Package ‘hmmm’

October 17, 2007

Version 1.0.26

Date 2007-04-21

Title hierarchical multinomial marginal models

Author Roberto Colombi, Manuela Cazzaro

Maintainer Colombi Roberto <Colombi@unibg.it>

Description Functions to specify and fit hierarchical multinomial marginal models (HMMM), multinomial poisson homogeneous models (MPH) and homogeneous linear predictor models (HLP) for contingency tables. Inequality constraints on the parameters are allowed and can be tested.

Depends quadprog, MASS

License GPL Version 2 or later.

R topics documented:

`Marg.fct` *function to create marginalizing matrix*

Description

Suppose that y = y_ijk, where i=1,...,k1, j=1,...,k2, k=1,...k3, and the farther to the right the subscript the faster it changes (i.e. subscripts are in lexicographical order). This function creates an M (marginalizing) matrix such that My is the vector of marginal frequencies.

Usage

```
Marg.fct(margindex, levs)
```

Arguments

`margindex`	Collection of indices that will NOT be summed over (e.g. margindex=c(1,3) will give M13, see below.)
`levs`	Collection of levels (e.g. levs = c(3,3,5) means that k1=k2=3, k3=5).

Value

The marginalizing matrix.

Author(s)

Joseph B. Lang, Dept. of Stat. and Act. Sci., Univ. of Iowa (6/19/99, last update: 3/30/04).

Examples

```
M1 <- Marg.fct(1,c(3,3,3,3))
M2 <- Marg.fct(2,c(3,3,3,3))
M3 <- Marg.fct(3,c(3,3,3,3))
M4 <- Marg.fct(4,c(3,3,3,3))
M <- rbind(M1,M2,M3,M4)
```

`block.fct` *direct sum function*

Description

Function to create a block diagonal matrix

Usage

```
block.fct(...)
```

Arguments

`...`	Matrices of the direct sum

Value

Direct sum of matrices.

Author(s)

Joseph B. Lang, Dept. of Stat. and Act. Sci., Univ. of Iowa (6/16/99, last update: 3/30/04).

Examples

```
A <- factor(gl(4,4))
B <- factor(gl(4,1,16))
Ascore <- c(A)
Bscore <- c(B)
agree <- c(diag(1,4))

XA1 <- model.matrix(~A + B + Ascore*Bscore - Ascore-Bscore + agree)
XA2 <- XA1
X <- block.fct(XA1,XA2,diag(8))
```

`chibar.P` *Function to simulate chi-bar P values*

Description

This function simulates the weights and the P values of a chi-bar distribution for tests of type A and type B (Silvapulle and Sen, 2005) on inequality constraints on the parameters of HMMM, HLP, HPM models.

Usage

```
chibar.P(m, Z, ZF, d.fct = 0,
 h.fct = 0, test0 = 0, test1 = 0,
repli = 0, derdt.fct = 0, derht.fct = 0)
```

Arguments

`m`	Extimated expected counts.
`Z`	Population matrix - see the help of mphineq.fit.
`ZF`	Sample matrix - see the help of mphineq.fit.
`d.fct`	Inequality constraints function - see the help of mphineq.fit.
`h.fct`	Equality constraints function - see the help of mphineq.fit.
`test0`	Test statistics of type A - see Silvapulle and Sen, 2005.
`test1`	Test statistics of type B - see Silvapulle and Sen, 2005.
`repli`	Simulations number.
`derdt.fct`	Derivatives of inequality constraints - see the help of mphineq.fit.
`derht.fct`	Derivatives of equality constraints - see the help of mphineq.fit.

Details

The method "Simulation 2" described in Silvapulle and Sen, 2005, pg. 79 is used.

Value

A list with the test statistics of type A and B (Silvapulle and Sen, 2005, pg. 61) and their simulated P values.

Note

The function is not recommended, use chibar.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Silvapulle M.J., Sen P.K., Constrained statistical inference, Wiley, New Jersey (2005).

See Also

chibar, hmmm.chibar, chibar.summary

`chibar` *simulation of chi-bar P values*

Description

Function to simulate the weights and the P values of a chi-bar distribution for tests of type A and type B (Sen-Silvapulle, 2005) on inequality constraints on the parameters of HMMM, HLP, HPM models.

Usage

```
chibar(m, Z, ZF, d.fct = 0, h.fct = 0,
test0 = 0, test1 = 0, repli = 0, derdt.fct = 0,
 derht.fct = 0)
```

Arguments

`m`	Extimated expected counts.
`Z`	Population matrix - see the help of mphineq.fit.
`ZF`	Sample matrix - see the help of mphineq.fit.
`d.fct`	Inequality constraints function - see the help of mphineq.fit.
`h.fct`	Equality constraints function - see the help of mphineq.fit.
`test0`	Test statistics of type A - see Silvapulle and Sen, 2005.
`test1`	Test statistics of type B - see Silvapulle and Sen, 2005.
`repli`	Simulations number.
`derdt.fct`	Derivatives of inequality constraints - see the help of mphineq.fit.
`derht.fct`	Derivatives of equality constraints - see the help of mphineq.fit.

Details

The method "Simulation 2" described in Silvapulle and Sen, 2005, pg. 79 is used.

Value

A list with the the test statistics of type A and B (Silvapulle and Sen, 2005, pg. 61) and their simulated P values.

Note

Use chibar.summary to display the output, use hmmm.chibar for HMMM models.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Silvapulle M.J., Sen P.K., Constrained statistical inference, Wiley, New Jersey (2005).

See Also

chibar.P, hmmm.chibar,chibar.summary

Examples

```
y <-
c(56, 13, 7,
6, 4, 10,
1, 5, 15)
y<-matrix(y,9,1)
Z <- ZF <- matrix(1,9,1)
Lprobit.fct <- function(m) {
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
rbind(
qnorm(pr[2]+pr[3]),
qnorm(pr[3]),
qnorm(pc[2]+pc[3]),
qnorm(pc[3]),
log(m)
)
}
dprobit.fct <- function(m) {
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
-qnorm(pc[2]+pc[3])+qnorm(pr[2]+pr[3])

}

Xprobit <-
c(1, 0, 0,
0, 1, 0,
1 ,0 ,1,
0, 1 ,1)

Xprobit <- matrix(Xprobit,4,3,byrow=TRUE)
XHprobit<-Xprobit[,-3]

A <- factor(gl(3,3))
B <- factor(gl(3,1,9))
Xindep <- model.matrix(~A+B)
Xsat <- model.matrix(~A*B)

# FIT USING THE SATURATED ASSOCIATION MODEL
#Under ordering of the marginals

X <- block.fct(Xprobit,Xsat)

a <- mphineq.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X,d.fct=dprobit.fct,maxiter=3000 )

# FIT USING THE SATURATED ASSOCIATION MODEL

b <- mphineq.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X,maxiter=1000 )



#X <- block.fct(XHprobit,Xsat)

hprobit.fct<-function(m){
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
-(qnorm(pr[2]+pr[3])-qnorm(pc[2]+pc[3]))
+(qnorm(pr[3])-qnorm(pc[3]))
}
#test for inequalities
#t(create.U(X))

p<-chibar(y,Z=Z,ZF=ZF,d.fct=dprobit.fct,h.fct=hprobit.fct,test1=a$Gsq-b$Gsq,repli=6000)
c(p$testB,p$pvalB)
```

`chibar.summary` *output summary for chibar and hmmm.chibar functions*

Description

Function to print the results for tests of type A and B on inequality constraints and to tabulate the distribution functions of these two test statistics.

Usage

```
chibar.summary(P, plotflag = 0, step = 0.01, lsup = 0)
```

Arguments

`P`	Output from chibar or hmmm.chibar.

`plotflag`	If 0 only P-values are printed, if 2 the survival functions for type A and type B tests are also tabulated and finally if 3 are also plotted (in red: type B, in black: type A).
`step`	Distance between points at which the distribution functions are evaluated.
`lsup`	Distribution functions are evaluated in the interval 0 - lsup.

Value

Tabulation of the chi-bar distribution functions of the type A and B test statistics.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Silvapulle M.J., Sen P.K., Constrained statistical inference, Wiley, New Jersey (2005).

See Also

hmmm.chibar, chibar

`create.U` *function to create a matrix with column space equal to the null space of X*

Description

This function creates a full-column rank matrix, U, with column space equal to the orthogonal complement of the column space of X. That is, U has column space equal to the null space of the transpose of X.

Usage

```
create.U(X)
```

Arguments

`X`	Matrix of full column rank.

Value

Matrix that has column space equal to the null space of the transpose of X.

Author(s)

Joseph B. Lang, Dept. of Stat. and Act. Sci., Univ. of Iowa (6/19/99, last update: 3/30/04).

Examples

```
X <- c(
1, 1 ,1 ,1,
1, 1 ,0 ,0,
1 ,0, 1 ,0,
1 ,0 ,0, 0)

X <- matrix(X,4,4,byrow=TRUE)
X.indep <- X[,-4]

U <- create.U(X.indep)
```

`create.WCMAT` *function to create a design matrix X for models of independence in subtables*

Description

This function computes the X matrix corresponding to models of conditional independence (Cox Wermuth, 1998) or of Lumpability (Colombi Giordano, 2007)

Usage

```
create.WCMAT(modelfull, Selection, Subsets = NULL,
replace = TRUE,xmerge=NULL)
```

Arguments

`modelfull`	a model defined by hmmm.model
`Selection`	a list of lists like "list(NULL,list(1,0,c(1,3),c(1,2,3)),NULL)". There must be a sublist for every marginal set in modelfull. The sublist is NULL when in the corresponding marginal distribution no interactions are set to zero. 0 for a variable implies that the interactions involving this variable are not constrained In every non NULL sublist there are as many integer objects as variables in modelfull. Every integer object of a sublist contains indices of the interactions set to zero in the marginal corresponding to the sublist. More precisely in a marginal the interactions are set to zero when the indeces belong to the cartesian product of the numeric objects in the sublist. See the the details below.
`Subsets`	a list of logical objects like list(c(F,F,F),c(F,T,F,T),c(F,F,T,F,T,F,T,T)) There must be a logical object for every marginal set in modellfull. In the logical objects F correspond to interactions not to be constrained and T to interactions that must be set to zero according the list Selection See the details below .
`replace`	if TRUE a hmmm.model object is created otherwise a list is returned that contains the design matrix X and the positions of the interactions constrained to zero in the vector of all parameters of modelfull.
`xmerge`	an additional integer object for the positions of further interactions set to zero

Details

To the variables i1,i2,.,ij...,ik of a marginal set M are associated sets of indices S(i1),S(i2),...,S(ij),...,S(ik). A generalized marginal interaction defined by the interaction set I and the marginal set M is constrained to zero when its indices belong to the cartesian product of the S(ij) ,for all ij in M. The interaction defined in M that must be constrained according the previous rule are specified by the argument Subsets. Using xmerge further interactions can be set to zero by reporting their positions in the vector off all parameters of modelfull.

Value

a hmmm.model or a design matrix X together with the positions of the interactions set to zero

Author(s)

Roberto Colombi Colombi@unibg.it

References

Wermuth, Cox (1998): On the application of Conditional Independence to Ordinal Data Internetional statistical Review, 66,181-199

Cazzaro M.,Colombi R., Giordano S., (2007):Testing Markov Chain Lumpability, Proceedings of the 22st IWSM, Barcelona

See Also

hmmm.model,create.XMAT

Examples

```
marginal34<-list(marg=c(3,4),types=c("marg","marg","l","l"))
marginal134<-list(marg=c(1,3,4),types=c("r","marg","l","l"))
marginal1234<-list(marg=c(1,2,3,4),types=c("r","r","l","l"))
marginali<-list(marginal34,marginal134,marginal1234)

models<-hmmm.model(marg=marginali,lev=c(4,4,4,4),
strata=1,
cocacontr=list(matrix(c
(1,1,0,0,
 0,0,1,0,
 1,0,0,0,
 0,0,1,1,
 0,0,0,1,
 0,1,0,0
)
,6,4,byrow=TRUE),
matrix(c
(1,1,0,0,
 0,0,1,0,
 1,0,0,0,
 0,0,1,1,
 0,0,0,1,
 0,1,0,0
)
,6,4,byrow=TRUE)
)
```

```
)

descr<-hmmm.model.summary(models)

Selvec<-list(NULL,list(1,0,c(1,3),c(1,2,3)),NULL)
Sub<-list(c(FALSE,FALSE,FALSE),c(FALSE,TRUE,TRUE,TRUE),c(FALSE,FALSE,FALSE,FALSE,FALSE,FA
MoNEW3<-create.WCMAT(models,Selvec,Subsets=Sub,replace=FALSE)

Selvecc<-list(NULL,list(1,0,c(1,3),c(1,2,3)),list(1,c(1,2,3),c(1,3),c(1,2,3)))
scc<-list(c(FALSE,FALSE,FALSE),c(FALSE,TRUE,FALSE,TRUE),c(FALSE,FALSE,TRUE,FALSE,TRUE,FAL

MoNEW4<-create.WCMAT(models,Selvecc,Subsets=scc,replace=FALSE)
Selve<-list(NULL,list(c(1,2,3),0,c(1,2,3),c(1,2,3)),NULL)
Sub<-list(c(FALSE,FALSE,FALSE),c(FALSE,FALSE,TRUE,TRUE),c(FALSE,FALSE,FALSE,FALSE,FALSE,F
MoNEW2<-create.WCMAT(models,Selve,Subsets=Sub,replace=FALSE)
MoNEW2$del
MoNEW3$del
MoNEW4$del
```

`create.XMAT` *design matrix for an hmmm model*

Description

Function to specify the matrix X of the linear predictor C*lnM*m=X*beta for a hmmm model

Usage

```
create.XMAT(modello, Formula = NULL,
strata = 1, fnames = NULL, cocacontr = NULL,
ncocacontr = NULL, replace = TRUE)
```

Arguments

`modello`	Object created by hmmm.model.
`Formula`	List of model formula enclosed in ""; one formula for every marginal interaction.
`strata`	Number of categories of the factors that describe the strata.
`fnames`	Names of the factors that describe the strata.
`cocacontr`	List that describes the contrasts for every factor.
`ncocacontr`	Number of contrasts for every factor if NULL the maximum number is used.
`replace`	If TRUE X a new model object is produced if FALSE only the matrix X is returned.

Details

The list of model-formula must be like: list(" 1+f_0*area*age-f_0:area:age-area:age", " 1+f_0*area*age-f_0:area:age-area:age", " 1+f_0+area+age") where f_0 is a conventional name used to identify the factors that are associated to every marginal interaction type and where area and ag are the names of the factors that describe the strata.

For example in the case of logits defined in two marginals and a set of odds ratios the first two formula state for every marginal an additive effect of the covariates specific for every logit and the third formula states an additive effect common to every o.r.

The list of contrasts must be like list("contr.poly",matrix(c(-1,-1,1,1),4,1)), in other words it can be formed with matrices and standard R contrast specifications. If not supplied contr.treatment is used for every factor.

Use the formula "zero" to constrain to zero all the interactions of a given type.

Value

X matrix or a marginal model object (see the argument replace).

Author(s)

Roberto Colombi, Colombi@unibg.it.

See Also

hmmm.model, hmmm.model.summary

Examples

```
remove(list=ls())
library(hmmm)

######################################################
# Example from:
# Cazzaro M., Colombi R., A new family of interactions
# for modelling contingency tables, (2006), submitted.
######################################################

#============================================================
# MODELLING the effect of COVARIATES on RECURSIVE INTERACTIONS
#============================================================

# TABLE 3 - section 7.5:
# four way contingency table --> TYPE x TIME, given AGE and HOUR,
# accidents occured to workers.

# RESPONSE VARIABLES:
#variable n. 1-TYPE of the injury --> 3 categories:
#                        U=uncertain, A=avoidable, NA=not avoidable;
# variable n. 2-TIME to recover --> 4 categories, number of working days lost:
#                      0 |- 7, 7 |- 21, 21 |- 60, >= 60;

# COVARIATES:
#variable n.3 AGE of the worker --> 3 categories:
#                       <= 25, 26 - 44, >= 45;
# variable n.4 solar HOUR --> 2 categories, part of the day in which the accident occurec
#               M=morning, A=afternoon.
 # the lower the variable number is the faster the variable sub-scripit
# changes in the vectorized contingency table
y <- c(
 21,  9,  0, 10,  9,  0,  5,  1,  1,  2,  0,  1,
 78, 51,  1, 46, 28,  5, 15, 21, 10, 10,  4, 12,
 27, 16,  1, 25, 19,  2, 13, 16,  8,  1,  7, 11,
 35, 14,  1, 10,  7,  1,  3,  2,  3,  2,  1,  1,
104, 40,  1, 46, 40,  4, 21, 16,  8, 13,  8,  8,
 39, 23,  1, 29, 14,  2, 17, 12,  8,  6, 10, 16)

# univariate marginals
# logit type:
# l --> local,  g --> global,
# c --> cont., rc --> reverse cont.,
# r --> recursive
marg1<-list(marg=c(1),types=c("c","marg"))
marg2<-list(marg=c(2),types=c("marg","c"))

# bivariate marginals
marg12<-list(marg=c(1,2),types=c("c","c"))

marginals<-list(marg1,marg2,marg12)



# definition of the model
models<-hmmm.model(marg=marginals,lev=c(3,4),
strata=6
)

descr<-hmmm.model.summary(models,printflag=TRUE)

# estimation of the models

# REFERENCE model
al<-list(
"~f_0*AGE*HOUR",
"~f_0*AGE*HOUR",
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR")

modelref<-create.XMAT(models,Formula=al,strata=c(3,2),
fnames=c("AGE","HOUR"))



# model of uncostrained marginal and covariate (AGE, HOUR) additive effect on
# the o.r. between TYPE and TIME
al<-list(
"~f_0*AGE*HOUR",
"~f_0*AGE*HOUR",
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR"
)

model1<-create.XMAT(models,Formula=al,strata=c(3,2),
fnames=c("AGE","HOUR"))


# model of covariate (AGE, HOUR) additive effect on the marginal logits
# of the response variables (TYPE, TIME) and on the o.r. between TYPE and TIME
al<-list(
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR")

model2<-create.XMAT(models,Formula=al,strata=c(3,2),
fnames=c("AGE","HOUR"))


# model of covariate (AGE, HOUR) additive effect on the marginal logits
# of the response variables (TYPE, TIME) and constant association between TYPE and TIME
al<-list(
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"~f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"~f_0")

model3<-create.XMAT(models,Formula=al,strata=c(3,2),
fnames=c("AGE","HOUR"))



# model of stochastic independence between TYPE and TIME
# in each sub-table identified by the levels of the covariates AGE and HOUR
alind<-list(
"~1+f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"~1+f_0+AGE+HOUR+f_0:AGE+f_0:HOUR",
"zero")

modelind<-create.XMAT(models,Formula=alind,strata=c(3,2),
fnames=c("AGE","HOUR"))
```

`hmmm.CoxTest` *Cox Test for non-nested hmmm models*

Description

This function computes the Cox Test the Gourieroux Monfort score test and the Vuong testfor non nested hypotheses. P values are also provided

Inequalities are not allowed

Usage

```
hmmm.CoxTest(modelA, modelB, fitA, fitB=NULL, pseudtrue = FALSE,
pseudtrue.w=FALSE,score.test=FALSE, maxit = 1000, y.eps = 0)
```

Arguments

`modelA`	null hmmm.model
`modelB`	non nested alternative hmmm.model
`fitA`	modelA fitted
`fitB`	modelB fitted, if NULL only the score test is computed
`pseudtrue`	if TRUE pseudotrue values for modelB are estimated if FALSE ML estimates in fitB are used

`pseudtrue.w`	if FALSE pseudotrue values for modelB are not used in the denominator of the Cox statistics
`score.test`	if TRUE the Gourieroux, Monfort, Trognon score test for non nested hypotheses is also computed
`maxit`	max number of iterations to compute estimates of pseudo true values
`y.eps`	see hmmm.mlfit

Value

A list with components CoxTest VuongTest and ScoreTest. The Cox test,the score test and the p values are returned together with the LR statistics of model A and model B. The vuongTest is returned with the p value

...

Author(s)

Colombi@unibg.it

References

C. Gourieroux, A. Monfort (1989): Statistics and Econometric Models

---

`hmmm.chibar.P` *chi-bar P values for hmmm models*

Description

Function to simulate the weights and the P values of a chi-bar distribution. The function arguments must be objects created by hmmm.mlfit and hmmm.model.

Usage

```
hmmm.chibar.P(model, nullfit, disfit, satfit, repli = 6000)
```

Arguments

`model`	The model created by hmmm.model.
`nullfit`	The estimated model with inequalities turned into equalities.
`disfit`	The estimated model with inequalities.
`satfit`	The estimated model without inequalities.
`repli`	Number of simulations.

Details

The method "Simulation 2" described in Silvapulle snd Sen, 2005, pg. 79 is used.

Value

A list with the test statistics of type A and B (Silvapulle and Sen, 2005, pg. 61) and their simulated P values.

Note

The function is not recommended, use hmmm.chibar.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Silvapulle M.J., Sen P.K., Constrained statistical inference, Wiley, New Jersey (2005).

See Also

chibar, hmmm.chibar, chibar.summary

`hmmm.chibar` *chi-bar P values for hmmm models*

Description

Function to simulate the weights and the P values of a chi-bar distribution. The function arguments must be objects created by hmmm.mlfit and hmmm.model.

Usage

```
hmmm.chibar(model, nullfit, disfit, satfit, repli = 6000)
```

Arguments

`model`	The model created by hmmm.model.
`nullfit`	The estimated model with inequalities turned into equalities.
`disfit`	The estimated model with inequalities.
`satfit`	The estimated model without inequalities.
`repli`	Number of simulations.

Details

The method "Simulation 2" described in Silvapulle and Sen, 2005, pg. 79 is used.

Value

A list with the test statistics of type A and B (Silvapulle and Sen, 2005, pg. 61) and their simulated P values.

Note

Use chibar.summary to display the output.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Silvapulle M.J., Sen P.K., Constrained statistical inference, Wiley, New Jersey (2005).

See Also

chibar, chibar.summary

Examples

```
#===============================================
# MODELS with INEQUALITY CONTRAINTS on the o.r.
#       - MONOTONE DEPENDENCE HYPOTHESES -
#===============================================

# TABLE 1 - section 7.1:
# two way contingency table --> CUSTOMER SERVICE x TRAINING,
# 377 customers, users of a machine tool.
# (4 categories each, levels of satisfaction:
#  U=unsatisfied, S=satisfied, RS=really satisfied, ES=extremely satisfied).
library(hmmm)
y <- c(
 9, 10, 22, 25,
 3,  4, 10,  4,
12, 15, 81, 37,
11,  6, 31, 97)

Z <- ZF <- matrix(1,16,1)


############################################################
# LIKELIHOOD RATIO monotone dependence: log-local o.r. >= 0
############################################################

# identity design matrix 15 x 15
# NB: no constraints on the marginal logits
XX<-block.fct(diag(1,6),diag(1,9))

# univariate marginals
# logit type:
# l --> local,  g --> global,
# c --> cont., rc --> reverse cont.,
# r --> recursive
# NB: in this case the univariate marginals can be of any type
# as they are not constrained.
marginal1<-list(marg=c(1),int=list(1),types=c("l","marg"))
marginal2<-list(marg=c(2),int=list(2),types=c("marg","l"))

# bivariate marginals
marginal12<-list(marg=c(1,2),int=list(c(1,2)),types=c("l","l"))

marginals<-list(marginal1,marginal2,marginal12)

# marginal list involved in the INEQUALITIES
dism<-list(marginal12)

# definition of the model
```

```
models<-hmmm.model(marg=marginals,dismarg=dism,lev=c(4,4),
cocacontr=NULL,strata=1,Z=Z,ZF=ZF,X=XX)

descr<-hmmm.model.summary(models,printflag=TRUE)

# estimation of the models

# SATURATED model
asat<-hmmm.mlfit(y,models)

descrfitsat<-hmmm.model.summary(models,asat,aname="Saturated model")

# model with INEQUALITIES on the log-local o.r., no EQUALITY constraints
# on the univariate marginal logits: "Likelihood ratio monotone dependence model"
a <- hmmm.mlfit(y,models,noineq=FALSE)

descrfitdis<-hmmm.model.summary(models,a,aname="Model with inequalities")

# model with INEQUALITIES turned into EQUALITIES, no EQUALITY constraints
# on the univariate marginal logits: "Stochastic independence model"
XX0<-rbind(diag(1,6),matrix(0,9,6))

models0<-hmmm.model(marg=marginals,lev=c(4,4),
cocacontr=NULL,strata=1,Z=Z,ZF=ZF,X=XX0)

anull <- hmmm.mlfit(y,models0)

descrfitnull<-hmmm.model.summary(models0,anull,aname="Independence model")

# HYPOTHESES TESTED:
# NB: testA --> H0=(anull model) vs H1=(a model)
#     testB --> H0=(a model) vs H1=(asat model)

P<-hmmm.chibar(model=models,nullfit=anull,disfit=a,satfit=asat,repli=6000)

chibar.summary(P,plotflag=0)
```

`hmmm.mlfit` *hmmm.mlfit*

Description

Function to estimate a hierarchical multinomial marginal model.

Usage

```
hmmm.mlfit(y, model, noineq = TRUE, maxit = 1000,
norm.diff.conv = 1e-05, norm.score.conv = 1e-05,
 y.eps = 0, iter.orig = 5, chscore.criterion = 2,
 m.initial = y, mup = 1, step = 1)
```

Arguments

`y`	A vectorized sample contingency table.
`model`	An object created by hmmm.model.
`noineq`	If TRUE inequality constraints specified in model are ignored.
`maxit`	Maximum number of interactions.
`norm.diff.conv`	Convergence criterium value on the parameters.
`norm.score.conv`	Convergence criterium value on the constraints.
`y.eps`	Non-negative constant to be added to the original counts in y (default: y.eps=0).
`iter.orig`	Not used.
`chscore.criterion`	If equal to zero convergence informations are printed at every iteration.
`m.initial`	Initial estimate of m (default: m.initial=y).
`mup`	Weight for the constraints penality part of the merit function.
`step`	Interval lenght for the line search.

Details

A sequential quadratic procedure is used to maximize the log-likelihood under inequality and equality constraints. This function call the procedure mphineq.fit which is a generalization of the procedure mph.fit of Lang (2004).

Value

An object that can be displaied using hmmm.model.summary or mph.summary see the help of these functions.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Lang J.B. (2004): Multinomial Poisson Homogeneous models for contingency tables; The Annals of Statistics, 32, 340-383.

Bartolucci F., Colombi R., Forcina A. (2007): An extended class of marginal link functions for modelling contingency tables by equality and inequality constraints, Statistica Sinica, to appear.

See Also

hmmm.model, hmmm.model.summary , mph.summary

Examples

```
#variables
#1 jobs, three categories
#2 education, three categories
#3 age,two categories
#4 area, four categories
```

```
#the lower the variable number is the faster the category  sub-script changes in
#the vectorized table

library(hmmm)
y<-
 c(413, 3184, 2281 ,   1  , 42 ,  28 ,   0,   12,    5 ,
  79,  197,   95,    1,    3,    2,    0,    0,    0,   52,
  330,  274,    2,    9 ,  12 ,   0,    5,    5 ,  15,   21,
   16,    1,    0 ,   2,    0,    1 ,   0 , 523, 2945,
 1632 ,  38,  210,  103,   14,   88,   42,  113,  302 ,  81,
    7,   11,    2,    2,    3 ,   0,  560, 2330, 1365,
  49,  139,   81 ,  19,   64,   24,  116,  253,   56,    6,
    8,    0,    1,    0,    1)

Z<-ZF<-kronecker(diag(1,8),matrix(1,9,1))
marg1<-list(marg=c(2),int=list(2),types=c("marg","c"))
marg12<-list(marg=c(1,2),int=list(c(1),c(1,2)),types=c("g","l"))
marginali<-list(marg1,marg12)
marg12bis<-list(marg=c(1,2),int=list(c(1,2)),types=c("g","l"))
dism=list(marg12bis)

modello<-hmmm.model(marg=marginali,lev=c(3,3),dismarg=dism,
strata=8,Z=Z,ZF=ZF,D=diag(-1,32))
descr<-hmmm.model.summary(modello,printflag=FALSE)
al<-list(
"~1+f_0*area*ag-f_0:area:ag-area:ag",
"~1+f_0*area*ag-f_0:area:ag-area:ag",
"~1+f_0+area*ag"
)

modellosat<-create.XMAT(modello,Formula=al,strata=c(2,4),
fnames=c("ag","area"))

mods<-hmmm.mlfit(y,modellosat,y.eps=0.00001,maxit=2000,mup=1,step=1,
m.initial=y)


moddis<-hmmm.mlfit(y,modellosat,y.eps=0.00001,maxit=1000,mup=1,step=1,
m.initial=mods$m,noineq=FALSE,norm.diff.conv=1e-5)
descrfitsat<-hmmm.model.summary(modellosat,mods,aname="modello saturo")
descrfitdis<-hmmm.model.summary(modellosat,moddis,aname="modello con diseg.")
```

`hmmm.model` *define an hmmm model*

Description

Function to define a hierarchical multinomial marginal model object

Usage

```
hmmm.model(marg=NULL, dismarg = 0, lev, cocacontr = NULL,
 strata = 1, Z=NULL, ZF=Z, X = 0, D = TRUE, E = TRUE)
```

Arguments

`marg`	A list of the marginal sets and their marginal interactions as described in Colombi, Bartolucci and Forcina (2007). Every element of marg is a list with elements marg: the marginal set, int: the interaction set and type which describes the logits used for every variable ("g"=global, "l"=local ,"c"=continuation, "rc"=reverse continuation, "r"=recursive, "b"=baseline, "marg" for the variables not belonging to the marginal set). This list is used to create the link function C*ln(M*m) and its derivative.
`dismarg`	Similar to marg but used to define inequalities K*ln(A*m)>0. Default 0 if there are no inequalities.
`lev`	Number of categories of the variables.
`cocacontr`	Needed only for "r" and "b" logits; list of matrices of contrasts for "r" and "b"logits. Arbitrary (es: 0) for other logits.
`strata`	Number of strata.
`Z`	Zero one matrix describing the strata.
`ZF`	Zero one matrix for strata with fixed nobs.
`X`	Design matrix for C*ln(M*m)=X*beta. Can be defined later or changed only by using the function create.XMAT.
`D`	If is a matrix the inequalities will be D*K*ln(A*m)>0. Useful for changing the sign of the inequalities or for selecting a subset of K*ln(A*m)>0.
`E`	If E is a matrix and if X = 0 defines the equality contrasts E*C*ln(M*m)=0.

Details

Every element of marg is like:

list(marg=c(1,2),int=list(c(1),c(1,2)),types=c("g","l"))

where marg is the marginal set, int is the list of interaction sets and type describe the logits used for every variable ("g"=global, "l"=local ,"c"=continuation, "rc"=reverse continuation, "r"=recursive, "b"=baseline ,"marg" for the variables not belonging to the marginal set). Variables are denoted by integers, the variable with lower integer must have the modality index that runs faster in the vectorized contingency table. If the interaction lists int are not given then a complete hierarchical marginal set of interactions is defined as described in Bartolucci, Colombi and Forcina (2007) and in Bergsma and Rudas (2002). In this case the order of the marginal sets in the marg list is relevant and in a marginal distribution are defined all the interactions that are not defined in previous marginal sets Every marginal set must not be a subset of previous marginal sets. If marg is not specified the Glonek and McCullagh (1995) Multivariate Logit Model with interactions of type local is used.

Z is of dimension cxK, where c is the number of counts and K is the number of strata or populations. Thus, the rows correspond to observation numbers and the columns correspond to the strata. A '1' in row i of column j implies that the ith count came from the jth stratum. Note that Z will have exactly one '1' in each row, and at least one '1' in each column. The population matrix Z which is a column vector of '1's, implies that all the counts came from the same, and only, stratum. For HMMM models it is assumed that all the strata have the same number of response levels. If Z is not entered a population Z matrix corresponding to data entered by strata is defined and ZF=Z.

For non-zero ZF, the columns are a subset of the columns in population matrix Z. If the jth column of Z is included in ZF, then the jth stratum sample size is considered fixed, otherwise, if the jth column of Z is NOT included in ZF, the jth stratum sample size is taken to be a realization of a Poisson random variable. ZF=Z, means that all of the stratum sample sizes are fixed; this is the (product-)multinomial setting.

When X is omitted it is assumed to be the identity matrix and a saturated model is defined.

The argument cocacontr must be equal to a list of zero-one matrices; one for every variable of the joint distribution. If the j-th variable has r_j categories, the first r_j-1 rows of the j-th matrix must describe the indicator functions of the r_j-1 category sets, the probabilities of which are the numerators of the logits and the remaining r_j-1 rows must describe the indicator functions of the sets the probabilities of which are the denominators of the logits. These matrices may be arbitrary (as 0) for variables to which logits "r" and "b" are never assigned. In particular cocacontr=NULL if logits "b" or "r" are never used.

Value

An object that describes a marginal model that can be estimated by hmmm.mlfit.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Bergsma W.P. and Rudas T. (2002): Marginal models for categorical data, The Annals of Statistics, 30, 140-159.

Glonek G.F.V. and McCullagh P. (1995): Multivariate logistic models for Contingency Tables, Journal of the Royal Statistical Society, B, 57, 533-546.

Lang J.B. (2004): Multinomial Poisson Homogeneous models for contingency tables, The Annals of Statistics, 32, 340-383.

Bartolucci F., Colombi R., Forcina A. (2000): An extended class of marginal link functions for modelling contingency tables by equality and inequality constraints, Statistica Sinica, to appear.

See Also

hmmm.mlfit,create.XMAT, hmm.model.summary, mph.summary

Examples

```
#==================================================
# MODEL of STOCHASTIC INDEPENDENCE in SUB-TABLES
#==================================================

# TABLE 2 - section 7.4:
# two way contingency table --> GROWTH x COMPETITIVENESS,
# (114 companies belonging to the manufacturing industry)
# GROWTH --> 4 categories, percentage og growth:
#             L=low, ML=medium/low, MH=medium/high, H=high;
# COMPETITIVENESS --> 3 categories, level of competitiveness
#                      L=low, M=medium, H=high.

y <- c(
 4, 11, 14,  4,
17, 22, 13,  2,
 6, 10,  6,  5)

# NB
# variable GROWTH --> recursive approach,
# the non-minimal sets are: M1=(L,ML,MH,H); M2=(MH,H); M3=(L,ML);
# variable COMPETITIVENESS --> continuation approach.

# identity design matrix 15 x 15
# NB: no constraints on the marginal logits
XX<-diag(1,11)

# univariate marginals
# logit type:
# l --> local,  g --> global,
# c --> cont., rc --> reverse cont.,
# r --> recursive
# NB: in this case the univariate marginals can be of any type
# as they are not constrained.
marginal1<-list(marg=c(1),types=c("r","marg"))
marginal2<-list(marg=c(2),types=c("marg","c"))

# bivariate marginals
marginal12<-list(marg=c(1,2),types=c("r","c"))

marginals<-list(marginal1,marginal2,marginal12)

# "COCACONTR" specifications
# ===========================
# NB: 'cocacontr' is needed only for recursive 'r' or baseline 'b' logits.
#     It is a list of matrices of constrasts for 'r' and 'b' logits; it follows
#     the natural order of the variables. Arbitrary (for example 0) for
#     other logits.

#  For example: variable A with categories a1, a2, a3, a4;
#  the non-minimal sets M1, M2, M3 could be as follows:
#
#     _____|M1___
#     |         |
#   __|M3__   __|M2__
#   |     |   |     |
#   a1   a2   a3   a4
#
# then 'cocacontr' should be
#       cocacontr=list(
#                    matrix(c(1,1,0,0,
#                             0,0,1,0,
#                             1,0,0,0,
#                             0,0,1,1,
#                             0,0,0,1,
#                             0,1,0,0),6,4,byrow=TRUE)
#                                                     ,0),
#
# as the first 3 rows represent the 'LEFT children' of the node Mm, m=1,2,3
#               1 1 0 0  --> M1
#               0 0 1 0  --> M2
#               1 0 0 0  --> M3
```

```
#
# and the last 3 rows represent the 'RIGHT children' of the node Mm, m=1,2,3
#               0 0 1 1  --> M1
#               0 0 0 1  --> M2
#               0 1 0 0  --> M3

# NB: The marginal logits are builded as follows:
#     --> eta1 on the M1 node
#     --> eta2 on the M2 node
#     --> eta3 on the M3 node

# ===========================


# definition of the model
models<-hmmm.model(marg=marginals,lev=c(4,3),
cocacontr=list(matrix(c
(1,1,0,0,
 0,0,1,0,
 1,0,0,0,
 0,0,1,1,
 0,0,0,1,
 0,1,0,0
)
,6,4,byrow=TRUE),0
),
strata=1,X=XX)

descr<-hmmm.model.summary(models)
# model with all the log-recursive o.r. =0 except for o.r.(1,1),
# that is "Model of stochastic independence in sub-tables"
# (except for the table referring to the first node of each hierarchy of sets)
XX<-diag(1,11)
XX<-XX[,-c(7,8,9,10,11)]

modelsor<-hmmm.model(marg=marginals,lev=c(4,3),
cocacontr=list(matrix(c
(1,1,0,0,
 0,0,1,0,
 1,0,0,0,
 0,0,1,1,
 0,0,0,1,
 0,1,0,0
)
,6,4,byrow=TRUE),0
),
strata=1,X=XX)
```

`hmmm.model.summary` *hmmm model summary*

Description

Function to print informations on the interactions of a fitted hierarchical multinomial marginal model estimated by hmmm.mlfit or a model defined by hmmm.model.

Usage

```
hmmm.model.summary(modelfull, fitmod = NULL, printflag = TRUE,

 aname = "modfit")
```

Arguments

`modelfull`	A model object created by hmmm.model.
`fitmod`	Model fitted by hmmm.mlfit.
`printflag`	If TRUE the output is printed otherwise returned without printing.
`aname`	Title for the output.

Details

The output changes according if the number of strata is >1 or not and if fitmod is NULL or not.

Value

A matrix containing informations on the marginal interactions and also of their estimates if fitmod is not NULL.

Author(s)

Roberto Colombi, Colombi@unibg.it.

Examples

```
#==============================================
# MODELS with INEQUALITY CONTRAINTS on the o.r.
#   - DOUBLE MONOTONE DEPENDENCE HYPOTHESES -
#==============================================

# TABLE 1 - section 7.3:
# two way contingency table --> CUSTOMER SERVICE x TRAINING,
# 377 customers, users of a machine tool.
# (4 categories each, levels of satisfaction:
#  U=unsatisfied, S=satisfied, RS=really satisfied, ES=extremely satisfied).

y <- c(
 9, 10, 22, 25,
 3,  4, 10,  4,
12, 15, 81, 37,
11,  6, 31, 97)

Z <- ZF <- matrix(1,16,1)



############################################################
# DOUBLE UNIFORM monotone dependence:
# log-continuation-local & log-local-continuation o.r. >= 0
############################################################

# identity design matrix 15 x 15
# NB: no constraints on the marginal logits
XX<-block.fct(diag(1,6),diag(1,9))

# univariate marginals
# logit type:
# l --> local,  g --> global,
# c --> cont., rc --> reverse cont.,
# r --> recursive
# NB: in this case the univariate marginals can be of any type
# as they are not constrained.
marginal1<-list(marg=c(1),int=list(1),types=c("l","marg"))
marginal2<-list(marg=c(2),int=list(2),types=c("marg","l"))

# bivariate marginals
marginal12<-list(marg=c(1,2),int=list(c(1,2)),types=c("l","l"))

marginals<-list(marginal1,marginal2,marginal12)

# marginal list involved in the INEQUALITIES
marginal12dis<-list(marg=c(1,2),int=list(c(1,2)),types=c("c","l"))
marginal12bis<-list(marg=c(1,2),int=list(c(1,2)),types=c("l","c"))
dism<-list(marginal12dis,marginal12bis)

# D is the matrix that involves the INEQUALITIES
D<-diag(1,18)

# deletion of REDUNDANT CONSTRAINTS
# NB: it is necessary to know which they are...
D<-D[-c(7,8,9,12,15),]

# definition of the model
models<-hmmm.model(marg=marginals,dismarg=dism,lev=c(4,4),
cocacontr=NULL,strata=1,Z=Z,ZF=ZF,X=XX,D=D)

descr<-hmmm.model.summary(models,printflag=TRUE)

# estimation of the models

# SATURATED model
asat<-hmmm.mlfit(y,models)

descrfitsat<-hmmm.model.summary(models,asat,aname="Saturated model")

# model with INEQUALITIES on the log-c-l o.r. and log-l-c o.r., no EQUALITY constraints
# on the univariate marginal logits: "Double uniform monotone dependence model"
a <- hmmm.mlfit(y,models,noineq=FALSE)

descrfitdis<-hmmm.model.summary(models,a,aname="Model with inequalities")

# model with INEQUALITIES turned into EQUALITIES, no EQUALITY constraints
# on the univariate marginal logits: "Stochastic independence model"
XX0<-rbind(diag(1,6),matrix(0,9,6))

models0<-hmmm.model(marg=marginals,lev=c(4,4),
cocacontr=NULL,strata=1,Z=Z,ZF=ZF,X=XX0)

anull <- hmmm.mlfit(y,models0)
```

```
descrfitnull<-hmmm.model.summary(models0,anull,aname="Independence model")
```

`loglin.model` *log-linear models specification*

Description

This function can be used to specify a hierarchical log-linear model.

Usage

```
loglin.model(lev, int = NULL, strata = 1, dismarg = 0, type = "b",
D = TRUE, c.gen=TRUE,printflag=TRUE)
```

Arguments

`lev`	Numbers of categories of the variables.
`int`	Generating class of the log-linear model (must be a list) or list of all the interactions included.
`strata`	Number of strata.
`dismarg`	List of interactions constrained by inequalities - see hmmm.model.
`type`	"b" for baseline logits "l" for local logits.
`D`	See the help of hmmm.model.
`c.gen`	If FALSE int must be the list of the minimal interaction sets to be excluded.
`printflag`	If TRUE informations on the included and excluded interactions are given.

Details

This function performs the same task of hmmm.model but it is easier to use in the case of loglinear models. The model can be estimated by hmmm.mlfit.

Value

An object that describes a log-linear model that can be estimated by hmmm.mlfit.

Note

If int is not supplied a saturated log-linear model is defined. For log-linear models where the parameters depend on covariates first define a saturated log-linear model and then use the function create.XMAT.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Agresti A., Categorical data Analysis, (2ed), Wiley, New York (2002).

See Also

hmmm.model, hmmm.mlfit, hmmm.model.summary, mph.summary

Examples

```
d<-hmmm.model.summary(loglin.model(c(3,4,2),list(c(1,2),c(1,3),c(2,3))))


m<-loglin.model(lev=c(2,2,2),int=list(c(1,3),c(2,3),c(1,2)),c.gen=FALSE)
 m$matrici$X
dm<-hmmm.model.summary(m)
```

`marg.list` *lists of marginal sets*

Description

A friendly way to define the first argument marg of the function hmmm.model that is the list of marginal sets used to specify an hmmm model.

Usage

```
marg.list(all.m, sep = "-", mflag = "marg")
```

Arguments

`all.m`	Must be a character vector, One rows for every marginal set. Every row must be like: "marg-g-c" - see the details below.
`sep`	The separator used between logits type; default "-".
`mflag`	The symbol used to denote variables that are marginalized. Default "marg".

Details

A row of all.m is a string defining a marginal set and the logits types that must be used to build the interactions The variables not included in the marginal set are denoted by mflag. The variables in the marginal set are denoted by a symbol that identifies a logit type ("b" baseline, "g" global, "c" continuation, "rc" reverse continuation, "r" recursive, "l" local). Symbols are separated by sep.

Value

A list marg that can be used as the first argument in hmmm.model; see the help of this function.

Note

The function marg.list does not create the list of the interactions that must be defined in a marginal set. So the function hmmm.model will use the hierarchical approach described in the help of this function. In particular (because of what just said) marg.list cannot be used to define the interactions subject to inequality constraints. See the help of hmmm.model.

Author(s)

Roberto Colombi, Colombi@unibg.it.

See Also

hmmm.model

Examples

```
library(hmmm)
 mm<-c("m-m-g","m-g-m","g-m-m","l-l-l")
 marg<-marg.list(mm,mflag="m")
 mod<-hmmm.model(marg=marg,lev=c(3,3,3))
 model.description<-hmmm.model.summary(mod)
```

`mph.fit` *mph models estimation*

Description

This function computes ML estimates of a mph model by using the AS algorithm

Usage

```
mph.fit(y, Z, ZF = Z, h.fct = 0, derht.fct = 0,
L.fct = 0, derLt.fct = 0, X = 0, maxiter = 100,
step = 1, norm.diff.conv = 1e-05,
norm.score.conv = 1e-05, y.eps = 0,
iter.orig = 5, chscore.criterion = 2, m.initial = y)
```

Arguments

`y`	Vector of table counts.
`Z`	Population matrix describing stratification scheme.
`ZF`	Sampling constraint matrix.
`h.fct`	Constraint function. If h.fct is not equal to the constant 0 (the default), it must be a function of the single variable m, and it must return a column vector, (default: h.fct=0).
`derht.fct`	Function that computes analytic derivative of h.fct.
`L.fct`	Model link function. L.fct must be a function of the single variable m and must return a column vector, (default: L.fct=0).
`derLt.fct`	Function that computes analytic derivative of L.cft.
`X`	Design matrix for the link function L(m)=X*beta.
`maxiter`	Maximum number of iterations.
`step`	Step-size value.
`norm.diff.conv`	Convergence criteria value.

`norm.score.conv`	Convergence criteria value.
`y.eps`	Non-negative constant to be temporarily added to the original counts in y.
`iter.orig`	Iteration at which the original counts will be used, (default: iter.orig=5).
`chscore.criterion`	The maximum multiplicative change allowed for ratio norm.score.new/norm.score.old in the iterative updating.
`m.initial`	Initial estimate of m.

Details

This function computes ML estimates, standard errors, and fit statistics for contingency table models of the general form h(m) = 0 or L(m)=X*beta where m is the vector of expected counts and h (L) is any sufficiently smooth (continuous second derivatives) constraint function that satisfies non-restrictive homogeneity conditions. The ML estimates are based on the observed contingency table counts y, where y can be specified as coming from a wide variety of sampling models; in particular, y can be a realization of any MP (Multinomial-Poisson) random vector that is characterized by a population matrix Z and a sampling constraint matrix ZF. The matrix Z specifies the stratification sampling plan and ZF specifies which strata sample sizes are fixed. Note that ZF'y = n gives the vector of fixed sample sizes. (Non-fixed sample sizes are assumed to be realizations of Poisson random variables).

The population matrix Z comprises '0's and '1's and is of dimension cxK, where c is the number of counts in y and K is the number of strata or populations. Thus, the rows correspond to observation numbers and the columns correspond to the strata. A '1' in row i of column j implies that the ith count came from the jth stratum. Note that Z will have exactly one '1' in each row, and at least one '1' in each column. The population matrix Z which is a column vector of '1's, implies that all the counts came from the same, and only, stratum.

The sampling constraint matrix tells which of the strata sample sizes are fixed. Note: non-fixed sample sizes are assumed to be realizations of Poisson random variables. (default: ZF=Z, i.e. "product-multinomial" sampling). For non-zero ZF, the columns are a subset of the columns in population matrix Z. If the jth column of Z is included in ZF, then the jth stratum sample size is considered fixed, otherwise, if the jth column of Z is NOT included in ZF, the jth stratum sample size is taken to be a realization of a Poisson random variable. When ZF=0, all of the stratum sample sizes are taken to be realizations of Poisson random variables. The default, ZF=Z, means that all of the stratum sample sizes are fixed; this is the (product-)multinomial setting. Note that ZF'y = n is the vector of fixed sample sizes.

Value

Output can be displayed using mph.summary.

Note

The function mphineq.fit is more general because it allows also for inequality constraints d(m)>0. See the help of mphineq.fit for more details.

Author(s)

Joseph B. Lang, Dept. of Statistics and Actuarial Science Univ. of Iowa, Iowa City, IA 52242

References

Lang, J.B. (2004):ăMultinomial-Poisson Homogeneous Models for Contingency Tables, The Annals of Statistics, 32, 340-383.

Lang, J.B. (2005): Homogeneous Linear Predictor Models for Contingency Tables, JASA, 100, 121-134.

See Also

mphineq, mph.summary, create.U

Examples

```
y <-
c(56, 13, 7 ,
6, 4, 10,
1 ,5, 15)

Z <- ZF <- matrix(1,9,1)
Lprobit.fct <- function(m) {
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
rbind(
qnorm(pr[2]+pr[3]),
qnorm(pr[3]),
qnorm(pc[2]+pc[3]),
qnorm(pc[3]),
log(m)
)
}

Xprobit <-
c(1, 0, 0,
0 ,1, 0,
1, 0, 1,
0 ,1, 1)

Xprobit <- matrix(Xprobit,4,3,byrow=TRUE)
A <- factor(gl(3,3))
B <- factor(gl(3,1,9))
Xindep <- model.matrix(~A+B)
Xsat <- model.matrix(~A*B)

# FIT USING THE SATURATED ASSOCIATION MODEL

X <- block.fct(Xprobit,Xsat)
a <- mph.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X )

mph.summary(a)
# TEST WHETHER b(AGE) = 0 (i.e. Test Marginal Homogeneity) vs. b(AGE) > 0 using the Wald

a$beta[3]/a$covbeta[3,3]**0.5

1-pnorm(3.046738)

# USE THE INDEPENDENCE ASSOCIATION MODEL
```

```
X <- block.fct(Xprobit,Xindep)
b <- mph.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X )

mph.summary(b)
```

`mph.summary` *model summary*

Description

This function computes and prints a collection of summary statistics of the fitted model.

Usage

```
mph.summary(mph.out, cell.stats = FALSE, model.info = FALSE)
```

Arguments

`mph.out`	Output of hmmm.mlfit.
`cell.stats`	Logical variable indicating whether cell specific statistics are to be output, (default: cell.stats=FALSE).
`model.info`	Logical variable indicating whether model information is to be output, (default: model.info=FALSE).

Author(s)

Joseph B. Lang, Dept of Statistics and Actuarial Science Univ of Iowa, Iowa City, IA 52242 8/16/01

See Also

hmmm.mlfit, hmmm.model.summary

Examples

```
 #================================================
# MODEL of STOCHASTIC INDEPENDENCE in SUB-TABLES
#================================================

# TABLE 2 - section 7.4:
# two way contingency table --> GROWTH x COMPETITIVENESS,
# (114 companies belonging to the manufacturing industry)
# GROWTH --> 4 categories, percentage og growth:
#            L=low, ML=medium/low, MH=medium/high, H=high;
# COMPETITIVENESS --> 3 categories, level of competitiveness
#                     L=low, M=medium, H=high.

y <- c(
 4, 11, 14,  4,
17, 22, 13,  2,
 6, 10,  6,  5)
```

```
# NB
# variable GROWTH --> recursive approach,
# the non-minimal sets are: M1=(L,ML,MH,H); M2=(MH,H); M3=(L,ML);
# variable COMPETITIVENESS --> continuation approach.

# identity design matrix 15 x 15
# NB: no constraints on the marginal logits


# univariate marginals
# logit type:
# l --> local,  g --> global,
# c --> cont., rc --> reverse cont.,
# r --> recursive
# NB: in this case the univariate marginals can be of any type
# as they are not constrained.
marginal1<-list(marg=c(1),types=c("r","marg"))
marginal2<-list(marg=c(2),types=c("marg","c"))

# bivariate marginals
marginal12<-list(marg=c(1,2),types=c("r","c"))

marginals<-list(marginal1,marginal2,marginal12)

# "COCACONTR" specifications
# ===========================
# NB: 'cocacontr' is needed only for recursive 'r' or baseline 'b' logits.
#     It is a list of matrices of constrasts for 'r' and 'b' logits; it follows
#     the natural order of the variables. Arbitrary (for example 0) for
#     other logits.

#  For example: variable A with categories a1, a2, a3, a4;
#  the non-minimal sets M1, M2, M3 could be as follows:
#
#     _____|M1___
#     |         |
#   __|M3__   __|M2__
#   |     |   |     |
#   a1   a2   a3   a4
#
# then 'cocacontr' should be
#       cocacontr=list(
#                    matrix(c(1,1,0,0,
#                             0,0,1,0,
#                             1,0,0,0,
#                             0,0,1,1,
#                             0,0,0,1,
#                             0,1,0,0),6,4,byrow=TRUE)
#                                                     ,0),
#
# as the first 3 rows represent the 'LEFT children' of the node Mm, m=1,2,3
#               1 1 0 0  --> M1
#               0 0 1 0  --> M2
#               1 0 0 0  --> M3
#
# and the last 3 rows represent the 'RIGHT children' of the node Mm, m=1,2,3
#               0 0 1 1  --> M1
#               0 0 0 1  --> M2
#               0 1 0 0  --> M3

# NB: The marginal logits are builded as follows:
#     --> eta1 on the M1 node
#     --> eta2 on the M2 node
#     --> eta3 on the M3 node

# ==========================





# model with all the log-recursive o.r. =0 except for o.r.(1,1),
# that is "Model of stochastic independence in sub-tables"
# (except for the table referring to the first node of each hierarchy of sets)
XX<-diag(1,11)
XX<-XX[,-c(7,8,9,10,11)]

modelsor<-hmmm.model(marg=marginals,lev=c(4,3),
cocacontr=list(matrix(c
(1,1,0,0,
 0,0,1,0,
 1,0,0,0,
 0,0,1,1,
 0,0,0,1,
 0,1,0,0
)
,6,4,byrow=TRUE),0
),
strata=1,X=XX)

am<-hmmm.mlfit(y,modelsor)

# H0=(am model) vs H1=(amsat model)
mph.summary(am)
```

`mphineq.fit` *mph and hmmm models fitting under inequality constraints*

Description

Function to maximize the loglikelihood of a multinomial poisson model under nonlinear equality and inequality constraints

Usage

```
mphineq.fit(y, Z, ZF = Z, h.fct = 0, derht.fct = 0, d.fct = 0,
 derdt.fct = 0, L.fct = 0, derLt.fct = 0, X = 0, maxiter = 100,
 step = 1, norm.diff.conv = 1e-05, norm.score.conv = 1e-05,
 y.eps = 0, iter.orig = 5, chscore.criterion = 2, m.initial = y,
mup = 1)
```

Arguments

`y`	Vectorized sample contingency table.
`Z`	Population matrix. The population matrix Z comprises '0's and '1's and is of dimension cxK, where c is the number of counts in y and K is the number of strata or populations. Thus, the rows correspond to observation numbers and the columns correspond to the strata. A '1' in row i of column j implies that the ith count cames from the jth stratum. Note that Z will have exactly one '1' in each row, and at least one '1' in each column. The population matrix Z = matrix(1,length(y),1), which is a column vector of '1's, implies that all the counts came from the same, and only, stratum.
`ZF`	Sample constraints matrix. For non-zero ZF, the columns are a subset of the columns in population matrix Z. If the jth column of Z is included in ZF, then the jth stratum sample size is considered fixed, otherwise, if the jth column of Z is NOT included in ZF, the jth stratum sample size is taken to be a realization of a Poisson random variable. When ZF=0, all of the stratum sample sizes are taken to be realizations of Poisson random variables. The default, ZF=Z, means that all of the stratum sample sizes are fixed; this is the (product-)multinomial setting. Note that ZF'y = n is the vector of fixed sample sizes.
`h.fct`	Function h(m) of equality constraints. This function of m must return a vector.
`derht.fct`	Derivative of h(m) if not supplied numerical derivative are used.
`d.fct`	Function for inequality constraints d(m)>0. This function of m must return a vector.
`derdt.fct`	Derivative of d(m) if not supplied numerical derivative are used.
`L.fct`	Link function for the linear model L(m)=X*beta.
`derLt.fct`	Derivative of L(m) if not supplied numerical derivative are used.
`X`	Model matrix for L(m)=X*beta.
`maxiter`	Maximum number of iterations.
`step`	Interval lenght for the linear search.
`norm.diff.conv`	Convergence crirerium for parameters.
`norm.score.conv`	Convergence criterium for constraints.
`y.eps`	Non-negative constant to be temporarily added to the original counts in y, (default: y.eps=0).
`iter.orig`	Not used.
`chscore.criterion`	If zero convergence information are printed at every iteration.
`m.initial`	Initial estimate of m.
`mup`	Weight for the constraint part of the merit function.

Details

Adapted from mph.fit of Joseph B. Lang, Dept of Statistics and Actuarial Science Univ of Iowa, Iowa City, IA 52242 8/16/01, in order to include inequality constraints.

In particular the AS algorithm has been replaced by a sequential quadratic algorithm which is equivalent to AS when inequalities are not present.

The R functions quadprog and optimize have been used to implement the sequential quadratic algorithm.

More precisely the AS updating formulas are replaced by an equality-inequality constrained quadratic programming problem. The mph.fit step halving linear search is replaced by an optimal step lenght search performed by optimize.

Value

Use mph.summary to display the output.

Author(s)

Roberto Colombi, Colombi@unibg.it.

References

Lang J.B. (2004):ăMultinomial-Poisson Homogeneous Models for Contingency Tables, The Annals of Statistics, 32, 340-383.

Lang J.B. (2005): Homogeneous Linear Predictor Models for Contingency Tables, JASA, 100, 121-134.

See Also

hmmm.mlfit, create.XMAT, hmm.model.summary, mph.summary

Examples

```
y <-
c(56, 13, 7,
6, 4, 10,
1, 5, 15)

Z <- ZF <- matrix(1,9,1)
Lprobit.fct <- function(m) {
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
rbind(
qnorm(pr[2]+pr[3]),
qnorm(pr[3]),
qnorm(pc[2]+pc[3]),
qnorm(pc[3]),
log(m)
)
}
dprobit.fct <- function(m) {
p <- matrix(m,3,3,byrow=TRUE)/sum(m)
pr <- apply(p,1,sum)
pc <- apply(p,2,sum)
rbind(
-qnorm(pc[2]+pc[3])+qnorm(pr[2]+pr[3]),
-qnorm(pc[3])+qnorm(pr[3])
)
}

Xprobit <-
c(1, 0, 0,
0, 1, 0,
1, 0, 1,
0, 1, 1)

Xprobit <- matrix(Xprobit,4,3,byrow=TRUE)
A <- factor(gl(3,3))
B <- factor(gl(3,1,9))
Xindep <- model.matrix(~A+B)
Xsat <- model.matrix(~A*B)

# FIT USING THE SATURATED ASSOCIATION MODEL

X <- block.fct(Xprobit,Xsat)
a <- mphineq.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X,d.fct=dprobit.fct,maxiter=1000 )

mph.summary(a)
# TEST WHETHER b(AGE) = 0 (i.e. Test Marginal Homogeneity) vs. b(AGE) > 0 using the Wald

a$beta[3]/a$covbeta[3,3]**0.5

1-pnorm(3.046738)

# USE THE INDEPENDENCE ASSOCIATION MODEL

X <- block.fct(Xprobit,Xindep)
b <- mphineq.fit(y,Z,ZF,L.fct=Lprobit.fct,X=X,d.fct=dprobit.fct,maxiter=1000 )

mph.summary(b)
```

`pop` *population matrix Z*

Description

This function creates a population (Z) matrix corresponding to data entered by strata. It is assumed that all the strata have the same number of response levels.

Usage

```
pop(npop, nlev)
```

Arguments

`npop`	Number of populations or strata.
`nlev`	Number of response levels per stratum.

Value

Population matrix of dimension (npop*nlev X npop).

Author(s)

Joseph B. Lang

`rmult` *function to create a matrix of multinomial realizations.*

Description

A matrix of multinomial realizations is simulated. There will be N columns, each column contains a realization of a multinomial(n,p) random vector.

Usage

```
rmult(N = 1, n, p)
```

Arguments

`N`	Replication number.
`n`	Integer parameter of the multinomial.
`p`	Multinomial probabilities.

Value

A matrix with N columns, each column contains a realization of a multinomial(n,p) random vector.

Author(s)

Joseph B. Lang Dept of Stat and Act Sci Univ of Iowa

Index