```

MPLE *Logistic autoregression*

Description

Fits an autologistic or automultinomial logit model by pseudolikelihood. Fits the traditional uncentered autologistic model by default, or its multicategory analog. Also capable of fitting the centered autologistic/automultinomial models. Performs an optional group LASSO variable selection step, with an option to group related variables. For the automultinomial model, the variable selection step treats coefficients relating to a single predictor as grouped, as in glmnet with the option type.multinomial="grouped". This means that entire rows of the multinomial coefficient matrix are selected or removed.

Usage

```
MPLE(X, z, A = NULL, innerIndices = NULL, groups = 2:dim(X)[2],
  nLambda = 101, centered = FALSE, BIC = TRUE, select = TRUE,
  standardize = TRUE, constraint = "diagonal", NR = FALSE)
```

Arguments

`X`	nxp design matrix, where the first column contains an intercept
`z`	nxk response matrix or a length n vector containing the response type of each observation
`A`	an adjacency matrix specifying the neighborhood structure
`innerIndices`	vector containing points to be treated as internal to the grid (defaults to all points if unspecified)
`groups`	the grouping of the (non-intercept) coefficients. A vector of length (p-1)
`nLambda`	the length of the lasso path, defaults to 101
`centered`	logical, use centered model (TRUE) or uncentered model (FALSE)? Defaults to FALSE
`BIC`	logical, default TRUE uses BIC and FALSE uses AIC for variables selection
`select`	logical, do variable selection? defaults to TRUE to include a variable selection step
`standardize`	logical, standardize non-intercept columns? Default is TRUE and usually makes more sense
`constraint`	the constraint on the eta matrix to ensure a valid joint distribution: "diagonal" (default) or "symmetric"
`NR`	use Newton-Raphson optimizer? Defaults to false, but NR may give faster and sharper convergence for uncentered model

Value

a list containing model fits and variance estimates

Examples

```
## Not run:
#Simulate data and estimate coefficients for binary data (example 1)
#and multicategory data (example 2)


###### Example 1: binary data

m=100

#Generate adjacency matrix for first spatial nearest neighbors on 100x100 grid
#with boundary conditions

#the first element of A is an adjacency matrix; the second contains the locations
#of internal grid points
A=adjMat(m,m,boundary=TRUE)

#generate data and design matrix using example coefficient values
set.seed(42)

#some of the predictor values are zero: these variables are irrelevant to the response.
#In the model fitting step, setting select=TRUE will
#use the LASSO to attempt to select the relevant variables.
beta=cbind(c(-0.1,0.5,-0.5,0.4,-0.4,rep(0,7)))
eta=0.5

theta=c(beta,eta)

#X matrix with 12 predictors and 10000 observations for a 100x100 grid
X=cbind(rep(1,m^2),replicate(11,rnorm(m^2)))

#generate data for 100x100 grid with "0" boundary conditions
z1=simulateData(theta=theta,X=X,m,m,centered=FALSE,k=2,burn=30,draws=1,boundary=0)

#view responses with plotGrid
plotGrid(z1,m+2,m+2)

#model fitting
model1=MPLE(X,z1,A=A$A,innerIndices=A$innerIndices,nLambda=101,centered=FALSE,BIC=TRUE,select=TRUE)

#significance testing
pValues1=significanceTest(model1)


###### Example 2: Data with more than two response categories

#i) simulate data from the automultinomial (Potts) model with 4
#response categories on a 100x100 grid without boundary conditions

#ii) model fitting

m=100
```

```
#Generate adjacency matrix for first spatial nearest neighbors on 100x100 grid
A=adjMat(m,m,boundary=FALSE)$A

#generate data and design matrix using example coefficient values
set.seed(42)
#for multinomial data with k categories, beta and eta will have (k-1) columns.
#in this example, some of the rows of beta, corresponding
#to a single variable in the X matrix, are zero.
#These variables are irrelevant to the response. In the model fitting step,
#setting select=TRUE will use the group LASSO to attempt to select the relevant variables.

beta=cbind(c(0.1,0.2,0.4,-0.3,0.5,rep(0,7)),c(-0.5,0.5,-0.5,0.2,-0.4,rep(0,7)),
c(-0.3,0.2,0.7,-0.3,0.2,rep(0,7)))
#for multicategory data, eta[i,j] corresponds to the effect of neighbors of type i
#on the response probability of type j, treating the reference category as response type 0.
#eta is assumed to be a diagonal or symmetric (k-1) by (k-1) matrix.
eta=cbind(c(0.5,-0.1,-0.3),c(-0.1,0.4,0.1),c(-0.3,0.1,0.5))

theta=rbind(beta,eta)

#X matrix with 12 predictors and 10000 observations for a 100x100 grid
X=cbind(rep(1,m^2),replicate(11,rnorm(m^2)))

#generate data for 100x100 grid.
z2=simulateData(theta=theta,X=X,m,m,centered=FALSE,k=4,burn=30,draws=1)

#view responses with plotGrid
plotGrid(z2,m,m)

#model fitting

model2=MPLE(X,z2,A,nLambda=101,centered=FALSE,
BIC=TRUE,select=TRUE,standardize=TRUE,constraint="symmetric")

#significance testing
pValues2=significanceTest(model2)


## End(Not run)
```

`MPLEinternal` *Multinomial regression*

Description

Fits an automultinomial model (by pseudolikelihood). Fits the uncentered auto-model by default, #when an adjacency matrix is given. Performs an optional variable selection step, treating each #variable as a group for all response categories, with an option to group related variables.

Usage

```
MPLEinternal(X, z, A = NULL, groups = 2:dim(X)[2], nLambda = 101,
  centered = TRUE, BIC = TRUE, select = TRUE, standardize = TRUE,
  constraint, polish, innerIndices)
```

Arguments

`X`	nxp design matrix, where the first column must contain an intercept
`z`	nxk response indicator matrix, or a length n vector with the response type of each observation
`A`	an adjacency matrix or list of adjacency matrices defining neighborhood structures
`groups`	the grouping of the (non-intercept) coefficients. A vector of length (p-1)
`nLambda`	the length of the lasso path, defaults to 101
`centered`	logical, TRUE for centered and FALSE for uncentered, default is TRUE
`BIC`	logical, default TRUE uses BIC and FALSE uses AIC
`select`	logical, TRUE for variable selection step, default is TRUE
`standardize`	logical, standarize non-intercept columns? Default is TRUE, FALSE is almost certainly a bad idea
`constraint`	"symmetric" or "diagonal"
`polish`	use full Newton-Raphson to sharpen convergence from BFGS
`innerIndices`	indices of internal grid points

Value

a list containing the (group) lasso path,

Examples

```
#
```

`multinomialBCD` *Block coordinate descent*

Description

Computes, with group lasso regularization penalty, a lasso path for a model with user specified loglikelihood gradient and hessian.

Usage

```
multinomialBCD(X, z, groups, penaltyFactor, nLambda, H = NULL, theta = NULL)
```

Arguments

`X`	design matrix
`z`	response matrix
`groups`	a grouping of coefficients
`penaltyFactor`	a vector specifying the relative penalty level for each coefficient group
`nLambda`	number of penalty parameters on the lasso path
`H`	optional Hessian at loglikelihood maximum, used for the quadratic approximation in the centered model
`theta`	optional coefficients at loglikelihood maximum, used for the quadratic approximation in the centered model

Value

list containing the lambdas on the coordinate descent path and the coefficients for each lambda

Examples

```
#
```

`multProb` *Multinomial probabilities*

Description

Takes a design matrix and coefficients and computes the fitted probabilities for these coefficients.

Usage

```
multProb(X, beta)
```

Arguments

`X`	nxp design matrix
`beta`	a pxk coefficient matrix

Value

an nxk matrix of probabilities

Examples

```
#probs=multProb(X,beta)
```

plotGrid — *Visualize response values on a grid*

Description

Takes in an nxk matrix or a length n vector containing the response type of each observation and creates an image of the response for a specified grid size

Usage

```
plotGrid(z, m, n)
```

Arguments

z	nxk matrix or length n vector of responses, where z contains the responses of the grid in column-major order
m	the number of rows in the grid
n	the number of columns in the grid

Value

void

Examples

```
#plotGrid(z,m,n)
```

sandwichVariance — *Compute sandwich variance estimate*

Description

Computes the sandwich variance estimate for a given neighborhood structure and coefficient estimate

Usage

```
sandwichVariance(theta, X, z, A, centered, I = NULL, constraint, thetaInds,
  innerIndices)
```

Arguments

`theta`	a coefficient matrix
`X`	a design matrix
`z`	a matrix of multinomial responses
`A`	a list of adjacency matrices
`centered`	use centered or uncentered model
`I`	Fisher information matrix
`constraint`	constraint on eta
`thetaInds`	relevant indices
`innerIndices`	internal grid indices

Value

a covariance matrix

Examples

```
#
```

`significanceTest` *Significance tests for autologistic/automultinomial models*

Description

Performs two-sided hypothesis tests for each coefficient in a centered or uncentered autologistic model. The tests use the asymptotic normality of pseudolikelihood estimates with the Godambe sandwich variance estimator. A test for the spatial correlation eta is also performed.

Usage

```
significanceTest(model)
```

Arguments

`model`	a centered or uncentered autologistic fit

Details

For multinomial models, tests are performed for each row of the coefficient matrix, as each row corresponds to a single predictor. The matrix of eta coefficients is tested against the null hypothesis of no spatial correlation.

Value

a list of p-values and z-values

Examples

```
#model=MPLE(...)
#significanceTest(model)
```

---

simulateData *Simulate data*

Description

Simulates data from the centered or uncentered binomial or multinomial distribution

Usage

```
simulateData(theta, X, m, n, boundary = -1, centered = FALSE, k = 2,
  burn = 40, draws = 1)
```

Arguments

theta	a vector of coefficients for binary response, or a matrix of coefficients for multi-category response
X	n by p design matrix
m	width of the grid
n	height of the grid
boundary	(uncentered only) -1 for no boundary conditions, i>=0 to surround grid with class i observations
centered	logical, generates data from centered model if TRUE and uncentered if FALSE
k	number of response categories
burn	number of burn-in iterations for the Gibbs sampler
draws	number of simulated samples to return

Value

an nxk matrix of responses for draws=1, and a length draws list of nxk response matrices for draws>1

Examples

```
#generate outcomes on a grid using example coefficient values and design matrix
set.seed(42)
#for a multinomial response, beta and eta will have more than one column
#for a binary response, beta and eta will only have one column
#here, we the response variable takes 3 possible values, so beta and eta have two columns
beta=cbind(c(0,1.25,-1,2,-0.8),c(-3,0.25,-5,1,-0.4))

#setting the eta coefficients
```

```
eta=cbind(c(0.6,0),c(0,0.6))

#these are the final coefficient values we'll use to generate the data
theta=rbind(beta,eta)

#X matrix with 5 predictors and 900 observations for a 10x10 grid
X=cbind(rep(1,900),replicate(4,rnorm(900)))

#generate data for 30x30 grid
z=simulateData(theta,X,30,30,centered=TRUE,k=3,burn=1)
```

---

totalX *Add columns to X matrix*

Description

Adds the A

Usage

```
totalX(theta, X, z, A)
```

Arguments

`theta`	matrix of coefficients
`X`	design matrix
`z`	matrix of multinomial responses
`A`	list of adjacency matrices

Value

an expanded design matrix

Examples

```
#
```

uncenteredLogLikGrad *Uncentered gradient*

Description

Takes coefficient values and a design matrix and computes the gradient of the loglikelihood

Usage

```
uncenteredLogLikGrad(theta, X, z, A, constraint, thetaInds, innerIndices)
```

Arguments

theta	matrix of coefficients
X	an nxp design matrix
z	an nxk response matrix
A	list of adjacency matrices/list containing adjacency matrix
constraint	constraint on eta
thetaInds	relevant indices
innerIndices	the (1-dim.) location/index for internal points in the grid

Value

a gradient vector

Examples

```
#gradient=uncenteredLogLikGrad(theta,X,z,A,constraint,thetaInds)
```

uncenteredLogLikHess *Uncentered hessian*

Description

Takes coefficient values and a design matrix and computes the hessian matrix of the loglikelihood

Usage

```
uncenteredLogLikHess(theta, X, z, A, constraint, thetaInds)
```

Arguments

`theta`	matrix of coefficients
`X`	an nxp design matrix
`z`	response matrix
`A`	a list of adjacency matrices/list containing an adjacency matrix
`constraint`	constraint on eta
`thetaInds`	relevant indices

Value

a hessian matrix

Examples

```
#hess=uncenteredLogLikHess(theta,X,z,A)
```

`unStandardize` *Unstandardize coefficients*

Description

Converts coefficients on a standardized scale to the original scale

Usage

```
unStandardize(thetaList, means, sds, nRow)
```

Arguments

`thetaList`	a list of coefficient matrices
`means`	means of the non-intercept columns on the original scale
`sds`	standard deviations of the coefficients on the original scale
`nRow`	the number of coefficient rows that are affected by standardization and unstandardization

Value

a list of coefficient matrices on the original scale

Examples

```
#
```

Index