Package ‘gnm’

April 22, 2015

Title  Generalized Nonlinear Models
Version  1.0-8
Date  2015-04-21
Description  Functions to specify and fit generalized nonlinear models, including models with multiplicative interaction terms such as the UNIDIFF model from sociology and the AMMI model from crop science, and many others. Over-parameterized representations of models are used throughout; functions are provided for inference on estimable parameter combinations, as well as standard methods for diagnostics etc.

URL  http://go.warwick.ac.uk/gnm
License  GPL-2 | GPL-3
Depends  R (>= 2.3.0)
Imports  MASS, stats, graphics, Matrix, nnet, qvcalc (>= 0.8-3), relimp
Suggests  vcdExtra
LazyData  yes
NeedsCompilation  yes
Author  Heather Turner [aut, cre],
        David Firth [aut],
        Brian Ripley [ctb],
        Bill Venables [ctb],
        Douglas M. Bates [ctb],
        Martin Maechler [ctb]
Maintainer  Heather Turner <ht@heatherturner.net>
Repository  CRAN
Date/Publication  2015-04-22 12:08:50

R topics documented:

  gnm-package .................................................. 3
  anova.gnm .................................................... 4
  asGnm ........................................................ 5
  backPain ....................................................... 6
barley ........................................... 8
barleyHeights ................................... 10
cautres ........................................... 11
checkEstimable .................................. 12
confint.gnm ...................................... 13
Const ............................................. 15
Diag ............................................. 16
Dref ............................................. 17
erikson .......................................... 22
exitInfo ......................................... 23
Exp ............................................... 24
expandCategorical ............................... 25
friend ........................................... 27
gnminfo ......................................... 28
House2001 ...................................... 30
instances ....................................... 36
Inv ............................................... 39
InvResiduals .................................... 40
mentalHealth ................................... 43
model.matrix.gnm ................................. 44
MPinv ........................................... 45
MultHomog ...................................... 46
Multiplicative interaction ....................... 48
nonlin.function ................................ 50
ofInterest .................................... 53
parameters ..................................... 55
pickCoef ....................................... 56
plot.gnm ....................................... 57
predict.gnm .................................... 57
profile.gnm ..................................... 60
residSVD ....................................... 62
se ............................................... 65
summary.gnm ................................... 66
Symm ......................................... 68
termPredictors .................................. 70
Topo ............................................ 71
vcov.gnm ....................................... 72
voting .......................................... 74
wedderburn ..................................... 76
wheat ........................................... 78
yaish .......................................... 81

Index ........................................... 83
Description

Functions to specify, fit and evaluate generalized nonlinear models.

Details

gnm provides functions to fit generalized nonlinear models by maximum likelihood. Such models extend the class of generalized linear models by allowing nonlinear terms in the predictor.

Some special cases are models with multiplicative interaction terms, such as the UNIDIFF and row-column association models from sociology and the AMMI and GAMMI models from crop science; stereotype models for ordered categorical response, and diagonal reference models for dependence on a square two-way classification.

gnm is a major re-working of an earlier Xlisp-Stat package, "Llama". Over-parameterized representations of models are used throughout; functions are provided for inference on estimable parameter combinations, as well as standard methods for diagnostics etc.

The following documentation provides further information on the gnm package:

\texttt{gnmOverview} vignette("gnmOverview", package = "gnm")
\texttt{NEWS} file.show(system.file("NEWS", package = "gnm"))

Author(s)

Heather Turner and David Firth

Maintainer: Heather Turner <ht@heatherturner.net>

References

http://www.warwick.ac.uk/go/gnm

See Also

\texttt{gnm} for the model fitting function, with links to associated functions.

Examples

demo(gnm)
Analysis of Deviance for Generalized Nonlinear Models

Description

Compute an analysis of deviance table for one or more generalized nonlinear models

Usage

```r
## S3 method for class 'gnm'
anova(object, ..., dispersion = NULL, test = NULL)
```

Arguments

- `object` an object of class gnm
- `...` additional objects of class gnm or glm
- `dispersion` the dispersion parameter for the fitting family. By default it is derived from object
- `test` (optional) a character string, (partially) matching one of "Chisq", "F", or "Cp". See `stat.anova`.

Details

Specifying a single object gives a sequential analysis of deviance table for that fit. The rows of the table show the reduction in the residual deviance and the current residual deviance as each term in the formula is added in turn.

If more than one object is specified, the rows of the table show the residual deviance of the current model and the change in the residual deviance from the previous model. (This only makes statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

If `test` is specified, the table will include test statistics and/or p values for the reduction in deviance. For models with known dispersion (e.g., binomial and Poisson fits) the chi-squared test is most appropriate, and for those with dispersion estimated by moments (e.g., `gaussian`, `quasibinomial` and `quasipoisson` fits) the F test is most appropriate. Mallows’ Cp statistic is the residual deviance plus twice the estimate of $\sigma^2$ times the residual degrees of freedom, which is closely related to AIC (and a multiple of it if the dispersion is known).

Value

An object of class "anova" inheriting from class "data.frame".

Warning

The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values and R’s default of `na.action = na.omit` is used; an error will be given in this case.
asGnm

Author(s)
Modification of anova.glm by the R Core Team. Adapted for "gnm" objects by Heather Turner.

See Also
gnm, anova

Examples
set.seed(1)

## Fit a uniform association model separating diagonal effects
Rscore <- scale(as.numeric(row(occupationalStatus)), scale = FALSE)
Cscore <- scale(as.numeric(col(occupationalStatus)), scale = FALSE)
Uniform <- glm(Freq ~ origin + destination + Diag(origin, destination) +
               Rscore:Cscore, family = poisson, data = occupationalStatus)

## Fit an association model with homogeneous row-column effects
RChomog <- gnm(Freq ~ origin + destination + Diag(origin, destination) +
               MultHomog(origin, destination), family = poisson,
               data = occupationalStatus)

## Fit an association model with separate row and column effects
RC <- gnm(Freq ~ origin + destination + Diag(origin, destination) +
          Mult(origin, destination), family = poisson,
          data = occupationalStatus)

anova(RC, test = "Chisq")

anova(Uniform, RChomog, RC, test = "Chisq")

---

asGnm

Coerce Linear Model to gnm Object

Description
asGnm is a generic function which coerces objects of class "glm" or "lm" to an object of class "gnm".

Usage
asGnm(object, ...)

Arguments

object an object of class "glm" or "lm".

... additional arguments for method functions.
Details
Components are added to or removed from object to produce an object of class "gnm". This can be useful in model building, see examples.

Value
An object of class "gnm" - see gnm for full description.

Author(s)
Heather Turner

References

See Also
gnm, glm, lm

Examples
set.seed(1)

## Scale yields to reproduce analyses reported in Vargas et al (2001)
yield.scaled <- wheat$yield * sqrt(3/1000)
treatment <- interaction(wheat$tillage, wheat$summerCrop, wheat$manure, wheat$N, sep = "")
## Fit linear model
mainEffects <- lm(yield.scaled ~ year + treatment, data = wheat)

## Convert to gnm object to allow addition of Mult() term
svdStart <- residSVD(mainEffects, year, treatment, 3)
bilinear1 <- update(asGnm(mainEffects), . ~ . +
    Mult(year, treatment),
    start = c(coef(mainEffects), svdStart[,1]))

backPain

Data on Back Pain Prognosis, from Anderson (1984)

Description
Data from a study of patients suffering from back pain. Prognostic variables were recorded at presentation and progress was categorised three weeks after treatment.

Usage
backPain
backPain

Format

A data frame with 101 observations on the following 4 variables.

- **x1** length of previous attack.
- **x2** pain change.
- **x3** lordosis.
- **pain** an ordered factor describing the progress of each patient with levels worse < same < slight.improvement < moderate.improvement < marked.improvement < complete.relief.

Source

http://ideas.repec.org/c/boc/bocode/s419001.html

References


Examples

```r
set.seed(1)
summary(backPain)

### Re-express as count data
backPainLong <- expandCategorical(backPain, "pain")

### Fit models described in Table 5 of Anderson (1984)

### Logistic family models
noRelationship <- gnm(count ~ pain, eliminate = id,
                      family = "poisson", data = backPainLong)

### stereotype model
oneDimensional <- update(noRelationship,
                          ~ . + Mult(pain, x1 + x2 + x3))

### multinomial logistic
doDimensional <- update(noRelationship, ~ . + pain:(x1 + x2 + x3))

### Models to determine distinguishability in stereotype model
# constrain scale of category-specific multipliers
oneDimensional <- update(noRelationship,
                          ~ . + Mult(pain, offset(x1) + x2 + x3))

### obtain identifiable contrasts & id possibly indistinguishable slopes
generalContrasts(oneDimensional, pickCoef(oneDimensional, "[.]pain"))

### Not run:
# (this part not needed for package testing)
# fit simpler models and compare
.pain <- backPainLong$pain
```
levels(.pain)[2:3] <- paste(levels(.pain)[2:3], collapse = " | ")
fiveGroups <- update(noRelationship, 
  " . + Mult(.pain, x1 + x2 + x3)"
)
levels(.pain)[4:5] <- paste(levels(.pain)[4:5], collapse = " | ")
fourGroups <- update(fiveGroups)
levels(.pain)[2:3] <- paste(levels(.pain)[2:3], collapse = " | ")
threeGroups <- update(fourGroups)

### Grouped continuous model, aka proportional odds model
library(MASS)
sixCategories <- polr(pain ~ x1 + x2 + x3, data = backPain)

### Obtain number of parameters and log-likelihoods for equivalent multinomial models as presented in Anderson (1984)
logLikMultinom <- function(model, size){
  object <- get(model)
  if (inherits(object, "gmm")) {
    l <- sum(object$y * log(object$fitted/size))
    c(nParameters = object$rank - nlevels(object$eliminate),
      logLikelihood = l)
  } else
    c(nParameters = object$edf, logLikelihood = -deviance(object)/2)
}
size <- tapply(backPainLong$count, backPainLong$id, sum)[backPainLong$id]
models <- c("threeDimensional", "oneDimensional", "noRelationship",
  "fiveGroups", "fourGroups", "threeGroups", "sixCategories")
t(sapply(models, logLikMultinom, size))

### End(Not run)

barley

**Jenkyn’s Data on Leaf-blotch on Barley**

**Description**

Incidence of *R. secalis* on the leaves of ten varieties of barley grown at nine sites.

**Usage**

barley

**Format**

A data frame with 90 observations on the following 3 variables.

- **y** the proportion of leaf affected (values in [0,1])
- **site** a factor with 9 levels A to I
- **variety** a factor with 10 levels c(1:9, "X")
Note

This dataset was used in Wedderburn’s original paper (1974) on quasi-likelihood.

Source


References


Wedderburn, R W M (1974). Quasilikelihood functions, generalized linear models and the Gauss-

Examples

c> set.seed(1)

c### Fit Wedderburn’s logit model with variance proportional to [mu(1-mu)]^2

clogitModel <- glm(y ~ site + variety, family = wedderburn, data = barley)
cfit <- fitted(logitModel)
cprint(sum((barleyy - fit)^2 / (fit * (1-fit))^2))
c### Agrees with the chi-squared value reported in McCullagh and Nelder

c### (1989, p331), which differs slightly from Wedderburn’s reported value.

c### Fit the biplot model as in Gabriel (1998, p694)
cbiplotModel <- gnm(y ~ -1 + instances(Mult(site, variety), 2),
cfamily = wedderburn, data = barley)
cbarleySVD <- svd(matrix(biplotModel$predictors, 10, 9))
cA <- sweep(barleySVD$v, 2, sqrt(barleySVD$d), "*"[, 1:2]
cB <- sweep(barleySVD$u, 2, sqrt(barleySVD$d), "*"[, 1:2]
c### These are essentially A and B as in Gabriel (1998, p694), from which

c### the biplot is made by

cplot(rbind(A, B), pch = c(levels(barley$site), levels(barley$variety)))

c### Fit the double-additive model as in Gabriel (1998, p697)
cvariety.binary <- factor(match(barley$variety, c(2,3,6), nomatch = 0) > 0,
clabels = c("rest", "2,3,6"))
cdoubleAdditive <- gnm(y ~ variety + Mult(site, variety.binary),
cfamily = wedderburn, data = barley)
c### It is unclear why Gabriel’s chi-squared statistics differ slightly

c### from the ones produced in these fits. Possibly Gabriel adjusted the

c### data somehow prior to fitting?
barleyHeights

Heights of Barley Plants

Description

Average heights for 15 genotypes of barley recorded over 9 years.

Usage

barleyHeights

Format

A data frame with 135 observations on the following 3 variables.

- height  average height over 4 replicates (cm)
- year   a factor with 9 levels 1974 to 1982
- genotype  a factor with 15 levels 1:15

Source


References


Examples

```r
set.seed(1)
## Fit AMMI-1 model
barleyModel <- gnm(height ~ year + genotype + Mult(year, genotype),
                   data = barleyHeights)

## Get row and column scores with se's
gamma <- getContrasts(barleyModel, pickCoef(barleyModel, "[.y"),
                      ref = "mean", scaleWeights = "unit")
delta <- getContrasts(barleyModel, pickCoef(barleyModel, "[.g"),
                      ref = "mean", scaleWeights = "unit")

## Corresponding CI's similar to Chadoeuf & Denis (1991) Table 8
## (allowing for change in sign)
gamma[,1] + (gamma[,2]) %*% c(-1.96, 1.96)
delta[,1] + (delta[,2]) %*% c(-1.96, 1.96)

## Multiplier of row and column scores
height <- matrix(scale(barleyHeights$height, scale = FALSE), 15, 9)
```
## Data on Class, Religion and Vote in France

### Description
A 4-way contingency table of vote by class by religion in four French elections

### Usage
cautres

### Format
A table of counts, with classifying factors vote (levels 1:2), class (levels 1:6) and religion (levels 1:4) and election (levels 1:4).

### Source
Bruno Cautres

### References

### Examples
```r
set.seed(1)

# Fit a "double UNIDIFF" model with the religion-vote and class-vote interactions both modulated by nonnegative election-specific multipliers
doubleUnidiff <- gnm(Freq ~ election*vote + election*class*religion +
    Mult(Exp(election), religion:vote) +
    Mult(Exp(election), class:vote),
    family = poisson, data = cautres)

# Deviance should be 133.04

# Examine the multipliers of the class-vote log odds ratios
ofInterest(doubleUnidiff) <- pickCoef(doubleUnidiff, "class:vote[]")
coef(doubleUnidiff)

# Coefficients of interest:
# Mult(Exp(.), class:vote).election1
# -0.38357138
# Mult(Exp(.), class:vote).election2
# 0.29816599
# Mult(Exp(.), class:vote).election3
# 0.06580307
```
checkEstimable

Check Whether One or More Parameter Combinations in a gnm Model are Identified

Description

For each of a specified set of linear combinations of parameters from a gnm model, checks numerically whether the combination’s estimate is invariant to re-parameterization of the model.

Usage

checkEstimable(model, combMatrix = diag(length(coef(model))),
    tolerance = NULL)

Arguments

model a model object of class “gnm”
combMatrix numeric: either a vector of length the same as length(coef(model)), or a matrix with that number of rows. Coefficients of one or more linear combinations of the model’s parameters.
tolerance numeric: a threshold value for detection of non-estimability. If NULL, the default value of the tol argument to rankMatrix is used.
Value

A logical vector of length equal to the number of parameter combinations tested; NA where a parameter combination is identically zero.

Author(s)

David Firth and Heather Turner

References


See Also

gnm, se, getContrasts

Examples

```r
set.seed(1)

## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
               Mult(Exp(educ), orig:dest), family = poisson,
               data = yaish, subset = (dest != 7))

## Check whether multiplier contrast educ4 - educ5 is estimable
ofInterest(unidiff) <- pickCoef(unidiff, "[.]educ")
mycontrast <- numeric(length(coef(unidiff)))
mycontrast[ofInterest(unidiff)[4:5]] <- c(1, -1)
checkEstimable(unidiff, mycontrast)
## should be TRUE

## Check whether multiplier educ4 itself is estimable
mycontrast[ofInterest(unidiff)[5]] <- 0
checkEstimable(unidiff, mycontrast)
## should be FALSE -- only *differences* are identified here
```

---

**confint.gnm**

*Compute Confidence Intervals of Parameters in a Generalized Nonlinear Model*

**Description**

Computes confidence intervals for one or more parameters in a generalized nonlinear model, based on the profiled deviance.
Usage

## S3 method for class 'gnm'
confint(object, parm = ofInterest(object), level = 0.95,
         trace = FALSE, ...)

## S3 method for class 'profile.gnm'
confint(object, parm = names(object), level = 0.95, ...)

Arguments

object
  an object of class "gnm" or "profile.gnm"

parm
  (optional) either a numeric vector of indices or a character vector of names,
  specifying the parameters for which confidence intervals are to be estimated. If
  parm is missing, confidence intervals are found for all parameters.

level
  the confidence level required.

trace
  a logical value indicating whether profiling should be traced.

... arguments passed to or from other methods

Details

These are methods for the generic function confint in the base package.

For "gnm" objects, profile.gnm is first called to profile the deviance over each parameter specified
by parm, or over all parameters in the model if parm is missing.

The method for "profile.gnm" objects is then called, which interpolates the deviance profiles to
estimate the limits of the confidence interval for each parameter, see profile.gnm for more details.

If a "profile.gnm" object is passed directly to confint, parameters specified by parm must be a
subset of the profiled parameters.

For unidentified parameters a confidence interval cannot be calculated and the limits will be returned
as NA. If the deviance curve has an asymptote and a limit of the confidence interval cannot be
reached, the limit will be returned as -Inf or Inf appropriately. If the range of the profile does not
extend far enough to estimate a limit of the confidence interval, the limit will be returned as NA. In
such cases, it may be desirable create a profile object directly, see profile.gnm for more details.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.
These will be labelled as (1-level)/2 and 1 - (1-level)/2 in % (by default 2.5% and 97.5%).

Author(s)

Modification of confint.glm by W. N. Venables and B. D. Ripley. Adapted for "gnm" objects by
Heather Turner.

See Also

profile.gnm, gnm, profile.glm
Examples

```r
### Example in which profiling doesn't take too long
count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting/total - count)
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
  constrain = "delta1", family = binomial,
  data = voting)
# profile diagonal effects
confint(classMobility, parm = 3:7, trace = TRUE)

### Not run:
### Profiling takes much longer here, but example more interesting!
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
  Mult(Exp(educ), orig:dest),
  ofInterest = "[.]educ", constrain = "[.]educ1",
  family = poisson, data = yaish, subset = (dest != 7))

### Letting 'confint' compute profile
confint(unidiff, trace = TRUE)
## 2.5 %  97.5 %
## Mult(Exp(.), orig:dest).educ1 NA    NA
## Mult(Exp(.), orig:dest).educ2 -0.5978901 0.1022447
## Mult(Exp(.), orig:dest).educ3 -1.4836854 -0.2362378
## Mult(Exp(.), orig:dest).educ4 -2.5792398 -0.2953420
## Mult(Exp(.), orig:dest).educ5   -Inf  -0.7007616

### Creating profile object first with user-specified stepsize
prof <- profile(unidiff, trace = TRUE, stepsize = 0.1)
confint(prof, ofInterest(unidiff)[2:5])
## 2.5 %  97.5 %
## Mult(Exp(.), orig:dest).educ2 -0.5978324 0.1022441
## Mult(Exp(.), orig:dest).educ3 -1.4834753 -0.2362138
## Mult(Exp(.), orig:dest).educ4  -NA -0.2950790
## Mult(Exp(.), orig:dest).educ5  -NA    NA

### For 95% confidence interval, need to estimate parameters for which
### z = +/- 1.96. Profile has not gone far enough for last two parameters
range(prof[4][2])
### -1.566681  2.408650
range(prof[5][2])
### -0.5751376   1.1989487

### End(Not run)
```

### Const

**Specify a Constant in a "nonlin" Function Predictor**

**Description**

A symbolic wrapper to specify a constant in the predictor of a "nonlin" function.
Usage

Const(const)

Arguments

const a numeric value.

Value

A call to rep used to create a variable representing the constant in the model frame.

Note

Const may only be used in the predictor of a "nonlin" function. Use offset to specify a constant in the model formula.

Author(s)

Heather Turner

See Also

gnm, formula, offset

Examples

## One way to fit the logistic function without conditional linearity as in ?nls
library(gnm)
set.seed(1)
DNase1 <- subset(DNase, Run == 1)

test <- gnm(density ~ 1 + Mult(1, Inv(Const(1)) + Exp(Mult(1 + offset(-log(conc)), Inv(1)))),
start = c(NA, 0, 1), data = DNase1, trace = TRUE)
coef(test)

---

### Diag

Equality of Two or More Factors

Description

Converts two or more factors into a new factor whose value is 0 where the original factors are not all equal, and nonzero otherwise.

Usage

Diag(..., binary = FALSE)
Arguments

... One or more factors
binary Logical

Details

Used mainly in regression models for data classified by two or more factors with the same levels. By default, operates on k-level factors to produce a new factor having k+1 levels; if binary = TRUE is specified, the result is a coarser binary variable equal to 1 where all of the input factors are equal and 0 otherwise.

Value

Either a factor (if binary = FALSE) or a 0-1 numeric vector (if binary = TRUE).

Author(s)

David Firth and Heather Turner

See Also

Symm

Examples

row <- gl(4, 4, 16)
col <- gl(4, 1, 16)
diag4by4 <- Diag(row, col)
matrix(Diag(row, col, binary = TRUE), 4, 4)
Details

`Dref` specifies diagonal reference terms as introduced by Sobel (1981, 1985). Such terms comprise an additive component for each factor of the form

\[ w_f \gamma_l \]

where \( w_f \) is the weight for factor \( f \), \( \gamma_l \) is the diagonal effect for level \( l \) and \( l \) is the level of factor \( f \) for the given data point.

The weights are constrained to be nonnegative and to sum to one as follows

\[ w_f = \frac{e^{\delta_f}}{\sum_i e^{\delta_i}} \]

and the \( \delta_f \) are modelled as specified by the `delta` argument (constant weights by default). The returned parameters are those in the model for \( \delta_f \), rather than the implied weights \( w_f \). The `DrefWeights` function will take a fitted `gnm` model and return the weights \( w_f \), along with their standard errors.

If the factors passed to `Dref` do not have exactly the same levels, the set of levels in the diagonal reference term is taken to be the union of the factor levels, sorted into increasing order.

Value

A list with the anticipated components of a "nonlin" function:

- **predictors** the factors passed to `Dref` and the formulae for the weights.
- **common** an index to specify that common effects are to be estimated across the factors.
- **term** a function to create a deparsed mathematical expression of the term, given labels for the predictors.
- **start** a function to generate starting values for the parameters.
- **call** the call to use as a prefix for parameter labels.

Author(s)

Heather Turner

References


**See Also**

gnm, formula, nonlin.function

**Examples**

```r
## Examples from Clifford and Heath paper
## (Results differ slightly - possible transcription error in
## published data?)
set.seed(1)

## reconstruct counts voting Labour/non-Labour
count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting$total - count)

## fit diagonal reference model with constant weights
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                      family = binomial, data = voting)
DrefWeights(classMobility)

## create factors indicating movement in and out of salariat (class 1)
upward <- with(voting, origin != 1 & destination == 1)
downward <- with(voting, origin == 1 & destination != 1)

## fit separate weights for the "socially mobile" groups
socialMobility <- gnm(yvar ~ -1 + Dref(origin, destination,
                      delta = ~ 1 + downward + upward),
                      family = binomial, data = voting)
DrefWeights(socialMobility)

## fit separate weights for downwardly mobile groups only
downwardMobility <- gnm(yvar ~ -1 + Dref(origin, destination,
                      delta = ~ 1 + downward),
                      family = binomial, data = voting)
DrefWeights(downwardMobility)

## Not run:
## Examples from Van der Slik paper
## For illustration only - data not publically available
## Using data in data.frame named 'conformity', with variables
## MCFM - mother's conformity score
## FCFF - father's conformity score
## MOPLM - a factor describing the mother's education with 7 levels
## FOPLF - a factor describing the father's education with 7 levels
## AGEM - mother's birth cohort
## MRMHM - mother's traditional role model
## FRMF - father's traditional role model
## MWORK - mother's employment
## MFRCM - mother's family conflict score
## FFCC - father's family conflict score

set.seed(1)
```
## Models for mothers' conformity score as specified in Figure 1

A <- gnm(MCFM ~ -1 + AGEM + MRMM + FRMF + MWORK + MFCM +
          Dref(MOPLM, FOPLF), family = gaussian, data = conformity,
          verbose = FALSE)

A

## Call:
## gnm(formula = MCFM ~ -1 + AGEM + MRMM + FRMF + MWORK + MFCM +
##      Dref(MOPLM, FOPLF), family = gaussian, data = conformity,
##      verbose = FALSE)

## Coefficients:
##                  AGEM         MRMM
## agem             0.06363    -0.32425
## frrm             -0.25324    -0.06430
## mwork            -0.06043    -0.33731
## mfcm Dref(MOPLM, FOPLF)delta
## dref             -0.02505     4.95121
## Dref(MOPLM, FOPLF)delta2 Dref(., .).MOPLM|FOPLF1
## dref            -4.86329     4.86458
## Dref(., .).MOPLM|FOPLF2 Dref(., .).MOPLM|FOPLF3
## dref            4.72343     4.43516
## Dref(., .).MOPLM|FOPLF4 Dref(., .).MOPLM|FOPLF5
## dref            4.18873     4.43378

## Deviance:     425.3389
## Pearson chi-squared: 425.3389
## Residual df:  576

## Weights as in Table 4

DrefWeights(A)

## Refitting with parameters of first Dref weight constrained to zero
## $MOPLM
## weight  se
## 0.4225636 0.1439829
##
## $FOPLF
## weight  se
## 0.5774364 0.1439829

F <- gnm(MCFM ~ -1 + AGEM + MRMM + FRMF + MWORK + MFCM +
          Dref(MOPLM, FOPLF, delta = -1 + MFCM), family = gaussian,
          data = conformity, verbose = FALSE)

F

## Call:
## gnm(formula = MCFM ~ -1 + AGEM + MRMM + FRMF + MWORK + MFCM +
##      Dref(MOPLM, FOPLF, delta = -1 + MFCM), family = gaussian,
##      data = conformity, verbose = FALSE)

## Coefficients:
##                  AGEM
## Standard error for MFCM == 1 lower than reported by Van der Slik et al

DrefWeights(F)

# Refitting with parameters of first Dref weight constrained to zero

### MOPLM

<table>
<thead>
<tr>
<th>MFCM</th>
<th>weight</th>
<th>se</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.02974675</td>
<td>0.2277711</td>
</tr>
<tr>
<td>2</td>
<td>0.74465224</td>
<td>0.2006916</td>
</tr>
</tbody>
</table>

### FOPLF

<table>
<thead>
<tr>
<th>MFCM</th>
<th>weight</th>
<th>se</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9702532</td>
<td>0.2277711</td>
</tr>
<tr>
<td>2</td>
<td>0.2553478</td>
<td>0.2006916</td>
</tr>
</tbody>
</table>

## End(Not run)
Intergenerational Class Mobility in England/Wales, France and Sweden

Description

Intergenerational class mobility among the male populations of England and Wales; France, and Sweden.

Usage

erikson

Format

A table of counts, with classifying factors origin (father’s class; levels I, II, III, IVa, IVb, IVc, V/VI, VIIa, VIIb) destination (son’s class; levels as before), and country (son’s country of residence; levels EW, F, S).

Source


References


Examples

```r
set.seed(1)

### Collapse to 7 by 7 table as in Erikson (1982)

erikson <- as.data.frame(erikson)
lvl <- levels(erikson$origin)
levels(erikson$origin) <- levels(erikson$destination) <-
c(rep(paste(lvl[1:2], collapse = " + "), 2), lvl[3],
 rep(paste(lvl[4:5], collapse = " + "), 2), lvl[6:9])
erikson <- xtabs(Freq ~ origin + destination + country, data = erikson)

### Fit the models given in first half of Table 3 of Xie (1992)

## Null association between origin and destination
nullModel <- gnm(Freq ~ country*origin + country*destination,
 family = poisson, data = erikson)
```
## exitInfo

### Description

A utility function to print information on final iteration in gnm fit, intended for use when gnm has not converged.

### Usage

```r
exitInfo(object)
```

### Arguments

- **object**: a gnm object.
Exp

Specify the Exponential of a Predictor in a gnm Model Formula

Description

A function of class "nonlin" to specify the exponential of a predictor in the formula argument to gnm.

Usage

Exp(expression, inst = NULL)
**Arguments**

- **expression**: a symbolic expression representing the (possibly nonlinear) predictor.
- **inst**: (optional) an integer specifying the instance number of the term.

**Details**

The expression argument is interpreted as the right hand side of a formula in an object of class "formula", except that an intercept term is not added by default. Any function of class "nonlin" may be used in addition to the usual operators and functions.

**Value**

A list with the components required of a "nonlin" function:

- **predictors**: the expression argument passed to Exp
- **term**: a function to create a deparsed mathematical expression of the term, given a label for the predictor.
- **call**: the call to use as a prefix for parameter labels.

**Author(s)**

Heather Turner and David Firth

**See Also**

`gnm, formula, nonlin.function`

**Examples**

```r
set.seed(1)

## Using 'Mult' with 'Exp' to constrain the first constituent multiplier
## to be non-negative

## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
               Mult(Exp(educ), orig:dest),
               family = poisson, data = yaish, subset = (dest != 7))
```

**Description**

Expands the rows of a data frame by re-expressing observations of a categorical variable specified by `catvar`, such that the column(s) corresponding to `catvar` are replaced by a factor specifying the possible categories for each observation and a vector of 0/1 counts over these categories.
expandCategorical

Usage

expandCategorical(data, catvar, sep = ".", countvar = "count",
                   idvar = "id", as.ordered = FALSE, group = TRUE)

Arguments

data         a data frame.
catvar       a character vector specifying factors in data whose interaction will form the basis of the expansion.
sep          a character string used to separate the concatenated values of catvar in the name of the new interaction factor.
countvar     (optional) a character string to be used for the name of the new count variable.
idvar        (optional) a character string to be used for the name of the new factor identifying the original rows (cases).
as.ordered    logical - whether the new interaction factor should be of class "ordered".
group        logical: whether or not to group individuals with common values over all covariates.

Details

Each row of the data frame is replicated \( c \) times, where \( c \) is the number of levels of the interaction of the factors specified by catvar. In the expanded data frame, the columns specified by catvar are replaced by a factor specifying the \( r \) possible categories for each case, named by the concatenated values of catvar separated by sep. The ordering of factor levels will be preserved in the creation of the new factor, but this factor will not be of class "ordered" unless the argument as.ordered = TRUE. A variable with name countvar is added to the data frame which is equal to 1 for the observed category in each case and 0 elsewhere. Finally a factor with name idvar is added to index the cases.

Value

The expanded data frame as described in Details.

Note

Re-expressing categorical data in this way allows a multinomial response to be modelled as a poisson response, see examples.

Author(s)

Heather Turner

References

See Also

gnm, multinom, reshape, mclgen

Examples

```r
### Example from help(multinom, package = "nnet")
library(MASS)
extample(birthwt)
library(nnet)
bwt.mu <- multinom(low ~ ., data = bwt)

### Equivalent using gnm - include unestimable main effects in model so
### that interactions with low0 automatically set to zero, else could use
### 'constrain' argument.
bwtLong <- expandCategorical(bwt, "low", group = FALSE)
bwt.po <- gnm(count ~ low*(. - id), eliminate = id, data = bwtLong, family =
   "poisson")
summary(bwt.po) # same deviance; df reflect extra id parameters

### Example from ?backpain
set.seed(1)
simmary(backpain)
backPainLong <- expandCategorical(backPain, "pain")

### Fit models described in Table 5 of Anderson (1984)
noRelationship <- gnm(count ~ pain, eliminate = id,
family = "poisson", data = backPainLong)

oneDimensional <- update(noRelationship,
   ~ . + Mult(pain, x1 + x2 + x3))
```

friend

<table>
<thead>
<tr>
<th>Occupation of Respondents and Their Closest Friend</th>
</tr>
</thead>
</table>

Description

Cross-classification of the occupation of respondent and that of their closest friend. Data taken from wave 10 (year 2000) of the British Household Panel Survey.

Usage

friend

Format

A table of counts, with classifying factors r (respondent’s occupational category; levels 1:31) and c (friend’s occupational category; levels 1:31).
getContrasts

Source

Examples

```r
set.seed(1)

### Fit an association model with homogeneous row-column effects
rc1 <- gnm(Freq ~ r + c + Diag(r,c) + MultHomog(r, c),
           family = poisson, data = friend)
rc1

## Not run:
### Extend to two-component interaction
rc2 <- update(rc1, . ~ . + MultHomog(r, c, inst = 2),
              etastart = rc1$predictors)
rc2

## End(Not run)
```

---

getContrasts

*Estimated Contrasts and Standard Errors for Parameters in a gnm Model*

Description
Computes contrasts or scaled contrasts for a set of (non-eliminated) parameters from a gnm model, and computes standard errors for the estimated contrasts. Where possible, quasi standard errors are also computed.

Usage

```r
getContrasts(model, set = NULL, ref = "first", scaleRef = "mean",
             scaleWeights = NULL, dispersion = NULL, check = TRUE, ...)
```

Arguments

- **model** a model object of class "gnm".
- **set** a vector of indices (numeric) or coefficient names (character). If NULL, a dialog will open for parameter selection.
- **ref** either a single numeric index, or a vector of real numbers which sum to 1, or one of the character strings "first", "last" or "mean".
- **scaleRef** as for ref
- **scaleWeights** either NULL, a vector of real numbers, "unit" or "setLength".
dispersion

either NULL, or a positive number by which the model’s variance-covariance matrix should be scaled.

check

TRUE or FALSE or a numeric vector – for which of the specified parameter combinations should estimability be checked? If TRUE, all are checked; if FALSE, none is checked.

... arguments to pass to other functions.

Details

The indices in `set` must all be in `1:length(coef(object))`. If `set = NULL`, a dialog is presented for the selection of indices (model coefficients).

For the set of coefficients selected, contrasts and their standard errors are computed. A check is performed first on the estimability of all such contrasts (if `check = TRUE`) or on a specified subset (if `check` is a numeric index vector). The specific contrasts to be computed are controlled by the choice of `ref`: this may be "first" (the default), for contrasts with the first of the selected coefficients, or "last" for contrasts with the last, or "mean" for contrasts with the arithmetic mean of the coefficients in the selected set; or it may be an arbitrary vector of weights (summing to 1, not necessarily all non-negative) which specify a weighted mean against which contrasts are taken; or it may be a single index specifying one of the coefficients with which all contrasts should be taken.

Thus, for example, `ref = 1` is equivalent to `ref = "first"`, and `ref = c(1/3, 1/3, 1/3)` is equivalent to `ref = "mean"` when there are three coefficients in the selected set.

The contrasts may be scaled by

\[
\frac{1}{\sqrt{\sum v_r d_r^2}}
\]

where \( d_r \) is a contrast of the \( r \)'th coefficient in `set` with the reference level specified by `scaleRef` and \( v \) is a vector of weights (of the same length as `set`) specified by `scaleWeights`. If `scaleWeights` is NULL (the default), `scaleRef` is ignored and no scaling is performed. Other options for `scaleWeights` are "unit" for weights equal to one and "setLength" for weights equal to the reciprocal of `length(set)`. If `scaleRef` is the same as `ref`, these options constrain the sum of squared contrasts to 1 and `length(set)` respectively.

Quasi-variances (and corresponding quasi standard errors) are reported for unscaled contrasts where possible. These statistics are invariant to the choice of `ref`, see Firth (2003) or Firth and Menezes (2004) for more details.

Value

An object of class `qv` — see `qvcalc`.

Author(s)

David Firth and Heather Turner

References


See Also

`gnm, se, checkEstimable, qvcal, ofInterest`

Examples

```r
### Unscaled contrasts ###
set.seed(1)

## Fit the "UNIDIFF" mobility model across education levels -- see ?yaish
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
    Mult(Exp(educ), orig:dest),
    ofInterest = "[.]educ", family = poisson,
    data = yaish, subset = (dest != 7))

## Examine the education multipliers (differences on the log scale):
unidiffContrasts <- getContrasts(unidiff, ofInterest(unidiff))
plot(unidiffContrasts,
    main = "Unidiff multipliers (log scale): intervals based on
    quasi standard errors",
    xlab = "Education level", levelNames = 1:5)

### Scaled contrasts (elliptical contrasts) ###
set.seed(1)

## Goodman Row-Column association model fits well (deviance 3.57, df 8)
mentalHealth$MHS <- C(mentalHealth$MHS, treatment)
mentalHealth$SES <- C(mentalHealth$SES, treatment)
RCImodel <- gnm(count ~ SES + MHS + Mult(SES, MHS),
    family = poisson, data = mentalHealth)

## Row scores and column scores are both unnormalized in this
## parameterization of the model

## The scores can be normalized as in Agresti's eqn (9.15):
rowProbs <- with(mentalHealth, tapply(count, SES, sum) / sum(count))
colProbs <- with(mentalHealth, tapply(count, MHS, sum) / sum(count))
mu <- getContrasts(RCImodel, pickCoef(RCImodel, "[.]SES"),
    ref = rowProbs, scaleRef = rowProbs,
    scaleWeights = rowProbs)

nu <- getContrasts(RCImodel, pickCoef(RCImodel, "[.]MHS"),
    ref = colProbs, scaleRef = colProbs,
    scaleWeights = colProbs)

all.equal(sum(mu$qv[,1] * rowProbs), 0)
all.equal(sum(nu$qv[,1] * colProbs), 0)
all.equal(sum(mu$qv[,1]^2 * rowProbs), 1)
all.equal(sum(nu$qv[,1]^2 * colProbs), 1)
```

----

`gnm`  
*Fitting Generalized Nonlinear Models*
Description

gnm fits generalised nonlinear models using an over-parameterised representation. Nonlinear terms are specified by calls to functions of class "nonlin".

Usage

gnm(formula, eliminate = NULL, ofInterest = NULL, constrain = numeric(0),
    constrainTo = numeric(length(constrain)), family = gaussian,
    data = NULL, subset, weights, na.action, method = "gnmFit",
    checkLinear = TRUE, offset, start = NULL, etastart = NULL,
    mustart = NULL, tolerance = 1e-06, iterStart = 2, iterMax = 500,
    trace = FALSE, verbose = TRUE, model = TRUE, x = TRUE,
    termPredictors = FALSE, ridge = 1e-08, ...)

Arguments

formula a symbolic description of the nonlinear predictor.
eliminate a factor to be included as the first term in the model. gnm will exploit the structure of this factor to improve computational efficiency. See details.
ofInterest optional coefficients of interest, specified by a regular expression, a numeric vector of indices, a character vector of names, or "[?]" to select from a Tk dialog. If missing, it is assumed that all non-eliminated coefficients are of interest.
constrain (non-eliminated) coefficients to constrain, specified by a regular expression, a numeric vector of indices, a logical vector, a character vector of names, or "[?]" to select from a Tk dialog.
constrainTo a numeric vector of the same length as constrain specifying the values to constrain to. By default constrained parameters will be set to zero.
family a specification of the error distribution and link function to be used in the model. This can be a character string naming a family function; a family function, or the result of a call to a family function. See family and wedderburn for possibilities.
data an optional data frame containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which gnm is called.
subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process.
na.action a function which indicates what should happen when the data contain NAs. If data is a contingency table, the default is "exclude". Otherwise the default is first, any na.action attribute of data; second, any na.action setting of options, and third, na.fail.
method the method to be used: either "gnmFit" to fit the model using the default maximum likelihood algorithm, "coefNames" to return a character vector of names for the coefficients in the model, "model.matrix" to return the model matrix, "model.frame" to return the model frame, or the name of a function providing an alternative fitting algorithm.
checkLinear logical: if TRUE glm.fit is used when the predictor is found to be linear
offset this can be used to specify an a priori known component to be added to the predictor during fitting. offset terms can be included in the formula instead or as well, and if both are specified their sum is used.
start a vector of starting values for the parameters in the model; if a starting value is NA, the default starting value will be used. Starting values need not be specified for eliminated parameters.
etastart starting values for the linear predictor.
mustart starting values for the vector of means.
tolerance a positive numeric value specifying the tolerance level for convergence.
iterStart a positive integer specifying the number of start-up iterations to perform.
iterMax a positive integer specifying the maximum number of main iterations to perform.
trace a logical value indicating whether the deviance should be printed after each iteration.
verbose logical: if TRUE and model includes nonlinear terms, progress indicators are printed as the model is fitted, including a diagnostic error message if the algorithm fails.
model logical: if TRUE the model frame is returned.
x logical: if TRUE the local design matrix from the last iteration is included as a component of returned model object.
termPredictors logical: if TRUE, a matrix is returned with a column for each term in the model, containing the additive contribution of that term to the predictor.
ridge numeric, a positive value for the ridge constant to be used in the fitting algorithm
...
... further arguments passed to fitting function.

Details

Models for gnm are specified by giving a symbolic description of the nonlinear predictor, of the form response ~ terms. The response is typically a numeric vector, see later in this section for alternatives. The usual symbolic language may be used to specify any linear terms, see formula for details.

Nonlinear terms may be specified by calls to functions of class "nonlin". There are several "nonlin" functions in the gnm package. Some of these specify simple mathematical functions of predictors: Exp, Mult, and Inv. Others specify more specialised nonlinear terms, in particular MultHomog specifies homogeneous multiplicative interactions and Dref specifies diagonal reference terms. Users may also define their own "nonlin" functions, see nonlin.function for details.

The eliminate argument may be used to specify a factor that is to be included as the first term in the model (since an intercept is then redundant, none is fitted). The structure of the factor is exploited to improve computational efficiency — substantially so if the eliminated factor has a large number of levels. Use of eliminate is designed for factors that are required in the model but are not of direct interest (e.g., terms needed to fit multinomial-response models as conditional Poisson models). See backpain for an example.

The ofInterest argument may be used to specify coefficients of interest, the indices of which are returned in the ofInterest component of the model object. print() displays of the model
object or its components obtained using accessor functions such as \texttt{coef()} etc, will only show these coefficients. In addition methods for "\texttt{gnm}" objects which may be applied to a subset of the parameters are by default applied to the coefficients of interest. See \texttt{ofInterest} for accessor and replacement functions.

For contingency tables, the data may be provided as an object of class "\texttt{table}" from which the frequencies will be extracted to use as the response. In this case, the response should be specified as \texttt{Freq} in the model formula. The "\texttt{predictors"}, "\texttt{fitted.values"}, "\texttt{residuals"}, "\texttt{prior.weights"}, "\texttt{weights"}, "\texttt{y}" and "\texttt{offset}" components of the returned \texttt{gnm} fit will be tables with the same format as the data, completed with \texttt{NAs} where necessary.

For binomial models, the response may be specified as a factor in which the first level denotes failure and all other levels denote success, as a two-column matrix with the columns giving the numbers of successes and failures, or as a vector of the proportions of successes.

The \texttt{gnm} fitting algorithm consists of two stages. In the start-up iterations, any nonlinear parameters that are not specified by either the \texttt{start} argument of \texttt{gnm} or a plug-in function are updated one parameter at a time, then the linear parameters are jointly updated before the next iteration. In the main iterations, all the parameters are jointly updated, until convergence is reached or the number or iterations reaches \texttt{iterMax}. To solve the (typically rank-deficient) least squares problem at the heart of the \texttt{gnm} fitting algorithm, the design matrix is standardized and regularized (in the Levenberg-Marquardt sense) prior to solving; the \texttt{ridge} argument provides a degree of control over the regularization performed (smaller values may sometimes give faster convergence but can lead to numerical instability).

Convergence is judged by comparing the squared components of the score vector with corresponding elements of the diagonal of the Fisher information matrix. If, for all components of the score vector, the ratio is less than \texttt{tolerance}^2, or the corresponding diagonal element of the Fisher information matrix is less than 1e-20, iterations cease. If the algorithm has not converged by \texttt{iterMax} iterations, \texttt{exitInfo} can be used to print information on the parameters which failed the convergence criteria at the last iteration.

By default, \texttt{gnm} uses an over-parameterized representation of the model that is being fitted. Only minimal identifiability constraints are imposed, so that in general a random parameterization is obtained. The parameter estimates are ordered so that those for any linear terms appear first. \texttt{getContrasts} may be used to obtain estimates of specified scaled contrasts, if these contrasts are identifiable. For example, \texttt{getContrasts} may be used to estimate the contrasts between the first level of a factor and the rest, and obtain standard errors.

If appropriate constraints are known in advance, or have been determined from a \texttt{gnm} fit, the model may be (re-)fitted using the \texttt{constrain} argument to specify coefficients which should be set to values specified by \texttt{constraintTo}. Constraints should only be specified for non-eliminated parameters. \texttt{update} provides a convenient way of re-fitting a \texttt{gnm} model with new constraints.

\textbf{Value}

If \texttt{method} = "\texttt{gnmFit}" , \texttt{gnm} returns \texttt{NULL} if the algorithm has failed and an object of class "\texttt{gnm}" otherwise. A "\texttt{gnm}" object inherits first from "\texttt{glm}" then "\texttt{lme}" and is a list containing the following components:

\begin{itemize}
  \item \texttt{call} the matched call.
  \item \texttt{formula} the formula supplied.
\end{itemize}
constrain a numeric vector specifying any coefficients that were constrained in the fitting process.

constrainTo a numeric vector of the same length as constrain specifying the values which constrained parameters were set to.

family the family object used.

prior.weights the case weights initially supplied.

terms the terms object used.

data the data argument.

na.action the na.action attribute of the model frame

xlevels a record of the levels of the factors used in fitting.

y the response used.

offset the offset vector used.

coefficients a named vector of non-eliminated coefficients, with an attribute "eliminated" specifying the eliminated coefficients if eliminate is non-NULL.

eliminate the eliminate argument.

ofInterest a named numeric vector of indices corresponding to non-eliminated coefficients, or NULL.

predictors the fitted values on the link scale.

fitted.values the fitted mean values, obtained by transforming the predictors by the inverse of the link function.

deviance up to a constant, minus twice the maximised log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.

aic Akaike’s An Information Criterion, minus twice the maximized log-likelihood plus twice the number of parameters (so assuming that the dispersion is known).

iter the number of main iterations.

conv logical indicating whether the main iterations converged, with an attribute for use by exitInfo if FALSE.

weights the working weights, that is, the weights used in the last iteration.

residuals the working residuals, that is, the residuals from the last iteration.

df.residual the residual degrees of freedom.

rank the numeric rank of the fitted model.

The list may also contain the components model, x, or termPredictors if requested in the arguments to gnm.

If a table was passed to data and the default for na.action was not overridden, the list will also contain a table.attr component, for use by the extractor functions.

If a binomial gnm model is specified by giving a two-column response, the weights returned by prior.weights are the total numbers of cases (factored by the supplied case weights) and the component y of the result is the proportion of successes.
The function `summary.gnm` may be used to obtain and print a summary of the results, whilst `plot.gnm` may be used for model diagnostics.

The generic functions `formula`, `family`, `terms`, `coefficients`, `fitted.values`, `deviance`, `extractAIC`, `weights`, `residuals`, `df.residual`, `model.frame`, `model.matrix`, `vcov` and `termPredictors` may be used to extract components from the object returned by `gnm` or to construct the relevant objects where necessary.

Note that the generic functions `weights` and `residuals` do not act as straightforward accessor functions for `gnm` objects, but return the prior weights and deviance residuals respectively, as for `glm` objects.

**Note**

Regular expression matching is performed using `grep` with default settings.

**Author(s)**

Heather Turner and David Firth

**References**


**See Also**

`formula` for the symbolic language used to specify formulae.

`Diag` and `Symm` for specifying special types of interaction.

`Exp`, `Mult`, `Inv`, `MultHomog`, `Dref` and `nonlin.function` for incorporating nonlinear terms in the formula argument to `gnm`.

`residuals.glm` and the generic functions `coef`, `fitted`, etc. for extracting components from `gnm` objects.

`exitInfo` to print more information on last iteration when `gnm` has not converged.

`getContrasts` to estimate (identifiable) scaled contrasts from a `gnm` model.

**Examples**

```r
### Analysis of a 4-way contingency table
set.seed(0)
print(cautres)

# Fit a "double UNIDIFF" model with the religion-vote and class-vote
# interactions both modulated by nonnegative election-specific
# multipliers.
doubleUnidiff <- gnm(Freq ~ election:vote + election:class:religion
  + Mult(Exp(election), religion:vote) +
  Mult(Exp(election), class:vote), family = poisson,
  data = cautres)
```
## Examine the multipliers of the class-vote log odds ratios

```r
ofInterest(doubleUnidiff) <- pickCoef(doubleUnidiff, "class:vote[]")
coef(doubleUnidiff)
```

## Coefficients of interest:

- **Mult(Exp(.), class:vote).election1**
  - Estimate: -0.38357138
- **Mult(Exp(.), class:vote).election2**
  - Estimate: 0.29816599
- **Mult(Exp(.), class:vote).election3**
  - Estimate: 0.06580307
- **Mult(Exp(.), class:vote).election4**
  - Estimate: -0.02174104

## Re-parameterize by setting first multiplier to zero

```r
getContrasts(doubleUnidiff, ofInterest(doubleUnidiff))
```

<table>
<thead>
<tr>
<th>Estimate</th>
<th>SE</th>
<th>quasiSE</th>
<th>quasiVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.052232363</td>
<td></td>
</tr>
<tr>
<td>0.08173740</td>
<td>0.24016440</td>
<td>0.005469913</td>
<td></td>
</tr>
<tr>
<td>0.44937450</td>
<td>0.24735210</td>
<td>0.008979340</td>
<td></td>
</tr>
<tr>
<td>0.36183013</td>
<td>0.25347547</td>
<td>0.011956081</td>
<td></td>
</tr>
</tbody>
</table>

## Same thing but with last multiplier as reference category:

```r
getContrasts(doubleUnidiff, rev(ofInterest(doubleUnidiff)))
```

<table>
<thead>
<tr>
<th>Estimate</th>
<th>SE</th>
<th>quasiSE</th>
<th>quasiVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000000</td>
<td>0.00000000</td>
<td>0.052232363</td>
<td></td>
</tr>
<tr>
<td>0.08754436</td>
<td>0.14468330</td>
<td>0.008979340</td>
<td></td>
</tr>
<tr>
<td>0.31990727</td>
<td>0.13200227</td>
<td>0.011956081</td>
<td></td>
</tr>
<tr>
<td>0.36183013</td>
<td>0.25347547</td>
<td>0.005469913</td>
<td></td>
</tr>
<tr>
<td>0.10934798</td>
<td>0.01195608</td>
<td>0.052232363</td>
<td></td>
</tr>
</tbody>
</table>

## Re-fit model with first multiplier set to zero

```r
update(doubleUnidiff, constrain = ofInterest(doubleUnidiff)[1])
```

## Examine the multipliers of the class-vote log odds ratios

```r
coef(doubleUnidiffConstrained)[ofInterest(doubleUnidiff)]
```

...as using 'getContrasts' (to 4 d.p.).
Description

The voting record of every representative in the 2001 House, on 20 roll calls selected by Americans for Democratic Action. Each row is the record of one representative; the first column records the representative’s registered party allegiance.

Usage

House2001

Format

A data frame with 439 observations on the following 21 variables.

- party a factor with levels D I N R
- HR333.BankruptcyOverhaul.Yes a numeric vector
- SJRes6.ErgonomicsRuleDisapproval.No a numeric vector
- HR3.IncomeTaxReduction.No a numeric vector
- HR6.MarriageTaxReduction.Yes a numeric vector
- HR8.EstateTaxRelief.Yes a numeric vector
- HR503.FetalProtection.No a numeric vector
- HR1.SchoolVouchers.No a numeric vector
- HR1836.TaxCutReconciliationBill.No a numeric vector
- HR2356.CampaignFinanceReform.No a numeric vector
- HJRes36.FlagDesecration.No a numeric vector
- HR7.FaithBasedInitiative.Yes a numeric vector
- HJRes50.ChinaNormalizedTradeRelations.Yes a numeric vector
- HR4.ANWDRDrillingBan.Yes a numeric vector
- HR2563.PatientsRightsHMOliability.No a numeric vector
- HR2563.PatientsBillOfRights.No a numeric vector
- HR2944.DomesticPartnerBenefits.No a numeric vector
- HR2586.USMilitaryPersonnelOverseas Abortions.Yes a numeric vector
- HR2975.AntiTerrorismAuthority.No a numeric vector
- HR3090.EconomicStimulus.No a numeric vector
- HR3000.TradePromotionAuthorityFastTrack.No a numeric vector

Details

Coding of the votes is as described in ADA (2002).

Source

References


Examples

```r
## Not run:
## This example takes some time to run!
summary(House2001)

## Put the votes in a matrix, and discard members with too many NAs etc:
House2001m <- as.matrix(House2001[-1])
informative <- apply(House2001m, 1, function(row){
  valid <- !is.na(row)
  validSum <- if (any(valid)) sum(row[valid]) else 0
  nValid <- sum(valid)
  uninformative <- (validSum == nValid) || (validSum == 0) || (nValid < 10)
} !uninformative))
House2001m <- House2001m[informative,]

## Make a vector of colours, blue for Republican and red for Democrat:
parties <- House2001party[informative]
partyColors <- rep("black", length(parties))
partyColors <- ifelse(parties == "D", "red", partyColors)
partyColors <- ifelse(parties == "R", "blue", partyColors)

## Expand the data for statistical modelling:
House2001v <- as.vector(House2001m)
House2001f <- data.frame(member = rownames(House2001m),
                         party = parties,
                         rollcall = factor(rep((1:20), rep(nrow(House2001m), 20))),
                         vote = House2001v)

## Now fit an "empty" model, in which all members vote identically:
baseModel <- glm(vote ~ 1 + rollcall, family = binomial, data = House2001f)
## From this, get starting values for a one-dimensional multiplicative term:
Start <- residSVD(baseModel, rollcall, member)

## Now fit the logistic model with one multiplicative term.
## For the response variable, instead of vote=0,1 we use 0.03 and 0.97,
## corresponding approximately to a bias-reducing adjustment of p/(2n),
## where p is the number of parameters and n the number of observations.
##
voteAdj <- 0.5 + 0.94*(House2001f$vote - 0.5)

House2001model1 <- gnm(voteAdj ~ Mult(rollcall, member),
                         eliminate = rollcall,
                         family = binomial, data = House2001f,
                         na.action = na.include, trace = TRUE, tolerance = 1e-03,
                         start = -Start)
## Deviance is 2234.847, df = 5574

## Plot the members' positions as estimated in the model:
```
instances <- Specify Multiple Instances of a Nonlinear Term in a gnm Model Formula

Description

A symbolic wrapper, for use in the formula argument to gnm, to specify multiple instances of a term specified by a function with an inst argument.

Usage

instances(term, instances = 1)

Arguments

term a call to a function with an inst argument, which specifies some term.
instances the desired number of instances of the term.
Value

A deparsed expression representing the summation of term specified with inst = 1, inst = 2, ..., inst = instances, which is used to create an expanded formula.

Author(s)

Heather Turner

See Also

gnm, formula, nonlin.function, Mult, MultHomog

Examples

```r
## Not run:
## (this example can take quite a while to run)
##
## Fitting two instances of a multiplicative interaction (i.e. a two-component interaction)
yield.scaled <- wheat$yield * sqrt(3/1000)
treatment <- factor(paste(wheat$tillage, wheat$summerCrop, wheat$manure, wheat$N, sep = ""))
bilinear2 <- gnm(yield.scaled ~ year + treatment +
                 instances(Mult(year, treatment), 2),
                 family = gaussian, data = wheat)

## End(Not run)
```

Description

A function of class "nonlin" to specify the reciprocal of a predictor in the formula argument to gnm.

Usage

```r
Inv(expression, inst = NULL)
```

Arguments

expression a symbolic expression representing the (possibly nonlinear) predictor.
inst (optional) an integer specifying the instance number of the term.
Details
The expression argument is interpreted as the right hand side of a formula in an object of class "formula", except that an intercept term is not added by default. Any function of class "nonlin" may be used in addition to the usual operators and functions.

Value
A list with the components required of a "nonlin" function:

- predictors: the expression argument passed to \texttt{Inv}
- term: a function to create a deparsed mathematical expression of the term, given a label for the predictor.
- call: the call to use as a prefix for parameter labels.

Author(s)
Heather Turner

See Also
\texttt{gnm}, \texttt{formula}, \texttt{nonlin.function}

Examples

\begin{verbatim}
## One way to fit the logistic function without conditional
## linearity as in \texttt{nls}
library(gnm)
set.seed(1)
DNase1 <- subset(DNase, Run == 1)

test <- gnm(density ~ Mult(1, Inv(Const(!)) + Exp(Mult(! + offset(-log(conc)),
Inv(!)))),
start = c(NA, 0, 1), data = DNase1, trace = TRUE)
coef(test)
\end{verbatim}

---

\texttt{meanResiduals} \hspace{1cm} \textit{Average Residuals within Factor Levels}

Description
Computes the mean working residuals from a model fitted using Iterative Weighted Least Squares for each level of a factor or interaction of factors.

Usage
\begin{verbatim}
meanResiduals(object, by, standardized=TRUE, as.table=TRUE, ...)
\end{verbatim}
meanResiduals

Arguments

object  model object for which object$residuals gives the working residuals and object$weights gives the working weights.

by  either a formula specifying a factor or interaction of factors (recommended), or a list of factors (the elements of which must correspond exactly to observations in the model frame). When a list of factors is specified, their interaction is used to specify the grouping factor.

standardized  logical: if TRUE, the mean residuals are standardized to be approximately standard normal.

as.table  logical: logical: if TRUE and by specifies an interaction of factors, the result is returned as a table cross-classified by these factors.

Value

An object of class "meanResiduals", for which print and summary methods are provided. A "meanResiduals" object is a list containing the following elements:

call  the call used to create the model object from which the mean residuals are derived.

by  a label for the grouping factor.

residuals  the mean residuals.

df  the degrees of freedom associated with the mean residuals.

standardized  the standardized argument.

weights  the weights corresponding to the mean residuals.

Details

For level $i$ of the grouping factor $A$ the mean working residual is defined as

$$ r_{ij} * w_{ij} $$

$$ \sum_{j=1}^{n_i} w_{ij} $$

where $r_{ij}$ is the $j$'th residual for level $i$, $w_{ij}$ is the corresponding working weight and $n_i$ is the number of observations for level $i$. The denominator gives the weight corresponding to mean residual.

For non-aggregated residuals, i.e. when the factor has one level per observation, the residuals are the same as Pearson residuals.

Author(s)

Heather Turner
mentalHealth

Examples

```r
## Fit a conditional independence model, leaving out
## the uninformative subtable for dest == 7:
CImodel <- gnm(Freq ~ educ*orig + educ*dest, family = poisson, 
data = yaish, subset = (dest != 7))

## compute mean residuals over origin and destination
meanRes <- meanResiduals(CImodel, - orig:dest)
meanRes
summary(meanRes)

## Not run:
## requires vcdExtra package
## display mean residuals for origin and destination
library(vcdExtra)
mosaic(CImodel, -orig+dest)

## End(Not run)

## non-aggregated residuals
res1 <- meanResiduals(CImodel, - educ:orig:dest)
res2 <- residuals(CImodel, type = "pearson")
all.equal(as.numeric(res1), as.numeric(res2))
```

---

### mentalHealth

**Data on Mental Health and Socioeconomic Status**

#### Description

A 2-way contingency table from a sample of residents of Manhattan. Classifying variables are child’s mental impairment (MHS) and parents’ socioeconomic status (SES).

#### Usage

`mentalHealth`

#### Format

A data frame with 24 observations on the following 3 variables.

- `count` a numeric vector
- `SES` an ordered factor with levels `a < b < c < d < e < f`
- `MHS` an ordered factor with levels `well < mild < moderate < impaired`

#### Source

From Agresti (2002, p381); originally in Srole et al. (1978, p289).
References


Examples

```r
set.seed(1)

## Goodman Row-Column association model fits well (deviance 3.57, df 8)
mentalHealth$MHS <- C(mentalHealth$MHS, treatment)
mentalHealth$SES <- C(mentalHealth$SES, treatment)
RCImodel <- gnm(count ~ SES + MHS + Mult(SES, MHS),
               family = poisson, data = mentalHealth)
## Row scores and column scores are both unnormalized in this
## parameterization of the model

## The scores can be normalized as in Agresti's eqn (9.15):
rowProbs <- with(mentalHealth, tapply(count, SES, sum) / sum(count))
colProbs <- with(mentalHealth, tapply(count, MHS, sum) / sum(count))
mu <- getContrasts(RCImodel, pickCoef(RCImodel, "[.]SES"),
                   ref = rowProbs, scaleRef = rowProbs,
                   scaleWeights = rowProbs)
nu <- getContrasts(RCImodel, pickCoef(RCImodel, "[.]MHS"),
                   ref = colProbs, scaleRef = colProbs,
                   scaleWeights = colProbs)
all.equal(sum(mu$sv[,1] * rowProbs), 0)
all.equal(sum(nu$sv[,1] * colProbs), 0)
all.equal(sum(mu$sv[,1]^2 * rowProbs), 1)
all.equal(sum(nu$sv[,1]^2 * colProbs), 1)
```

---

**model.matrix.gnm**

*Local Design Matrix for a Generalized Nonlinear Model*

**Description**

This method extracts or evaluates a local design matrix for a generalized nonlinear model.

**Usage**

```r
## S3 method for class 'gnm'
model.matrix(object, coef = NULL, ...)
```

**Arguments**

- `object` an object of class gnm.
- `coef` if specified, the vector of (non-eliminated) coefficients at which the local design matrix is evaluated.
- `...` further arguments.
**Value**

If \( \text{coef} = \text{NULL} \), the local design matrix with columns corresponding to the non-eliminated parameters evaluated at \( \text{coef(object)} \) (extracted from \( \text{object} \) if possible).

Otherwise, the local design matrix evaluated at \( \text{coef} \).

**Author(s)**

Heather Turner

**See Also**

gnm, model.matrix

**Examples**

```r
example(mentalHealth)
model.matrix(RC1model)
model.matrix(RC1model, coef = seq(coef(RC1model)))
```

---

**MPinv**

**Moore-Penrose Pseudoinverse of a Real-valued Matrix**

**Description**

Computes the Moore-Penrose generalized inverse.

**Usage**

```r
MPinv(mat, tolerance = 100*.Machine$double.eps,
       rank = NULL, method = "svd")
```

**Arguments**

- `mat`  
a real matrix.
- `tolerance`  
A positive scalar which determines the tolerance for detecting zeroes among the singular values.
- `rank`  
Either `NULL`, in which case the rank of `mat` is determined numerically; or an integer specifying the rank of `mat` if it is known. No check is made on the validity of any non-NULL value.
- `method`  
Character, one of "svd", "chol". The specification `method = "chol"` is valid only for symmetric matrices.

**Details**

Real-valuedness is not checked, neither is symmetry when `method = "chol"`. 
Value

A matrix, with an additional attribute named "rank" containing the numerically determined rank of the matrix.

Author(s)

David Firth and Heather Turner

References


See Also

ginv

Examples

```r
A <- matrix(c(1, 1, 0,
              1, 1, 0,
              2, 3, 4), 3, 3)
B <- MPinv(A)
A %*% B %*% A - A # essentially zero
B %*% A %*% B - B # essentially zero
attr(B, "rank") # here 2

## demonstration that "svd" and "chol" deliver essentially the same
## results for symmetric matrices:
A <- crossprod(A)
MPinv(A) - MPinv(A, method = "chol") # (essentially zero)
```

Description

A function of class "nonlin" to specify a multiplicative interaction with homogeneous effects in the formula argument to *gmm*.

Usage

```r
MultHomog(..., inst = NULL)
```
Arguments

... a comma-separated list of two or more factors.

inst (optional) an integer specifying the instance number of the term.

Details

`MultHomog` specifies instances of a multiplicative interaction in which the constituent multipliers are the effects of two or more factors and the effects of these factors are constrained to be equal when the factor levels are equal. Thus the interaction effect would be

\[ \gamma_i \gamma_j \ldots \]

for an observation with level \( i \) of the first factor, level \( j \) of the second factor and so on, where \( \gamma_l \) is the effect for level \( l \) of the homogeneous multiplicative factor.

If the factors passed to `MultHomog` do not have exactly the same levels, the set of levels is taken to be the union of the factor levels, sorted into increasing order.

Value

A list with the anticipated components of a "nonlin" function:

- predictors the factors passed to `MultHomog`
- common an index to specify that common effects are to be estimated across the factors
- term a function to create a deparsed mathematical expression of the term, given labels for the predictors.
- call the call to use as a prefix for parameter labels.

Note

Currently, `MultHomog` can only be used to specify a one-dimensional interaction. See examples for a workaround to specify interactions with more than one dimension.

Author(s)

Heather Turner

References


See Also

`gnm, formula, instances, nonlin.function, Mult`
Multiplicative interaction

Examples

```r
set.seed(1)

### Fit an association model with homogeneous row-column effects
rc1 <- gnm(Freq ~ r + c + Diag(r,c) + MultHomog(r, c),
            family = poisson, data = friend)
rc1

## Not run:
### Extend to two-component interaction
rc2 <- update(rc1, ~ . + MultHomog(r, c, inst = 2),
              etastart = rc1$predictors)
rc2

## End(Not run)

### For factors with a large number of levels, save time by
### setting diagonal elements to NA rather than fitting exactly;
### skipping start-up iterations may also save time
dat <- as.data.frame(friend)
id <- with(dat, r == c)
dat[id,] <- NA
rc2 <- gnm(Freq ~ r + c + instances(MultHomog(r, c), 2),
           family = poisson, data = dat, iterStart = 0)
```

---

**Multiplicative interaction**

*Specify a Product of Predictors in a gnm Model Formula*

---

Description

A function of class "nonlin" to specify a multiplicative interaction in the formula argument to `gnm`.

Usage

```r
Mult(..., inst = NULL)
```

Arguments

- `...` a comma-separated list of two or more symbolic expressions representing the constituent multipliers in the interaction.
- `inst` a positive integer specifying the instance number of the term.
Multiplicative interaction

Details

Mult specifies instances of a multiplicative interaction, i.e. a product of the form

\[ m_1 m_2 \ldots m_n, \]

where the constituent multipliers \( m_1, m_2, \ldots, m_n \) are linear or nonlinear predictors.

Models for the constituent multipliers are specified symbolically as unspecified arguments to Mult. These symbolic expressions are interpreted in the same way as the right hand side of a formula in an object of class "formula", except that an intercept term is not added by default. Offsets can be added to constituent multipliers, using offset.

The family of multiplicative interaction models include row-column association models for contingency tables (e.g., Agresti, 2002, Sec 9.6), log-multiplicative or UNIDIFF models (Erikson and Goldthorpe, 1992; Xie, 1992), and GAMMI models (van Eeuwijk, 1995).

Value

A list with the required components of a "nonlin" function:

- predictors: the expressions passed to Mult
- term: a function to create a deparsed mathematical expression of the term, given labels for the predictors.
- call: the call to use as a prefix for parameter labels.

Author(s)

Heather Turner

References


See Also

gnm, formula, instances, nonlin.function, MultHomog
Examples

set.seed(1)

## Using 'Mult' with 'Exp' to constrain the first constituent multiplier
## to be non-negative

## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
    Mult(Exp(educ), orig:dest),
    family = poisson, data = yaish, subset = (dest != 7))

## Not run:
## (this example can take quite a while to run)
##
## # Fitting two instances of a multiplicative interaction (i.e. a
## # two-component interaction)
## yield.scaled <- wheat$yield * sqrt(3/1000)
treatment <- factor(paste(wheat$tillage, wheat$summerCrop, wheat$manure,
                      wheat$N, sep = ""))
bilinear2 <- gnm(yield.scaled ~ year + treatment +
                    instances(Mult(year, treatment), 2),
                    family = gaussian, data = wheat)
formula(bilinear2)
## yield.scaled ~ year + treatment + Mult(year, treatment, inst = 1) +
##    Mult(year, treatment, inst = 2)

## End(Not run)

---

nonlin.function  

Functions to Specify Nonlinear Terms in gnm Models

Description

Nonlinear terms maybe be specified in the formula argument to gnm by a call to a function of class "nonlin". A "nonlin" function takes a list of arguments and returns a list of arguments for the internal nonlinTerms function.

Arguments

...  
    arguments required to define the term, e.g. symbolic representations of predictors in the term.

inst  
    (optional) an integer specifying the instance number of the term - for compatibility with instances.

Value

The function should return a list with the following components:
predictors          a list of symbolic expressions or formulae with no left hand side which represent
(possibly nonlinear) predictors that form part of the term. Intercepts will be
added by default to predictors specified by formulae. If predictors are named,
these names will be used as a prefix for parameter labels or the parameter label
itself in the single parameter case (in either case, prefixed by the call if supplied.)
Predictors that may include an intercept should always be named or matched to
a call.

variables          an optional list of expressions representing variables in the term.

term               a function which takes the arguments predLabels and varLabels, which are
vectors of labels defined by gnm that correspond to the specified predictors and
variables, and returns a deparsed mathematical expression of the full term. Only
functions recognised by deriv should be used in the expression, e.g. + rather
than sum.

common            an optional numeric index of predictors with duplicated indices identifying
single factor predictors for which homologous effects are to be estimated.

call              an optional call to be used as a prefix for parameter labels, specified as an R
expression.

match             (if call is non-NULL) a numeric index of predictors specifying which argu-
ments of call the predictors match to - zero indicating no match. If NULL,
predictors will not be matched. It is recommended that matches are specified
wherever possible, to ensure parameter labels are well-defined. Parameters in
matched predictors are labelled using "dot-style" labelling, see examples.

start             an optional function which takes a named vector of parameters corresponding to
the predictors and returns a vector of starting values for those parameters. This
function is ignored if the term is nested within another nonlinear term.

Author(s)
Heather Turner

See Also
Const to specify a constant, Dref to specify a diagonal reference term, Exp to specify the exponen-
tial of a predictor, Inv to specify the reciprocal of a predictor,
Mult to specify a multiplicative interaction, MultHomog to specify a homogeneous multiplicative
interaction,

Examples
### Equivalent of weighted.MM function in ?nls
weighted.MM <- function(resp, conc){
  list(predictors = list(Vm = substitute(conc), K = 1),
     variables = list(substitute(resp), substitute(conc)),
     term = function(predictors, variables) {
       pred <- paste("\n", predictors[1], "/\n", predictors[2], "+ \n", variables[2], ")", sep = "")
       pred <- paste("\n", variables[1], " - \n", pred, ")/sqrt(",
                        pred, "\n")", sep = "")
  })

nonlin.function

```r
## use to fitted weighted Michaelis-Menten model
Treated <- Puromycin[state == "treated", ]
Pur.wt.2 <- gnm(-1 + weighted.MM(rate, conc), data = Treated,
               start = c(Vm = 200, K = 0.1), verbose = FALSE)
Pur.wt.2
## Call:
gnm(formula = ~1 + weighted.MM(rate, conc), data = Treated,
     start = c(Vm = 200, K = 0.1), verbose = FALSE)
## Coefficients:
## Vm        K
## 206.83477  0.05461
## Deviance: 14.59690
## Pearson chi-squared: 14.59690
## Residual df: 10
##
## The definition of MultHomog
MultHomog <- function(..., inst = NULL){
  dots <- match.call(expand.dots = FALSE)["..."]
  list(predictors = dots,
       common = rep(1, length(dots)),
       term = function(predictors, ...) {
         paste("(" , paste(predictors, collapse = ", " ) , ", ", sep = "")
       },
       call = as.expression(match.call())
  }
}
class(MultHomog) <- "nonlin"
## use to fit homogeneous multiplicative interaction
set.seed(1)
RChomog <- gnm(Freq ~ origin + destination + Diag(origin, destination) +
               MultHomog(origin, destination), ofInterest = "MultHomog",
               family = poisson, data = occupationalStatus,
               verbose = FALSE)
RChomog
## Call:
gnm(formula = Freq ~ origin + destination + Diag(origin, destination) +
     MultHomog(origin, destination), ofInterest = "MultHomog", family = poisson,
     data = occupationalStatus, verbose = FALSE)
## Coefficients of interest:
## MultHomog(origin, destination)1
## -1.50089
## MultHomog(origin, destination)2
## -1.28260
## MultHomog(origin, destination)3
```
ofInterest

Coefficients of Interest in a Generalized Nonlinear Model

```r
## -0.68443
## MultHomog(origin, destination)4
## -0.10055
## MultHomog(origin, destination)5
## -0.08338
## MultHomog(origin, destination)6
## 0.42838
## MultHomog(origin, destination)7
## 0.84452
## MultHomog(., .).`origin|destination'8
## 1.08809
##
## Deviance: 32.56098
## Pearson chi-squared: 31.20716
## Residual df: 34
##
## the definition of Exp
## Exp <- function(expression, inst = NULL){
##   list(predictors = list(substitute(expression)),
##       term = function(predictors, ...){
##         paste("exp(" , predictors , ")", sep = "")
##       },
##       call = as.expression(match.call()),
##       match = 1)
##   class(Exp) <- "nonlin"
##}
##
## use to fit exponential model
## x <- 1:100
## y <- exp(- x / 10)
## set.seed(4)
## exp1 <- gnm(y ~ Exp(1 + x), verbose = FALSE)
## exp1
##
## Call:
## gnm(formula = y ~ Exp(1 + x), verbose = FALSE)
##
## Coefficients:
## (Intercept) Exp(. + x).(Intercept)
## 1.549e-11 -7.934e-11
## Exp(1 + .).x
## -1.000e-01
##
## Deviance: 9.342418e-20
## Pearson chi-squared: 9.342418e-20
## Residual df: 97
```
Description

Retrieve or set the "ofInterest" component of a "gnm" (generalized nonlinear model) object.

Usage

ofInterest(object)
ofInterest(object) <- value

Arguments

object an object of class "gnm".
value a numeric vector of indices specifying the subset of (non-eliminated) coefficients of interest, or NULL to specify that all non-eliminated coefficients are of interest.

Details

The "ofInterest" component of a "gnm" object is a named numeric vector of indices specifying a subset of the non-eliminated coefficients which are of specific interest.

If the "ofInterest" component is non-NULL, printed summaries of the model only show the coefficients of interest. In addition methods for "gnm" objects which may be applied to a subset of the parameters are by default applied to the coefficients of interest.

These functions provide a way of extracting and replacing the "ofInterest" component. The replacement function prints the replacement value to show which parameters have been specified by value.

Value

A named vector of indices, or NULL.

Note

Regular expression matching is performed using grep with default settings.

Author(s)

Heather Turner

See Also

grep, gnm, se, getContrasts, profile.gnm, confint.gnm

Examples

set.seed(1)

## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
                Mult(Exp(educ), orig:dest),
                data = mobility, method = "Newton")
ofInterest = c("[.]educ", family = poisson, 
data = yaish, subset = (dest != 7))
ofInterest(unidiff)

## Get all of the contrasts with educ1 in the UNIDIFF multipliers
getContrasts(unidiff, ofInterest(unidiff))

## Get estimate and se for the contrast between educ4 and educ5 in the
## UNIDIFF multiplier
mycontrast <- numeric(length(coef(unidiff)))
mycontrast[ofInterest(unidiff)[4:5]] <- c(1, -1)
se(unidiff, mycontrast)

---

parameters

Extract Constrained and Estimated Parameters from a gnm Object

Description

A function to extract non-eliminated parameters from a "gnm" object, including parameters that were constrained.

Usage

parameters(object)

Arguments

object an object of class "gnm".

Details

parameters acts like coefficients except that for constrained parameters, the value at which the parameter was constrained is returned instead of NA.

Value

A vector of parameters.

Author(s)

Heather Turner

See Also

coefficients.gnm
Examples

RChomog <- glm(Freq ~ origin + destination + Diag(origin, destination) + 
MultHomog(origin, destination), family = poisson, 
data = occupationalStatus, ofInterest = "MultHomog", 
constrain = "MultHomog.*1")
coefficients(RChomog)
parameters(RChomog)

pickCoef

Get Indices or Values of Selected Model Coefficients

Description

Get the indices or values of a subset of non-eliminated coefficients selected via a Tk dialog or by pattern matching.

Usage

pickCoef(object, pattern = NULL, value = FALSE, ...)

Arguments

object      a model object.
pattern     character string containing a regular expression or (with fixed = TRUE) a pattern to be matched exactly. If missing, a Tk dialog will open for coefficient selection.
value       if FALSE, a named vector of indices, otherwise the value of the selected coefficients.
...         arguments to pass on to pickFrom if pattern is missing, otherwise grep. In particular, fixed = TRUE specifies that pattern is a string to be matched as is.

Value

If value = FALSE (the default), a named vector of indices, otherwise the values of the selected coefficients. If no coefficients are selected the returned value will be NULL.

Author(s)

Heather Turner

See Also

regexp, grep, pickFrom, ofInterest
Examples

set.seed(1)

### Extract indices for use with ofInterest

## fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
               Mult(Exp(educ), orig:dest),
               family = poisson, data = yaish, subset = (dest != 7))

## set coefficients in first constituent multiplier as 'ofInterest'
## using regular expression
ofInterest(unidiff) <- pickCoef(unidiff, ".educ")

## summarise model, only showing coefficients of interest
summary(unidiff)

## get contrasts of these coefficients
getContrasts(unidiff, ofInterest(unidiff))

### Extract coefficients to use as starting values

## fit diagonal reference model with constant weights
set.seed(1)

## reconstruct counts voting Labour/non-Labour
count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting$total - count)

classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                     family = binomial, data = voting)

## create factors indicating movement in and out of salariat (class 1)
upward <- with(voting, origin != 1 & destination == 1)
downward <- with(voting, origin == 1 & destination != 1)

## extract diagonal effects from first model to use as starting values
diagCoef <- pickCoef(classMobility, "Dref(., .)", fixed = TRUE,
                      value = TRUE)

## fit separate weights for the "socially mobile" groups
## -- there are now 3 parameters for each weight
socialMobility <- gnm(yvar ~ -1 + Dref(origin, destination,
                       delta = ~ 1 + downward + upward),
                      family = binomial, data = voting,
                      start = c(rep(NA, 6), diagCoef))
Description

Five plots are available: a plot of residuals against fitted values, a Scale-Location plot of \( \sqrt{|\text{residuals}|} \) against fitted values, a Normal Q-Q plot, a plot of Cook’s distances versus row labels, and a plot of residuals against leverages. By default, all except the fourth are produced.

Usage

```r
## S3 method for class 'gnm'
plot(x, which = c(1:3, 5), caption = c("Residuals vs Fitted", "Normal Q-Q", "Scale-Location", "Cook's distance", "Residuals vs Leverage"),
panel = if (add.smooth) panel.smooth else points,
sub.caption = NULL, main = "",
ask = prod(par("mfcol")) < length(which) && dev.interactive(),
..., id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75,
qqline = TRUE, cook.levels = c(0.5, 1),
add.smooth = getOption("add.smooth"), label.pos = c(4, 2),
cex.caption = 1)
```

Arguments

- `x` a "gnm" object.
- `which` a subset of the numbers 1:5 specifying which plots to produce (out of those listed in Description section).
- `caption` captions to appear above the plots.
- `panel` panel function. The useful alternative to `points`, `panel.smooth` can be chosen by `add.smooth = TRUE`.
- `sub.caption` common title - above figures if there are multiple; used as `sub (s.title)` otherwise. If `NULL`, as by default, a possible shortened version of `deparse(x$call)` is used.
- `main` title to each plot - in addition to the above `caption`.
- `ask` logical; if `TRUE`, the user is asked before each plot, see `par(ask = .)`.
- `...` other parameters to be passed through to plotting functions.
- `id.n` number of points to be labelled in each plot starting with the most extreme.
- `labels.id` vector of labels, from which the labels for extreme points will be chosen. `NULL` uses observation numbers.
- `cex.id` magnification of point labels.
- `qqline` logical indicating if a `qqline()` should be added to the normal Q-Q plot.
- `cook.levels` levels of Cook’s distance at which to draw contours.
- `add.smooth` logical indicating if a smoother should be added to most plots; see also panel above.
- `label.pos` positioning of labels, for the left half and right half of the graph respectively, for plots 1-3.
- `cex.caption` controls the size of ‘caption’.
Details

sub.caption - by default the function call - is shown as a subtitle (under the x-axis title) on each plot when plots are on separate pages, or as a subtitle in the outer margin (if any) when there are multiple plots per page.

The "Scale-Location" plot, also called "Spread-Location" or "S-L" plot, takes the square root of the absolute residuals in order to diminish skewness ($\sqrt{|E|}$ is much less skewed than $|E|$ for Gaussian zero-mean $E$).

The S-L, the Q-Q, and the Residual-Leverage plot, use standardized residuals which have identical variance (under the hypothesis). They are given as $R[i]/(s \sqrt{\hat{h}_{ii}(1 - \hat{h}_{ii})})$ where $\hat{h}_{ii}$ are the diagonal entries of the hat matrix, influence()$hat, see also hat.

The Residual-Leverage plot shows contours of equal Cook's distance, for values of cook.levels (by default 0.5 and 1) and omits cases with leverage one. If the leverages are constant, as typically in a balanced aov situation, the plot uses factor level combinations instead of the leverages for the x-axis.

Author(s)

Modification of plot.lm by the R Core Team. Adapted for "gnm" objects by Heather Turner.

See Also

gnm, plot.lm

Examples

set.seed(1)

## Fit an association model with homogeneous row-column effects
RChomog <- gnm(Freq ~ origin + destination + Diag(origin, destination) +
             MultHomog(origin, destination), family = poisson,
             data = occupationalStatus)

## Plot model diagnostics
plot(RChomog)

## Put 4 plots on 1 page; allow room for printing model formula in outer margin:
par(mfrow = c(2, 2), oma = c(0, 0, 3, 0))
title <- paste(deparse(RChomog$formula, width.cutoff = 50), collapse = "\n")
plot(RChomog, sub.caption = title)

## Fit smoother curves
plot(RChomog, sub.caption = title, panel = panel.smooth)
plot(RChomog, sub.caption = title, panel = function(x,y) panel.smooth(x, y, span = 1))
Description

Obtains predictions and optionally estimates standard errors of those predictions from a fitted generalized nonlinear model object.

Usage

```r
## S3 method for class 'gnm'
predict(object, newdata = NULL,
        type = c("link", "response", "terms"), se.fit = FALSE, dispersion =
        NULL, terms = NULL, na.action = na.exclude, ...)
```

Arguments

- **object**: a fitted object of class inheriting from "gnm".
- **newdata**: optionally, a data frame in which to look for variables with which to predict. If omitted, the fitted predictors are used.
- **type**: the type of prediction required. The default is on the scale of the predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the predictor scale. The value of this argument can be abbreviated.
- **se.fit**: logical switch indicating if standard errors are required.
- **dispersion**: the dispersion of the fit to be assumed in computing the standard errors. If omitted, that returned by `summary` applied to the object is used.
- **terms**: with type="terms" by default all terms are returned. A character vector specifies which terms are to be returned
- **na.action**: function determining what should be done with missing values in `newdata`. The default is to predict NA.
- **...**: further arguments passed to or from other methods.

Details

If `newdata` is omitted the predictions are based on the data used for the fit. In that case how cases with missing values in the original fit is determined by the `na.action` argument of that fit. If `na.action = na.omit` omitted cases will not appear in the residuals, whereas if `na.action = na.exclude` they will appear (in predictions and standard errors), with residual value NA. See also `napredict`. 
Value

If `se = FALSE`, a vector or matrix of predictions. If `se = TRUE`, a list with components

- `fit`: predictions.
- `se.fit`: estimated standard errors.
- `residual.scale`: a scalar giving the square root of the dispersion used in computing the standard errors.

Note

Variables are first looked for in 'newdata' and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in 'newdata' if it was supplied.

Author(s)

Heather Turner

References


See Also

gnm

Examples

```r
set.seed(1)

## Fit an association model with homogeneous row-column effects
RChomog <- gnm(Freq ~ origin + destination + Diag(origin, destination) + MultHomog(origin, destination), family = poisson, data = occupationalStatus)

## Fitted values (expected counts)
predict(RChomog, type = "response", se.fit = TRUE)

## Fitted values on log scale
predict(RChomog, type = "link", se.fit = TRUE)
```
Profile Deviance for Parameters in a Generalized Nonlinear Model

Description

For one or more parameters in a generalized nonlinear model, profile the deviance over a range of values about the fitted estimate.

Usage

```r
## S3 method for class 'gnm'
profile(fitted, which = ofInterest(fitted), alpha = 0.05, maxsteps = 10,
        stepsize = NULL, trace = FALSE, ...)
```

Arguments

- `fitted`: an object of class "gnm".
- `which`: (optional) either a numeric vector of indices or a character vector of names, specifying the parameters over which the deviance is to be profiled. If missing, the deviance is profiled over all parameters.
- `alpha`: the significance level of the z statistic, indicating the range that the profile must cover (see details).
- `maxsteps`: the maximum number of steps to take either side of the fitted parameter.
- `stepsize`: (optional) a numeric vector of length two, specifying the size of steps to take when profiling down and up respectively, or a single number specifying the step size in both directions. If missing, the step sizes will be determined automatically.
- `trace`: logical, indicating whether profiling should be traced.
- `...`: further arguments.

Details

This is a method for the generic function `profile` in the base package.

For a given parameter, the deviance is profiled by constraining that parameter to certain values either side of its estimate in the fitted model and refitting the model.

For each updated model, the following "z statistic" is computed

\[ z(\theta) = (\hat{\theta} - \theta) \cdot \sqrt{\frac{D_{\theta} - D_{\hat{\theta}}}{\delta}} \]

where \( \theta \) is the constrained value of the parameter; \( \hat{\theta} \) is the original fitted value; \( D_{\theta} \) is the deviance when the parameter is equal to \( \theta \), and \( \delta \) is the dispersion parameter.

When the deviance is quadratic in \( \theta \), \( z \) will be linear in \( \theta \). Therefore departures from quadratic behaviour can easily be identified by plotting \( z \) against \( \theta \) using `plot.profile.gnm`. 
confint.profile.gnm estimates confidence intervals for the parameters by interpolating the deviance profiles and identifying the parameter values at which $z$ is equal to the relevant percentage points of the normal distribution. The alpha argument to profile.gnm specifies the significance level of $z$ which must be covered by the profile. In particular, the profiling in a given direction will stop when maxsteps is reached or two steps have been taken in which

$$z(\theta) > (\theta - \hat{\theta} \ast z(1-\alpha)/2$$

By default, the stepsize is

$$z(1-\alpha)/2 \ast s_{\hat{\theta}}$$

where $s_{\hat{\theta}}$ is the standard error of $\hat{\theta}$. Strong asymmetry is detected and the stepsize is adjusted accordingly, to try to ensure that the range determined by alpha is adequately covered. profile.gnm will also attempt to detect if the deviance is asymptotic such that the desired significance level cannot be reached. Each profile has an attribute "asymptote", a two-length logical vector specifying whether an asymptote has been detected in either direction.

For unidentified parameters the profile will be NA, as such parameters cannot be profiled.

Value

A list of profiles, with one named component for each parameter profiled. Each profile is a data.frame: the first column, "z", contains the z statistics and the second column "par.vals" contains a matrix of parameter values, with one column for each parameter in the model.

The list has two attributes: "original.fit" containing fitted and "summary" containing summary(fitted).

Author(s)

Modification of profile.glm from the MASS package. Originally D. M. Bates and W. N. Venables, ported to R by B. D. Ripley, adapted for "gnm" objects by Heather Turner.

References


See Also

confint.gnm, gnm, profile.glm, ofInterest

Examples

```r
set.seed(1)

### Example in which deviance is near quadratic
count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting$total - count)
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                     constrain = "delta1", family = binomial,
                     data = voting)
prof <- profile(classMobility, trace = TRUE)
plot(prof)
```
## Not run:
### These examples take longer to run
### Another near quadratic example

```r
cchomog <- gnm(Freq ~ origin + destination + diag(origin, destination) +
               MultHomog(origin, destination),
               ofInterest = "MultHomog", constrain = "MultHomog.*1",
               family = poisson, data = occupationalStatus)

prof <- profile(cchomog, trace = TRUE)
plot(prof)
```

### Another near quadratic example, with more complex constraints

```r
count <- with(voting, percentage/100 * total)
ycricket <- cbind(count, voting$total - count)
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                      family = binomial, data = voting)

wts <- prop.table(exp(coef(classMobility)))[1:2]
classMobility <- update(classMobility, constrain = "delta1",
                         constrainTo = log(wts[1]))

sum(exp(parameters(classMobility)))[1:2] #=1

prof <- profile(classMobility, trace = TRUE)
plot(prof)
```

### An example showing asymptotic deviance

```r
cunidiff <- gnm(Freq ~ educ*orig + educ*dest +
                Mult(Exp(educ), orig:dest),
                ofInterest = "[.]educ", constrain = "[.]educ1",
                family = poisson, data = yaish, subset = (dest != 7))

prof <- profile(cunidiff, trace = TRUE)
plot(prof)
```

### Clearly not quadratic for Mult1.Factor1.educ4 or Mult1.Factor1.educ5!

```r
confint(prof)
##
##                2.5 %    97.5 %
## Mult(Exp(.), orig:dest).educ1    NA     NA
## Mult(Exp(.), orig:dest).educ2   -0.5978901  0.1022447
## Mult(Exp(.), orig:dest).educ3  -1.4836854  -0.2362378
## Mult(Exp(.), orig:dest).educ4  -2.5792398  -0.2953420
## Mult(Exp(.), orig:dest).educ5     -Inf    0.7008889

coefData <- se(cunidiff)
```

```r
cbind(coefData[1] - 1.96 * coefData[2], coefData[1] + 1.96 * coefData[2])
```
### Multiplicative Approximation of Model Residuals

#### Description

This function uses the first $d$ components of the singular value decomposition in order to approximate a vector of model residuals by a sum of $d$ multiplicative terms, with the multiplicative structure determined by two specified factors. It applies to models of class `lm`, `glm` or `gnm`.

#### Usage

```r
cbind(coefData[1] - 1.96 * coefData[2], coefData[1] + 1.96 * coefData[2])```

#### Arguments

- `model`: an object of class `gnm`, `glm` or `lm`
- `fac1`: a factor
- `fac2`: a factor
- `d`: integer, the number of multiplicative terms to use in the approximation

#### Details

This function operates on the matrix of mean residuals, with rows indexed by `fac1` and columns indexed by `fac2`. For `glm` and `glm` models, the matrix entries are weighted working residuals. The primary use of `residsvd` is to generate good starting values for the parameters in `Mult` terms in models to be fitted using `gnm`.

#### Value

- If $d = 1$, a numeric vector; otherwise a numeric matrix with $d$ columns.
Description

Computes approximate standard errors for (a selection of) individual parameters or one or more linear combinations of the parameters in a `gnm` (generalized nonlinear model) object. By default, a check is made first on the estimability of each specified combination.

Usage

```r
se(model, estimate = NULL, checkEstimability = TRUE, Vcov = NULL, dispersion = NULL, ...)
```

Arguments

- `model` a model object of class `"gnm"`.
estimate (optional) specifies parameters or linear combinations of parameters for which to find standard errors. In the first case either a character vector of names, a numeric vector of indices or "[?]" to select from a Tk dialog. In the second case coefficients given as a vector or the rows of a matrix, such that NROW(estimate) is equal to length(coef(model)). If missing, standard errors are returned for all (non-eliminated) parameters in the model.

checkEstimability

logical: should the estimability of all specified combinations be checked?

Vcov

either NULL, or a matrix

dispersion

either NULL, or a positive number

... possible further arguments for checkEstimable.

Value

A data frame with two columns:

<table>
<thead>
<tr>
<th>Estimate</th>
<th>The estimated parameter combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std. Error</td>
<td>Their estimated standard errors</td>
</tr>
</tbody>
</table>

If available, the column names of coefMatrix will be used to name the rows.

Note

In the case where estimate is a numeric vector, se will assume that indices have been specified if all the values of estimate are in seq(length(coef(model))).

Where both Vcov and dispersion are supplied, the variance-covariance matrix of estimated model coefficients is taken to be Vcov * dispersion.

Author(s)

David Firth and Heather Turner

See Also

gnm, getContrasts, checkEstimable, ofInterest

Examples

set.seed(1)

## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
               Mult(Exp(educ), orig:dest),
               ofInterest = "[..]educ", family = poisson,
               data = yaish, subset = (dest != 7))

## Deviance is 200.3

## Get estimate and se for the contrast between educ4 and educ5 in the
## UNIDIFF multiplier
mycontrast <- numeric(length(coef(unidiff)))
mycontrast[ofInterest(unidiff)[4:5]] <- c(1, -1)
se(unidiff, mycontrast)

## Get all of the contrasts with educ5 in the UNIDIFF multipliers
getContrasts(unidiff, rev(ofInterest(unidiff)))

---

**summary.gnm**

*Summarize Generalized Nonlinear Model Fits*

**Description**

Summary method for objects of class "gnm"

**Usage**

## S3 method for class 'gnm'
summary(object, dispersion = NULL, correlation = FALSE,
         symbolic.cor = FALSE, with.eliminate = FALSE, ...)

## S3 method for class 'summary.gnm'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"),
      symbolic.cor = x$symbolic.cor, ...)

**Arguments**

- **object**: an object of class "gnm".
- **x**: an object of class "summary.gnm".
- **dispersion**: the dispersion parameter for the fitting family. By default it is obtained from object.
- **correlation**: logical: if TRUE, the correlation matrix of the estimated parameters is returned.
- **digits**: the number of significant digits to use when printing.
- **symbolic.cor**: logical: if TRUE, the correlations are printed in a symbolic form rather than numbers (see symnum).
- **signif.stars**: logical. If TRUE, "significance stars" are printed for each coefficient.
- **with.eliminate**: Logical. If TRUE, any eliminated coefficients are included in the summary.
- **...**: further arguments passed to or from other methods.
**Details**

`print.summary.gnm` prints the original call to `gnm`; a summary of the deviance residuals from the model fit; the coefficients of the model; the residual deviance; the Akaike’s Information Criterion value, and the number of main iterations performed.

Standard errors, z-values and p-values are printed alongside the coefficients, with "significance stars" if `signif.stars` is `TRUE`.

When the "summary.gnm" object has a "correlation" component, the lower triangle of this matrix is also printed, to two decimal places (or symbolically); to see the full matrix of correlations print `summary(object, correlation = TRUE)$correlation` directly.

The standard errors returned by `summary.gnm` are scaled by `sqrt(dispersion)`. If the dispersion is not specified, it is taken as 1 for the binomial and Poisson families, and otherwise estimated by the residual Chi-squared statistic divided by the residual degrees of freedom. For coefficients that have been constrained or are not estimable, the standard error is returned as `NA`.

**Value**

`summary.gnm` returns an object of class "summary.gnm", which is a list with components

- `call`: the "call" component from object.
- `ofInterest`: the "ofInterest" component from object.
- `family`: the "family" component from object.
- `deviance`: the "deviance" component from object.
- `aic`: the "aic" component from object.
- `df.residual`: the "df.residual" component from object.
- `iter`: the "iter" component from object.
- `deviance.resid`: the deviance residuals, see `residuals.glm`.
- `coefficients`: the matrix of coefficients, standard errors, z-values and p-values.
- `elim.coefs`: if `with.eliminate = TRUE` a matrix of eliminated coefficients, standard errors, z-values and p-values.
- `dispersion`: either the supplied argument or the estimated dispersion if the latter is `NULL`.
- `df`: a 3-vector of the rank of the model; the number of residual degrees of freedom, and number of unconstrained coefficients.
- `cov.scaled`: the estimated covariance matrix scaled by dispersion (see `vcov.gnm` for more details).
- `correlation`: (only if `correlation` is `TRUE`) the estimated correlations of the estimated coefficients.
- `symbolic.cor`: (only if `correlation` is `TRUE`) the value of the argument `symbolic.cor`.

**Note**

The `gnm` class includes generalized linear models, and it should be noted that `summary.gnm` differs from `summary.glm` in that it does not omit coefficients which are `NA` from the objects it returns. (Such coefficients are `NA` since they have been fixed at 0 either by use of the `constrain` argument to `gnm` or by a convention to handle linear aliasing).
**Author(s)**
Modifyiong of `summary.glm` by the R Core Team. Adapted for "gnm" objects by Heather Turner.

**See Also**
gnm, summary

**Examples**
```r
### First example from ?Dref
set.seed(1)

## reconstruct counts voting Labour/non-Labour
count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting$total - count)

## fit diagonal reference model with constant weights
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                     family = binomial, data = voting)

## summarize results - note diagonal weights are over-parameterised
summary(classMobility)

## refit setting first weight to zero (as DrefWeights() does)
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                     family = binomial, data = voting,
                     constrain = "delta1")
summary(classMobility)
```

---

**Symm**

**Symmetric Interaction of Factors**

**Description**
Symm codes the symmetric interaction of factors having the same set of levels, for use in regression models of symmetry or quasi-symmetry.

**Usage**

`Symm(...)`

**Arguments**

`...` one or more factors.

**Value**
A factor whose levels index the symmetric interaction of all factors supplied as input.
termPredictors

Description

termPredictors is a generic function which extracts the contribution of each term to the predictor from a fitted model object.

Usage

termPredictors(object, ...)

Arguments

  object          a fitted model object.
  ...             additional arguments for method functions.

Details

The default method assumes that the predictor is linear and calculates the contribution of each term from the model matrix and fitted coefficients. A method is also available for gnm objects.

Value

A matrix with the additive components of the predictor in labelled columns.

Author(s)

Heather Turner
Topo  

**Topological Interaction of Factors**

### Description

Given two or more factors Topo creates an interaction factor as specified by an array of levels, which may be arbitrarily structured.

### Usage

`Topo(...)`, `spec = NULL`)

### Arguments

- ...  
  - two or more factors

- spec  
  - an array of levels, with dimensions corresponding to the number of levels of each factor in the interaction
Value

A factor of levels extracted from the levels array given in spec, using the given factors as index variables.

Author(s)

David Firth

References


See Also

`symm` and `diag` for special cases

Examples

```r
set.seed(1)

### Collapse to 7 by 7 table as in Erikson (1982)

erikson <- as.data.frame(erikson)
levl <- levels(erikson$origin)
levels(erikson$origin) <- levels(erikson$destination) <-
c(rep(paste(levl[1:2], collapse = " + "), 2), levl[3],
 rep(paste(levl[4:5], collapse = " + "), 2), levl[6:9])
erikson <- xtabs(Freq ~ origin * destination + country, data = erikson)

### Create array of interaction levels as in Table 2 of Xie (1992)

levelMatrix <- matrix(c(2, 3, 4, 6, 5, 6, 6,
                        3, 4, 6, 4, 5, 6,
                        4, 2, 5, 5, 5, 5,
                        6, 6, 5, 1, 6, 5, 2,
                        4, 4, 5, 6, 3, 4, 5,
                        5, 4, 5, 5, 3, 3, 5,
                        6, 6, 5, 3, 5, 4, 1), 7, 7, byrow = TRUE)

### Fit the levels models given in Table 3 of Xie (1992)

# Null association between origin and destination
nullModel <- glm(Freq ~ country:origin + country:destination,
                 family = poisson, data = erikson)

# Interaction specified by levelMatrix, common to all countries
commonTopo <- update(nullModel, ~ . +
                      Topo(origin, destination, spec = levelMatrix))
```
### Interaction specified by levelMatrix, different multiplier for each country

```r
multTopo <- update(nullModel, ~ . +
                    Mult(Exp(country),
                    Topo(origin, destination, spec = levelMatrix)))
```

### Interaction specified by levelMatrix, different effects for each country

```r
separateTopo <- update(nullModel, ~ . +
                        country:Topo(origin, destination,
                        spec = levelMatrix))
```

---

**vcov.gnm**

**Variance-covariance Matrix for Parameters in a Generalized Nonlinear Model**

### Description

This method extracts or computes a variance-covariance matrix for use in approximate inference on estimable parameter combinations in a generalized nonlinear model.

### Usage

```r
## S3 method for class 'gnm'
vcov(object, dispersion = NULL, with.eliminate = FALSE, ...)
```

### Arguments

- `object`: a model object of class `gnm`.
- `dispersion`: the dispersion parameter for the fitting family. By default it is obtained from `object`.
- `with.eliminate`: logical; should parts of the variance-covariance matrix corresponding to eliminated coefficients be computed?
- `...`: as for `vcov`.

### Details

The resultant matrix does not itself necessarily contain variances and covariances, since `gnm` typically works with over-parameterized model representations in which parameters are not all identified. Rather, the resultant matrix is to be used as the kernel of quadratic forms which are the variances or covariances for estimable parameter combinations.

The matrix values are scaled by `dispersion`. If the dispersion is not specified, it is taken as 1 for the binomial and Poisson families, and otherwise estimated by the residual Chi-squared statistic divided by the residual degrees of freedom. The dispersion used is returned as an attribute of the matrix.
The dimensions of the matrix correspond to the non-eliminated coefficients of the "gnm" object. If `use.eliminate = TRUE` then setting can sometimes give appreciable speed gains; see `gnm` for details of the `eliminate` mechanism. The `use.eliminate` argument is currently ignored if the model has full rank.

**Value**

A matrix with number of rows/columns equal to `length(coef(object))`. If there are eliminated coefficients and `use.eliminate = TRUE`, the matrix will have the following attributes:

- `covElim`: a matrix of covariances between the eliminated and non-eliminated parameters.
- `varElim`: a vector of variances corresponding to the eliminated parameters.

**Note**

The `gnm` class includes generalized linear models, and it should be noted that the behaviour of `vcov.gnm` differs from that of `vcov.glm` whenever any(is.na(coef(object))) is TRUE. Whereas `vcov.glm` drops all rows and columns which correspond to NA values in `coef(object)`, `vcov.gnm` keeps those columns (which are full of zeros, since the NA represents a parameter which is fixed either by use of the `constrain` argument to `gnm` or by a convention to handle linear aliasing).

**Author(s)**

David Firth

**References**


**See Also**

`getContrasts`, `se`

**Examples**

```r
set.seed(1)
## Fit the "UNIDIFF" mobility model across education levels
unidiff <- gnm(Freq ~ educ*orig + educ*dest +
              Mult(Exp(educ), orig:dest), family = poisson,
              data = yaish, subset = (dest != ?))
## Examine the education multipliers (differences on the log scale):
ind <- pickCoef(unidiff, "[].educ")
educMultipliers <- getContrasts(unidiff, rev(ind))
## Now get the same standard errors using a suitable set of
## quadratic forms, by calling `vcov()` directly:
cmat <- contr.sum(ind)
sterrs <- sqrt(diag(t(cmat))
  %-*% vcov(unidiff)[ind, ind]
  %-*% cmat))
all(sterrs == (educMultipliers$SE)[-1])  ## TRUE
```
**Data on Social Mobility and the Labour Vote**

**Description**

Voting data from the 1987 British general election, cross-classified by the class of the head of household and the class of their father.

**Usage**

voting

**Format**

A data frame with 25 observations on the following 4 variables.

- percentage  the percentage of the cell voting Labour.
- total  the cell count.
- origin  a factor describing the father’s class with levels Q:U.
- destination  a factor describing the head of household’s class with levels Q:U.

**Source**


**Examples**

```r
## Examples from Clifford and Heath paper
## (Results differ slightly - possible transcription error in
## published data?)
set.seed(1)

count <- with(voting, percentage/100 * total)
yvar <- cbind(count, voting$total - count)

## fit diagonal reference model with constant weights
classMobility <- gnm(yvar ~ -1 + Dref(origin, destination),
                     family = binomial, data = voting)
DrefWeights(classMobility)

## create factors indicating movement in and out of salariat (class 1)
upward <- with(voting, origin != 1 & destination == 1)
downward <- with(voting, origin == 1 & destination != 1)

## fit separate weights for the "socially mobile" groups
socialMobility <- gnm(yvar ~ -1 + Dref(origin, destination,}
```
Description

Creates a family object for use with glm, gnm, etc., for the variance function \( \mu(1 - \mu)^2 \) introduced by Wedderburn (1974) for response values in [0,1].

Usage

wedderburn(link = "logit")

Arguments

- link: The name of a link function. Allowed are "logit", "probit" and "cloglog".

Value

An object of class family.

Note

The reported deviance involves an arbitrary constant (see McCullagh and Nelder, 1989, p330); for estimating dispersion, use the Pearson chi-squared statistic instead.

Author(s)

Modification of binomial by the R Core Team. Adapted for the Wedderburn quasi-likelihood family by David Firth.

References

See Also
glm, gnm, family

Examples

```r
set.seed(1)
### Use data from Wedderburn (1974), see ?barley

### Fit Wedderburn's logit model with variance proportional to the
### square of mu(1-mu)
logitModel <- glm(y ~ site + variety, family = wedderburn, data = barley)
fit <- fitted(logitModel)
print(sum((barley$y - fit)^2 / (fit * (1-fit))^2))
# Agrees with the chi-squared value reported in McCullagh and Nelder
# (1989, p331), which differs slightly from Wedderburn's reported value.

### Fit the biplot model as in Gabriel (1998, p694)
bipplotModel <- gnm(y ~ -1 + instances(Mult(site, variety), 2),
                   family = wedderburn, data = barley)
barleySVD <- svd(matrix(bipplotModel$predictors, 10, 9))
A <- sweep(barleySVD$v, 2, sqrt(barleySVD$d), "*")[, 1:2]
B <- sweep(barleySVD$u, 2, sqrt(barleySVD$d), "*")[, 1:2]
# These are essentially A and B as in Gabriel (1998, p694), from which
# the biplot is made by
plot(rbind(A, B), pch = c(LETTERS[1:9], as.character(1:9), "X"))

### Fit the double-additive model as in Gabriel (1998, p697)
variety.binary <- factor(match(barley$variety, c(2,3,6), nomatch = 0),
                         labels = c("Rest", "2,3,6"))
doubleAdditive <- gnm(y ~ variety + Mult(site, variety.binary),
                      family = wedderburn, data = barley)
```
wheat

Format

A data frame with 240 observations on the following 33 variables.

yield  numeric, mean yield in kg/ha for 3 replicates
year   a factor with levels 1988:1997
tillage a factor with levels T t
summerCrop a factor with levels S s
manure  a factor with levels M m
N      a factor with levels 0 N n
MTD    numeric, mean max temp sheltered (deg C) in December
MTJ    same for January
MTF    same for February
MTM    same for March
MTA    same for April
mTD    numeric, mean min temp sheltered (deg C) in December
mTJ    same for January
mTF    same for February
mTM    same for March
mTA    same for April
mTUD   numeric, mean min temp unsheltered (deg C) in December
mTUJ   same for January
mTUF   same for February
mTUM   same for March
mTUA   same for April
PRD    numeric, total precipitation (mm) in December
PRJ    same for January
PRF    same for February
PRM    same for March
SHD    numeric, mean sun hours in December
SHJ    same for January
SHF    same for February
EVD    numeric, total evaporation (mm) in December
EVJ    same for January
EVF    same for February
EVM    same for March
EVA    same for April
Source


Examples

```r
set.seed(1)

## Scale yields to reproduce analyses reported in Vargas et al (2001)
yield_scaled <- wheat$yield * sqrt(3/1000)

## Reproduce (up to error caused by rounding) Table 1 of Vargas et al (2001)
aov(yield_scaled ~ year*tillage*summerCrop*manure*N, data = wheat)
treatment <- interaction(wheat$tillage, wheat$summerCrop, wheat$manure, wheat$N, sep = "")
mainEffects <- lm(yield_scaled ~ year + treatment, data = wheat)
svdStart <- residSVD(mainEffects, year, treatment, 3)
bilinear1 <- update(asGnm(mainEffects), . ~ . + Mult(year, treatment),
                      start = c(coef(mainEffects), svdStart[,1]))
bilinear2 <- update(bilinear1, . ~ . + Mult(year, treatment, inst = 2),
                      start = c(coef(bilinear1), svdStart[,2]))
bilinear3 <- update(bilinear2, . ~ . + Mult(year, treatment, inst = 3),
                      start = c(coef(bilinear2), svdStart[,3]))
anova(mainEffects, bilinear1, bilinear2, bilinear3)

## Examine the extent to which, say, mTF explains the first bilinear term
bilinear1mTF <- gnm(yield_scaled ~ year + treatment + Mult(1 + mTF, treatment),
                     family = gaussian, data = wheat)
anova(mainEffects, bilinear1mTF, bilinear1)

## How to get the standard SVD representation of an AMMI-n model
## We'll work with the AMMI-2 model, which here is called “bilinear2”
## First, extract the contributions of the 5 terms in the model:
## wheat.terms <- termPredictors(bilinear2)
## That's a matrix, whose 4th and 5th columns are the interaction terms
## Combine those two interaction terms, to get the total estimated
## interaction effect:
## wheat.interaction <- wheat.terms[, 4] + wheat.terms[, 5]
## That's a vector, so we need to re-shape it as a 24 by 10 matrix
## ready for calculating the SVD:
```
wheat.interaction <- matrix(wheat.interaction, 24, 10)
##
## Now we can compute the SVD:
##
## wheat.interaction.SVD <- svd(wheat.interaction)
##
## Only the first two singular values are nonzero, as expected
## (since this is an AMMI-2 model, the interaction has rank 2)
##
## So the result object can be simplified by re-calculating the SVD with
## just two dimensions:
##
## wheat.interaction.SVD <- svd(wheat.interaction, nu = 2, nv = 2)

---

### Class Mobility by Level of Education in Israel

**Description**

A 3-way contingency table of father/son pairs, classified by father's social class (orig), son's social class (dest) and son’s education level (educ).

**Usage**

yaish

**Format**

A table of counts, with classifying factors educ (levels 1:5), orig (levels 1:7) and dest (levels 1:7).

**Author(s)**

David Firth

**Source**


**References**


Examples

set.seed(1)

## Fit the "UNIDIFF" mobility model across education levels, leaving out
## the uninformative subtable for dest == 7:
##
## unidiff <- glm(Freq ~ educ*orig + educ*dest +
##     Mult(Exp(educ), orig:dest), family = poisson,
##     data = yaish, subset = (dest != 7))
##
## Deviance should be 200.3, 116 d.f.
##
## Look at the multipliers of the orig:dest association:
ofInterest(unidiff) <- pickCoeff(unidiff, "[.educ")
coef(unidiff)

## Coefficients of interest:
##
## Mult(Exp(.), orig:dest).educ1 Mult(Exp(.), orig:dest).educ2
##  -0.5513258  -0.7766976
##  -1.2947494  -1.5902644
## Mult(Exp(.), orig:dest).educ5
##  -2.8000285

## Get standard errors for the contrasts with educ1:
##
## getContrasts(unidiff, ofInterest(unidiff))
##
## getContrasts(unidiff, ofInterest(unidiff))
##
## Mult(Exp(.), orig:dest).educ1 0.0000000 0.0000000 0.09757438
## Mult(Exp(.), orig:dest).educ2 -0.2253718 0.1611874 0.12885047
## Mult(Exp(.), orig:dest).educ3 -0.7434236 0.2335083 0.21182123
## Mult(Exp(.), orig:dest).educ4 -1.0389386 0.3434256 0.32609380
## Mult(Exp(.), orig:dest).educ5 -2.2495026 0.9453764 0.95506643
##
## Table of model residuals:
##
## residuals(unidiff)
Index

* Topic **array**
  MPInv, 45

* Topic **datasets**
  backPain, 6
  barley, 8
  barleyHeights, 10
  cautres, 11
  erikson, 22
  friend, 27
  House2001, 36
  mentalHealth, 43
  voting, 76
  wheat, 78
  yaish, 81

* Topic **hplot**
  plot.gnm, 57

* Topic **manip**
  expandCategorical, 25

* Topic **models**
  anova.gnm, 4
  asGnm, 5
  checkEstimable, 12
  confint.gnm, 13
  Const, 15
  Diag, 16
  Dref, 17
  exitInfo, 23
  Exp, 24
  expandCategorical, 25
  getContrasts, 28
  gnm, 30
  gnm-package, 3
  instances, 39
  Inv, 40
  meanResiduals, 41
  model.matrix.gnm, 44
  MultHomog, 46
  Multiplicative interaction, 48
  nonlin.function, 50
  ofInterest, 53
  parameters, 55
  pickCoeff, 56
  plot.gnm, 57
  predict.gnm, 60
  profile.gnm, 62
  residSVD, 65
  se, 66
  summary.gnm, 68
  Symm, 70
  termPredictors, 71
  Topo, 72
  vcov.gnm, 74
  wedderburn, 77

* Topic **nonlinear**
  checkEstimable, 12
  confint.gnm, 13
  Const, 15
  Dref, 17
  exitInfo, 23
  Exp, 24
  getContrasts, 28
  gnm, 30
  gnm-package, 3
  instances, 39
  Inv, 40
  meanResiduals, 41
  model.matrix.gnm, 44
  MultHomog, 46
  Multiplicative interaction, 48
  nonlin.function, 50
  plot.gnm, 57
  predict.gnm, 60
  profile.gnm, 62
  residSVD, 65
  se, 66
  summary.gnm, 68
  vcov.gnm, 74

* Topic **package**
gnm-package, 3

Topic regression

asGnm, 5
checkEstimable, 12
Const, 15
dref, 17
exitInfo, 23
Exp, 24
getContrasts, 28
gnm, 30
gnm-package, 3
instances, 39
Inv, 40
meanResiduals, 41
model.matrix.gnm, 44
MultHomog, 46
Multiplicative interaction, 48
nonlin.function, 50
plot.gnm, 57
residSVD, 65
se, 66
summary.gnm, 68
termPredictors, 71
vcov.gnm, 74

anova, 5
anova(glm, 5
anova.gnm, 4
asGnm, 5

backPain, 6, 32
barley, 8
barleyHeights, 10
binomial, 77

cautres, 11
checkEstimable, 12, 30, 67
coeff, 35
coefficients, 35, 55
confint.glm, 14
confint.gnm, 13, 54, 63
confint.profile.gnm(confint.gnm), 13
Const, 15, 51

deviance, 35
df.residual, 35
Diag, 16, 35, 71, 73
dref, 17, 35, 51
DrefWeights (Dref), 17

erikson, 22
exitInfo, 23, 33–35
Exp, 24, 51
expandCategorical, 25
extractAIC, 35
family, 31, 35, 77, 78
fitted, 35
fitted.values, 35
formula, 16, 19, 25, 32, 35, 40, 41, 47, 49
friend, 27

getContrasts, 13, 28, 33, 35, 54, 67, 75
ginv, 46
glm, 6, 77, 78
gnm, 3, 5, 6, 12–14, 16, 17, 19, 24, 25, 27, 28,
   30, 30, 35, 39–41, 45–49, 54, 55, 59,
   61, 63, 65–67, 70–72, 75, 77, 78
gnm-package, 3
grep, 54, 56

hat, 59
House2001, 36

instances, 39, 47, 49, 50
Inv, 40, 51

lm, 6

mclgen, 27
meanResiduals, 41
mentalHealth, 43
model.frame, 35
model.matrix, 35, 45
model.matrix.gnm, 44
Mpinv, 45
Mult, 40, 47, 51, 65, 66
Mult (Multiplicative interaction), 48
MultHomog, 35, 40, 46, 49, 51

Multinom, 27

Multiplicative interaction, 48

napredict, 60
nonlin.function, 19, 25, 32, 35, 40, 41, 47, 49, 50

offset, 16

ofInterest, 30, 33, 53, 56, 63, 67

ofInterest <- (ofInterest), 53

parameters, 55
INDEX

pickCoef, 56
pickFrom, 56
plot.gnm, 35, 57
plot.lm, 59
plot.profile.gnm(profile.gnm), 62
predict.gnm, 60
print.summary.gnm(summary.gnm), 68
profile.glm, 14, 63
profile.gnm, 14, 54, 62

qvcalc, 29, 30

rankMatrix, 12
regexp, 56
reshape, 27
residSVD, 65
residuals, 35
residuals.glm, 35, 69

se, 13, 30, 54, 66, 75
stat.anova, 4
summary, 70
summary.glm, 69, 70
summary.gnm, 35, 68
Symm, 17, 35, 70, 73

termPredictors, 35, 71
terms, 35
topo, 72

update, 33

vcov, 35, 74
vcov.glm, 75
vcov.gnm, 69, 74
voting, 76

wedderburn, 31, 77
weights, 35
wheat, 78

yaish, 81