Generalized Nonlinear Models using the gnm Package

Heather Turner

Independent statistical/R consultant
Department of Statistics, University of Warwick, UK

Toulouse, 2019–07–09

Preface

Generalized linear models (logit/probit regression, log-linear models, etc.) are now part of the standard empirical toolkit.

Sometimes the assumption of a *linear* predictor is unduly restrictive.

This short course shows how *generalized nonlinear models* may be viewed as a unified class, and how to work with such models in R.

Part I: Introduction to Generalized Nonlinear Models in R

Background theory

Structured interactions

Introduction to the gnm package

Practical I

Part II: Further Examples of Generalized Nonlinear Models

RC Model with Heterogeneous Scores

Further Multiplicative Models

Diagonal Reference Model

Custom Nonlinear Terms

Practical II

Part I

Overview of Generalized Nonlinear Models in R

Linear models:

e.g.,

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 z_i$$

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

$$E(y_i) = \beta_0 + \gamma_1 \delta_1 x_i + \exp(\theta_2) z_i$$

In general:

$$E(y_i) = \eta_i(\beta) = \text{linear function of unknown parameters}$$

Also assumes variance essentially constant:

$$\text{var}(y_i) = \phi a_i$$

with a_i known (often $a_i \equiv 1$).

Linear models:

e.g.,

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 z_i$$

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

$$E(y_i) = \beta_0 + \gamma_1 \delta_1 x_i + \exp(\theta_2) z_i$$

In general:

$$E(y_i) = \eta_i(\beta) = \text{linear function of unknown parameters}$$

Also assumes variance essentially constant:

$$\text{var}(y_i) = \phi a_i$$

with a_i known (often $a_i \equiv 1$).

Linear models:

e.g.,

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 z_i$$

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

$$E(y_i) = \beta_0 + \gamma_1 \delta_1 x_i + \exp(\theta_2) z_i$$

In general:

$$E(y_i) = \eta_i(\beta) = \text{linear function of unknown parameters}$$

Also assumes variance essentially constant:

$$\text{var}(y_i) = \phi a_i$$

with a_i known (often $a_i \equiv 1$).

Linear models:

e.g.,

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 z_i$$

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

$$E(y_i) = \beta_0 + \gamma_1 \delta_1 x_i + \exp(\theta_2) z_i$$

In general:

$$E(y_i) = \eta_i(\beta) = \text{linear function of unknown parameters}$$

Also assumes variance essentially constant:

$$\text{var}(y_i) = \phi a_i$$

with a_i known (often $a_i \equiv 1$).

Generalized linear models

Problems with linear models in many applications:

- range of y is restricted (e.g., y is a count, or is binary, or is a duration)
- effects are not additive
- variance depends on mean (e.g., large mean $\Rightarrow$ large variance)

Generalized linear models specify a non-linear *link function* and *variance function* to allow for such things, while maintaining the simple interpretation of linear models.

Generalized linear models

Problems with linear models in many applications:

- range of y is restricted (e.g., y is a count, or is binary, or is a duration)
- effects are not additive
- variance depends on mean (e.g., large mean $\Rightarrow$ large variance)

Generalized linear models specify a non-linear *link function* and *variance function* to allow for such things, while maintaining the simple interpretation of linear models.

Generalized linear model:

$$g[E(y_i)] = \eta_i = \text{linear function of unknown parameters}$$
$$\text{var}(y_i) = \phi a_i V(\mu_i)$$

with the functions g (link function) and V (variance function) known.

Examples:

- binary logistic regressions
- rate models for event counts
- log-linear models for contingency tables (including multinomial logit models)
- multiplicative models for durations and other positive measurements
- hazard models for event history data

etc., etc.

e.g., binary logistic regression:

$$y_i = \begin{cases} 1 & \text{event happens} \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_i = E(y_i) = \text{probability that event happens}$$

$$\text{var}(y_i) = \mu_i(1 - \mu_i)$$

Variance is completely determined by mean.

Common link functions are logit, probit, and (complementary) log-log, all of which transform constrained μ into unconstrained η.

e.g., binary logistic regression:

$$y_i = \begin{cases} 1 & \text{event happens} \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_i = E(y_i) = \text{probability that event happens}$$

$$\text{var}(y_i) = \mu_i(1 - \mu_i)$$

Variance is completely determined by mean.

Common link functions are logit, probit, and (complementary) log-log, all of which transform constrained μ into unconstrained η.

e.g., binary logistic regression:

$$y_i = \begin{cases} 1 & \text{event happens} \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_i = E(y_i) = \text{probability that event happens}$$

$$\text{var}(y_i) = \mu_i(1 - \mu_i)$$

Variance is completely determined by mean.

Common link functions are logit, probit, and (complementary) log-log, all of which transform constrained μ into unconstrained η.

e.g., multiplicative (i.e., log-linear) rate model for event counts.

'Exposure' for observation i is a fixed, known quantity t_i.

Rate model:

$$E(y_i) = t_i \exp(\beta_0) \exp(\beta_1 x_i) \exp(\beta_2 z_i)$$

i.e.,

$$\log E(y_i) = \log t_i + \beta_0 + \beta_1 x_i + \beta_2 z_i$$

— effects are rate multipliers.

Variance is typically taken as the Poisson-like function $V(\mu) = \mu$ (variance is equal to, or is proportional to, the mean).

e.g., multiplicative (i.e., log-linear) rate model for event counts.

'Exposure' for observation i is a fixed, known quantity t_i.

Rate model:

$$E(y_i) = t_i \exp(\beta_0) \exp(\beta_1 x_i) \exp(\beta_2 z_i)$$

i.e.,

$$\log E(y_i) = \log t_i + \beta_0 + \beta_1 x_i + \beta_2 z_i$$

— effects are rate multipliers.

Variance is typically taken as the Poisson-like function $V(\mu) = \mu$ (variance is equal to, or is proportional to, the mean).

e.g., multiplicative (i.e., log-linear) rate model for event counts.

'Exposure' for observation i is a fixed, known quantity t_i.

Rate model:

$$E(y_i) = t_i \exp(\beta_0) \exp(\beta_1 x_i) \exp(\beta_2 z_i)$$

i.e.,

$$\log E(y_i) = \log t_i + \beta_0 + \beta_1 x_i + \beta_2 z_i$$

— effects are rate multipliers.

Variance is typically taken as the Poisson-like function $V(\mu) = \mu$ (variance is equal to, or is proportional to, the mean).

Generalized linear: $\eta = g(\mu)$ is a linear function of the unknown parameters. Variance depends on mean through $V(\mu)$.

Generalized *nonlinear*: still have g and V, but now relax the linearity assumption.

Many important aspects remain unchanged:

- fitting by maximum likelihood or quasi-likelihood
- analysis of deviance to assess significance of effects
- diagnostics based on residuals, etc.

But technically more difficult [essentially because $\partial\eta/\partial\beta = X$ becomes $\partial\eta/\partial\beta = X(\beta)$].

Generalized linear: $\eta = g(\mu)$ is a linear function of the unknown parameters. Variance depends on mean through $V(\mu)$.

Generalized *nonlinear*: still have g and V, but now relax the linearity assumption.

Many important aspects remain unchanged:

- fitting by maximum likelihood or quasi-likelihood
- analysis of deviance to assess significance of effects
- diagnostics based on residuals, etc.

But technically more difficult [essentially because $\partial\eta/\partial\beta = X$ becomes $\partial\eta/\partial\beta = X(\beta)$].

Some practical consequences of the technical difficulties:

- automatic detection and elimination of redundant parameters is very difficult — it's no longer just a matter of linear algebra
- automatic generation of good starting values for ML fitting algorithms is hard
- great care is needed in cases where the likelihood has more than one maximum (which cannot happen in the linear case).

Some motivation: structured interactions

GNMs are not exclusively about structured interactions, but many applications are of this kind.

A classic example is log-linear models for structurally-square contingency tables (e.g., pair studies, before-after studies, etc.).

Pairs are classified twice, into row and column of a table of counts.

The independence model is

$$\log E(y_{rc}) = \theta + \beta_r + \gamma_c$$

or with **glm**

```
glm(y ~ row + col, family = poisson)
```

Some standard (generalized linear) models for departure from independence are

- quasi-independence,

```
y ~ row + col + Diag(row, col)
```

- quasi-symmetry,

```
y ~ row + col + Symm(row, col)
```

- symmetry,

```
y ~ Symm(row, col)
```

Functions `Diag` and `Symm` are provided by the **gnm** package along with the function `Topo` for fully-specified *topological* association structures, see `?Topo`.

Row-column association

The uniform association model (for ordered categories) has

$$\log E(y_{rc}) = \beta_r + \gamma_c + \delta u_r v_c$$

with the u_r and v_c defined as fixed, equally-spaced scores for the rows and columns.

The row-column association (RC) model lets the *data* determine the scores (Goodman, 1979). This can be done either heterogeneously,

$$\log E(y_{rc}) = \beta_r + \gamma_c + \phi_r \psi_c$$

or (in the case of a structurally square table) homogeneously,

$$\log E(y_{rc}) = \beta_r + \gamma_c + \phi_r \phi_c$$

These are generalized non-linear models.

Row-column association

The uniform association model (for ordered categories) has

$$\log E(y_{rc}) = \beta_r + \gamma_c + \delta u_r v_c$$

with the u_r and v_c defined as fixed, equally-spaced scores for the rows and columns.

The row-column association (RC) model lets the *data* determine the scores (Goodman, 1979). This can be done either heterogeneously,

$$\log E(y_{rc}) = \beta_r + \gamma_c + \phi_r \psi_c$$

or (in the case of a structurally square table) homogeneously,

$$\log E(y_{rc}) = \beta_r + \gamma_c + \phi_r \phi_c$$

These are generalized non-linear models.

Introduction to the **gnm** package

The **gnm** package aims to provide a unified computing framework for specifying, fitting and criticizing generalized nonlinear models in R.

The central function is `gnm`, which is designed with the same interface as `glm`.

Since generalized linear models are included as a special case, the `gnm` function can be used in place of `glm`, and will give equivalent results.

For the special case $g(\mu) = \mu$ and $V(\mu) = 1$, the `gnm` fit is equivalent to an `nls` fit.

Nonlinear model terms

Nonlinear model terms are specified in model formulae using functions of class `"nonlin"`.

These functions specify the term structure, possibly also labels and starting values.

There are a number of `"nonlin"` functions provided by **gnm**. Some of these specify basic mathematical functions of predictors, e.g. `Mult` and `Exp`.

So heterogeneous row and column scores

$$\phi_r \psi_c$$

are specified as `Mult(row, col)`.

Specialized "nonlin" functions

There are two specialized "nonlin" functions provided by **gnm**

`MultHomog`: for homogeneous row and column scores, as in

$$\phi_r \phi_c$$

specified as `MultHomog(row, col)`

`Dref`: 'diagonal reference' dependence on a square classification,

$$w_1 \gamma_r + w_2 \gamma_c$$

(Sobel, 1981, 1985) specified as `Dref(row, col)`

Any (differentiable) nonlinear term can be specified by nesting existing "nonlin" functions or writing a custom "nonlin" function.

Specialized "nonlin" functions

There are two specialized "nonlin" functions provided by **gnm**

`MultHomog`: for homogeneous row and column scores, as in

$$\phi_r \phi_c$$

specified as `MultHomog(row, col)`

`Dref`: 'diagonal reference' dependence on a square classification,

$$w_1 \gamma_r + w_2 \gamma_c$$

(Sobel, 1981, 1985) specified as `Dref(row, col)`

Any (differentiable) nonlinear term can be specified by nesting existing "nonlin" functions or writing a custom "nonlin" function.

Specialized "nonlin" functions

There are two specialized "nonlin" functions provided by **gnm**

MultHomog: for homogeneous row and column scores, as in

$$\phi_r \phi_c$$

specified as `MultHomog(row, col)`

Dref: 'diagonal reference' dependence on a square classification,

$$w_1 \gamma_r + w_2 \gamma_c$$

(Sobel, 1981, 1985) specified as `Dref(row, col)`

Any (differentiable) nonlinear term can be specified by nesting existing "nonlin" functions or writing a custom "nonlin" function.

Specialized "nonlin" functions

There are two specialized "nonlin" functions provided by **gnm**

MultHomog: for homogeneous row and column scores, as in

$$\phi_r \phi_c$$

specified as `MultHomog(row, col)`

Dref: 'diagonal reference' dependence on a square classification,

$$w_1 \gamma_r + w_2 \gamma_c$$

(Sobel, 1981, 1985) specified as `Dref(row, col)`

Any (differentiable) nonlinear term can be specified by nesting existing "nonlin" functions or writing a custom "nonlin" function.

Over-parameterization

The gnm function makes no attempt to remove redundant parameters from nonlinear terms. This is deliberate.

As a consequence, fitted models are typically represented in a way that is *over-parameterized*: not all of the parameters are 'estimable' (i.e., 'identifiable', 'interpretable').

For example, for the RC model with homogeneous scores

$$\begin{aligned}\alpha_r + \beta_c + \phi_r\phi_c &= -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k) \\ &= \alpha_r^* + \beta_c^* + \phi_r^*\phi_c^*\end{aligned}$$

gnm will return one of the infinitely many parameterizations at random.

Over-parameterization

The gnm function makes no attempt to remove redundant parameters from nonlinear terms. This is deliberate.

As a consequence, fitted models are typically represented in a way that is *over-parameterized*: not all of the parameters are 'estimable' (i.e., 'identifiable', 'interpretable').

For example, for the RC model with homogeneous scores

$$\begin{aligned}\alpha_r + \beta_c + \phi_r\phi_c &= -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k) \\ &= \alpha^*_r + \beta^*_c + \phi^*_r\phi^*_c\end{aligned}$$

gnm will return one of the infinitely many parameterizations at random.

Over-parameterization

The gnm function makes no attempt to remove redundant parameters from nonlinear terms. This is deliberate.

As a consequence, fitted models are typically represented in a way that is *over-parameterized*: not all of the parameters are 'estimable' (i.e., 'identifiable', 'interpretable').

For example, for the RC model with homogeneous scores

$$\begin{aligned}\alpha_r + \beta_c + \phi_r\phi_c &= -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k) \\ &= \alpha^*_r + \beta^*_c + \phi^*_r\phi^*_c\end{aligned}$$

gnm will return one of the infinitely many parameterizations at random.

Practical I

1. Load the **gnm** package. This provides the `occupationalStatus` data set, which is a contingency table classified by the occupational status of fathers (`origin`) and their sons (`destination`).

2. Use the generic function `plot` to create a mosaic plot of the table. Print `occupationalStatus` to see the cell frequencies represented by the plot.

3. If a table is passed to the `data` argument of `gnm`, it will be converted to a data frame with a column for each of the row and column factors and a column for the frequencies named `Freq`.

Use `gnm` to fit an independence model to these data (see p14), assigning the result to a suitable name. Print this object.

4. Type `?plot.gnm` to open the help page on the `gnm` method for the `plot` function. Find out how to use `plot` to create a plot of residuals vs. fitted values and do this for the independence model. The poor fit should be very apparent!

5. Load the **vcdExtra** package. This provides the generic function `mosaic`, which has a method for `"gnm"` objects. Use this to visualize the goodness-of-fit of the independence model across the contingency table.

6. Fit a row-column association model with a homogeneous multiplicative interaction between origin and destination (see p16, p19). Check the fit with `mosaic`. Investigate the effect of modelling the diagonal elements separately, by adding `Diag(origin, destination)`.

7. Keeping the `Diag` term in the model, use `coef` to access the coefficients and assign the result. Re-run the model fit and assign the coefficients of the re-fitted model to another name. Compare the coefficients side-by-side using `cbind`. Which parameters have been automatically constrained to zero? Which coefficients are the same in both models?

8. Standard errors can only be obtained for estimable parameters. Use `summary` to confirm which parameters are estimable in the current model. The homogeneous scores can be identified by setting one of them to zero. Re-fit the model using the argument `constrain = "MultHomog(origin, destination)1"`. Compare the summary of the constrained model to the summary of the unconstrained model.

Part II

Further Examples of Generalized Nonlinear Models

RC Model with Heterogeneous Scores

In this case, for example,

$$\begin{aligned}\alpha_r + \beta_c + \phi_r\psi_c &= \alpha_r + (\beta_r - \psi_c) + (\phi_r + 1)\psi_c \\ &= \alpha_r + \beta_c + (2\phi_r)(\psi_c/2)\end{aligned}$$

so we need to constrain both the location and scale.

A standard convention is to constrain the scores so that

$$\sum_r \phi_r \pi_r = \sum_c \psi_c \pi_c = 0$$

$$\text{and } \sum_r \phi_r^2 \pi_r = \sum_c \psi_c^2 \pi_c = 1$$

where π_r and π_c are the row and column probabilities respectively. The full interaction is then given by $\sigma\phi_r\psi_c$, where $\sigma > 0$ is the *intrinsic association parameter*.

RC Model with Heterogeneous Scores

In this case, for example,

$$\begin{aligned}\alpha_r + \beta_c + \phi_r\psi_c &= \alpha_r + (\beta_r - \psi_c) + (\phi_r + 1)\psi_c \\ &= \alpha_r + \beta_c + (2\phi_r)(\psi_c/2)\end{aligned}$$

so we need to constrain both the location and scale.

A standard convention is to constrain the scores so that

$$\sum_r \phi_r\pi_r = \sum_c \psi_c\pi_c = 0$$

$$\text{and } \sum_r \phi_r^2\pi_r = \sum_c \psi_c^2\pi_c = 1$$

where π_r and π_c are the row and column probabilities respectively. The full interaction is then given by $\sigma\phi_r\psi_c$, where $\sigma > 0$ is the *intrinsic association parameter*.

Example: Mental Health Data

1660 residents of Manhattan cross-classified by child's mental impairment and parents' socioeconomic status (Agresti, 2013).

```
xtabs(count ~ SES + MHS, mentalHealth)
```

```
##     MHS
## SES well mild moderate impaired
##   A   64   94       58       46
##   B   57   94       54       40
##   C   57  105       65       60
##   D   72  141       77       94
##   E   36   97       54       78
##   F   21   71       54       71
```

We require treatment contrasts for the RC model

```
mentalHealth$MHS <- C(mentalHealth$MHS, treatment)
mentalHealth$SES <- C(mentalHealth$SES, treatment)
```

Example: Mental Health Data

1660 residents of Manhattan cross-classified by child's mental impairment and parents' socioeconomic status (Agresti, 2013).

```
xtabs(count ~ SES + MHS, mentalHealth)
```

```
##     MHS
## SES well mild moderate impaired
##   A   64   94       58       46
##   B   57   94       54       40
##   C   57  105       65       60
##   D   72  141       77       94
##   E   36   97       54       78
##   F   21   71       54       71
```

We require treatment contrasts for the RC model

```
mentalHealth$MHS <- C(mentalHealth$MHS, treatment)
mentalHealth$SES <- C(mentalHealth$SES, treatment)
```

We fit the RC model using the ofInterest argument to specify that only the parameters of the multiplicative interaction should be shown in model summaries.

```
RC <- gnm(count ~ SES + MHS + Mult(SES, MHS), family = poisson,
          data = mentalHealth, verbose = FALSE, ofInterest = "Mult")
coef(RC)


## Coefficients of interest:
##        Mult(., MHS).SESA        Mult(., MHS).SESB
##                  0.77389                  0.78012
##        Mult(., MHS).SESC        Mult(., MHS).SESD
##                  0.26620                 -0.00646
##        Mult(., MHS).SESE        Mult(., MHS).SESF
##                 -0.67995                 -1.23326
##     Mult(SES, .).MHSwell     Mult(SES, .).MHSmild
##                  0.41623                  0.04258
## Mult(SES, .).MHSmoderate Mult(SES, .).MHSimpaired
##                 -0.02485                 -0.33519
```

The constraints that the weighted sum of column scores should sum to zero and the weighted sum of squares should sum to one are met by the scaled contrasts

$$\frac{\psi_c - \sum_c \psi_c \pi_c}{\sqrt{\sum_c \pi_c (\psi_c - \sum_c \psi_c \pi_c)^2}}$$

These contrasts can be obtained with `getContrasts` as follows:

```
colProbs <- with(mentalHealth, tapply(count, MHS, sum) / sum(count))
colScores <- getContrasts(RC, pickCoef(RC, "[.]MHS"), ref = colProbs,
                          scaleRef = colProbs, scaleWeights = colProbs)
colScores
```

```
##                          Estimate Std. Error
## Mult(SES, .).MHSwell        1.678      0.194
## Mult(SES, .).MHSmild        0.140      0.200
## Mult(SES, .).MHSmoderate   -0.137      0.280
## Mult(SES, .).MHSimpaired   -1.414      0.172
```

The constraints that the weighted sum of column scores should sum to zero and the weighted sum of squares should sum to one are met by the scaled contrasts

$$\frac{\psi_c - \sum_c \psi_c \pi_c}{\sqrt{\sum_c \pi_c (\psi_c - \sum_c \psi_c \pi_c)^2}}$$

These contrasts can be obtained with `getContrasts` as follows:

```
colProbs <- with(mentalHealth, tapply(count, MHS, sum) / sum(count))
colScores <- getContrasts(RC, pickCoef(RC, "[.]MHS"), ref = colProbs,
                          scaleRef = colProbs, scaleWeights = colProbs)
colScores
```

```
##                          Estimate Std. Error
## Mult(SES, .).MHSwell        1.678      0.194
## Mult(SES, .).MHSmild        0.140      0.200
## Mult(SES, .).MHSmoderate   -0.137      0.280
## Mult(SES, .).MHSimpaired   -1.414      0.172
```

The row scores are computed in a similar way

```
rowProbs <- with(mentalHealth, tapply(count, SES, sum) / sum(count))
rowScores <- getContrasts(RC, pickCoef(RC, "[.]SES"), ref = rowProbs,
                          scaleRef = rowProbs, scaleWeights = rowProbs)
```

Then the intrinsic association parameter can be computed directly

```
phi <- pickCoef(RC, "[.]SES", value = TRUE)
psi <- pickCoef(RC, "[.]MHS", value = TRUE)
sqrt(sum(rowProbs*(phi - sum(rowProbs*phi))^2)) *
        sqrt(sum(colProbs*(psi - sum(colProbs*psi))^2))
```

```
## [1] 0.166
```

Since this value depends on the particular scaling used for the contrasts, it typically not of interest to conduct inference on this parameter directly. The standard error could be obtained, if desired, via the delta method.

The row scores are computed in a similar way

```
rowProbs <- with(mentalHealth, tapply(count, SES, sum) / sum(count))
rowScores <- getContrasts(RC, pickCoef(RC, "[.]SES"), ref = rowProbs,
                          scaleRef = rowProbs, scaleWeights = rowProbs)
```

Then the intrinsic association parameter can be computed directly

```
phi <- pickCoef(RC, "[.]SES", value = TRUE)
psi <- pickCoef(RC, "[.]MHS", value = TRUE)
sqrt(sum(rowProbs*(phi - sum(rowProbs*phi))^2)) *
        sqrt(sum(colProbs*(psi - sum(colProbs*psi))^2))


## [1] 0.166
```

Since this value depends on the particular scaling used for the contrasts, it typically not of interest to conduct inference on this parameter directly. The standard error could be obtained, if desired, via the delta method.

The row scores are computed in a similar way

```
rowProbs <- with(mentalHealth, tapply(count, SES, sum) / sum(count))
rowScores <- getContrasts(RC, pickCoef(RC, "[.]SES"), ref = rowProbs,
                          scaleRef = rowProbs, scaleWeights = rowProbs)
```

Then the intrinsic association parameter can be computed directly

```
phi <- pickCoef(RC, "[.]SES", value = TRUE)
psi <- pickCoef(RC, "[.]MHS", value = TRUE)
sqrt(sum(rowProbs*(phi - sum(rowProbs*phi))^2)) *
        sqrt(sum(colProbs*(psi - sum(colProbs*psi))^2))


## [1] 0.166
```

Since this value depends on the particular scaling used for the contrasts, it typically not of interest to conduct inference on this parameter directly. The standard error could be obtained, if desired, via the delta method.

Other Association Models

The row-column association models introduced so far are special cases of the RC(M) model:

$$\log(\mu_{rc}) = \alpha_r + \beta_c + \sum_{k=1}^{K} \sigma_k \phi_{kr} \psi_{kc},$$

Further association models include

- Models with skew-symmetric terms
- RC(M)-L models and UNIDIFF models for three-way tables

The **logmult** package enhances **gnm** by providing functions to support analyses involving log-multiplicative models.

Other Multiplicative Models

Several models with multiplicative terms have been proposed outside of the context of association modelling.

Prominent examples include

- the stereotype model (Anderson, 1984), for ordered categorical response
- certain Rasch models, for item responses
- the Lee-Carter model (Lee and Carter, 1992) for mortality data

In some cases the multiplicative term provides a simpler, more interpretable structure, whilst in other cases it provides a simple extension to a more flexible model.

Other Multiplicative Models

Several models with multiplicative terms have been proposed outside of the context of association modelling.

Prominent examples include

- the stereotype model (Anderson, 1984), for ordered categorical response
- certain Rasch models, for item responses
- the Lee-Carter model (Lee and Carter, 1992) for mortality data

In some cases the multiplicative term provides a simpler, more interpretable structure, whilst in other cases it provides a simple extension to a more flexible model.

Other Multiplicative Models

Several models with multiplicative terms have been proposed outside of the context of association modelling.

Prominent examples include

- the stereotype model (Anderson, 1984), for ordered categorical response
- certain Rasch models, for item responses
- the Lee-Carter model (Lee and Carter, 1992) for mortality data

In some cases the multiplicative term provides a simpler, more interpretable structure, whilst in other cases it provides a simple extension to a more flexible model.

Other Multiplicative Models

Several models with multiplicative terms have been proposed outside of the context of association modelling.

Prominent examples include

- the stereotype model (Anderson, 1984), for ordered categorical response
- certain Rasch models, for item responses
- the Lee-Carter model (Lee and Carter, 1992) for mortality data

In some cases the multiplicative term provides a simpler, more interpretable structure, whilst in other cases it provides a simple extension to a more flexible model.

Diagonal Reference Terms

Diagonal reference terms model the effect of factors with common levels. For factors indexed by $i(f)$, the term is defined as

$$\sum_f w_f \gamma_{i(f)}$$

where w_f is a weight for factor f and γ_l is the *diagonal effect* for level l.

Unlike the GNMs models considered so far, which structure interaction terms, this structures the main effects of the corresponding factors.

`Dref` constrains the weights to be non-negative and to sum to one by defining them as

$$w_f = \frac{e^{\delta_f}}{\sum_i e^{\delta_i}}$$

Example: Conformity to parental rules

Data from a study of the value that parents place on their children conforming to their rules van der Slik *et al.* (2002).

Covariates are education level of mother and of father (MOPLM, FOPLF) plus 5 others.

Basic diagonal reference model for mother's conformity score (MCFM):

$$E(y_{rc}) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \frac{e^{\delta_1}}{e^{\delta_1} + e^{\delta_2}}\gamma_r + \frac{e^{\delta_2}}{e^{\delta_1} + e^{\delta_2}}\gamma_c$$

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

In order for the diagonal weights to be identified, one of the δ_f must be constrained to zero. `DrefWeights` computes the weights w_f, re-fitting the model constraining $\delta_1 = 0$ if necessary:

```
w <- DrefWeights(A)
w
```

```
## $MOPLM
## weight     se
##  0.423  0.144
##
## $FOPLF
## weight     se
##  0.577  0.144
```

Inference on the Weights

If the diagonal weights are near to 0.5, then a Normal approximation can be use to obtain a confidence interval, e.g.

```
w$MOPLM["weight"] + qnorm(c(0.025, 0.975)) * w$MOPLM["se"]


## [1] 0.140 0.705
```

Since $0 < w_f < 1$, a t-test is not a valid test of $H_0 : w_1 = 0$. Instead use anova to compare against the implied GLM, e.g.

```
## Analysis of Deviance Table
##
## Model 1: MCFM ~ AGEM + MRMM + FRMF + MWORK + MFCM + FOPLF - 1
## Model 2: MCFM ~ AGEM + MRMM + FRMF + MWORK + MFCM + Dref(MOPLM, FOPLF)
##     1
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       577        428
## 2       576        425  1      2.9    0.048
```

Inference on the Weights

If the diagonal weights are near to 0.5, then a Normal approximation can be use to obtain a confidence interval, e.g.

```
w$MOPLM["weight"] + qnorm(c(0.025, 0.975)) * w$MOPLM["se"]


## [1] 0.140 0.705
```

Since $0 < w_f < 1$, a t-test is not a valid test of $H_0 : w_1 = 0$. Instead use `anova` to compare against the implied GLM, e.g.

```
## Analysis of Deviance Table
##
## Model 1: MCFM ~ AGEM + MRMM + FRMF + MWORK + MFCM + FOPLF - 1
## Model 2: MCFM ~ AGEM + MRMM + FRMF + MWORK + MFCM + Dref(MOPLM, FOPLF)
##     1
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       577        428
## 2       576        425  1      2.9    0.048
```

The `Dref` function allows dependence of the weights on other variables.

van der Slik *et al.* (2002) consider weights dependent upon mother's conflict score (MFCM), as in

$$\delta_k = \xi_k + \phi_k x_5 \qquad (k = 1, 2)$$

which can be specified in R as

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

In this case there are two sets of weights, one for when the mother's conflict score is less than average (coded as zero) and one for when the score is greater than average (coded as one).

```
DrefWeights(F)


## $MOPLM
##   MFCM weight    se
## 1    1 0.0297 0.228
## 2    0 0.7447 0.201
##
## $FOPLF
##   MFCM weight    se
## 1    1  0.970 0.228
## 2    0  0.255 0.201
```

Custom "nonlin" Functions

A ""nonlin"" function creates a list of arguments for the internal function nonlinTerms.

The term is viewed as a function of

predictors linear predictors with coefficients to be estimated, including the special case of single parameters

variables variables included in the term with a coefficient of 1

Example: Modelling Prey Consumption

A ecology student wished to use the Holling Type II function to model the number of prey eaten by a certain predator in a given time period:

$$y(x) = \frac{ax}{1 + ahx}$$

where x is the number of prey at the start of the experiment, a is the attack rate and h is the time the predator spends handling the prey.

In addition, she wished to allow the parameters to depend on a factor specifying the catchment.

We consider the simpler model first.

Example: Modelling Prey Consumption

A ecology student wished to use the Holling Type II function to model the number of prey eaten by a certain predator in a given time period:

$$y(x) = \frac{ax}{1 + ahx}$$

where x is the number of prey at the start of the experiment, a is the attack rate and h is the time the predator spends handling the prey.

In addition, she wished to allow the parameters to depend on a factor specifying the catchment.

We consider the simpler model first.

The model can be broken down into predictors and variables as follows

$$\frac{ax}{1 + \{a\}\{h\}x}$$

We start to build our `nonlin` function as follows:

```
TypeII <- function(x){
  list(predictors = list(a = 1, h = 1),
       variables = list(substitute(x)))
}
class(TypeII) <- "nonlin"
```

The `term` argument of `nonlinTerms` takes labels for the predictors and variables and returns a deparsed expression of the term:

```
term = function(predLabels, varLabels){
    paste0(predLabels[1], "*", varLabels[1], "/(1 + ",
           predLabels[1], "*", predLabels[2], "*", varLabels[1], ")")
}
term(c("a", "h"), "x")


## [1] "a*x/(1 + a*h*x)"
```

Or using `sprintf`

```
term = function(predLabels, varLabels){
    sprintf("%s * %s / (1 + %s * %s * %s)",
            predLabels[1], varLabels[1],
            predLabels[1], predLabels[2], varLabels[1])
}
```

Complete Function

```
TypeII <- function(x){
  list(predictors = list(a = 1, h = 1),
       variables = list(substitute(x)),
       term = function(predLabels, varLabels){
           sprintf("%s * %s / (1 + %s * %s * %s)",
                   predLabels[1], varLabels[1],
                   predLabels[1], predLabels[2], varLabels[1])
})
}
class(TypeII) <- "nonlin"
```

Some test data were provided:

```
Density <- rep(c(2,5,10,15,20,30), each = 4)
Eaten <- c(1,1,0,0,2,2,1,1,1,2,3,2,2,2,3,3,3,3,4,3,3,3,4,3)
```

The counts are expected to be underdispersed so we use the `quasipoisson` family with `link = "identity"`. Both a and h should be positive, so we provide starting values

```
mod1 <- gnm(Eaten ~ -1 + TypeII(Density), start = c(a = 0.1, h = 0.1),
            family = quasipoisson(link = "identity"))


## Running main iterations.....
## Done
```

```
##
## Call:
##
## gnm(formula = Eaten ~ -1 + TypeII(Density), family = quasipoisson(link
##     start = c(a = 0.1, h = 0.1))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1154  -0.2854  -0.0166   0.3345   0.5938
##
## Coefficients:
##   Estimate Std. Error t value Pr(>|t|)
## a   0.3546     0.0778    4.56  0.00016
## h   0.1975     0.0415    4.76  9.5e-05
##
## (Dispersion parameter for quasipoisson family taken to be 0.208)
##
## Residual deviance: 5.7279 on 22 degrees of freedom
## AIC: NA
##
## Number of iterations: 7
```

Incorporating dependence

The parameters *a* and *h* can be allowed to depend on a factor as follows

```
TypeII <- function(C, x){
  list(predictors = list(a = substitute(C), h = substitute(C)),
       variables = list(substitute(x)),
       term = function(predLabels, varLabels){
           sprintf("%s * %s / (1 + %s * %s * %s)",
                   predLabels[1], varLabels[1],
                   predLabels[1], predLabels[2], varLabels[1])
})
}
class(TypeII) <- "nonlin"
```

```
Catchment <- factor(rep(1:2, 6, each = 2))
mod2 <- gnm(Eaten ~ -1 + TypeII(Catchment, Density),
            start = rep(0.2, 4),
            family = quasipoisson(link = "identity"))

## Running main iterations.........
## Done

coef(mod2)

## Coefficients:
## aCatchment1 aCatchment2 hCatchment1 hCatchment2
##       0.697       0.246       0.318       0.107
```

If instead we wanted to allow a general predictor to be supplied by the user as a formula we would use

```
TypeII <- function(f, x){
  list(predictors = list(a = f, h = f),
       variables = list(substitute(x)),
       term = function(predLabels, varLabels){
           sprintf("(%s) * %s/ (1 + (%s) * (%s) * %s)",
                   predLabels[1], varLabels[1],
                   predLabels[1], predLabels[2], varLabels[1])
})
}
class(TypeII) <- "nonlin"
```

Note additional parentheses!

```
mod2 <- gnm(Eaten ~ -1 + TypeII(~ 1 + Catchment, Density),
            start = c(0.2, -0.1, 0.2, -0.1),
            family = quasipoisson(link = "identity"))

## Running main iterations..........
## Done

coef(mod2)

## Coefficients:
## a(Intercept)  aCatchment2 h(Intercept)  hCatchment2
##        0.697       -0.451        0.318       -0.211
```

Practical IIa

1. The `voting` data in `gnm` are from the 1987 British general election. The data frame comprises the percentage voting Labour (`percentage`), the total number of people (`total`), the class of the head of household (`destination`) and the class of their father (`origin`). We shall fit a diagonal reference model to these data.

First we want to convert `percentage` into a binomial response. So that `gnm` will automatically weight the proportion of successes by the group size, we choose to do this by creating a two-column matrix with the columns giving the number of households voting Labour ('success') and the number of households voting otherwise ('failure'):

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

2. Use gnm to model yvar by a diagonal reference term based on origin and destination (see p35), with family = binomial. Look at the summary - does the model fit well? Use the mosaic function from **vcdExtra** to examine the residuals over the origin by destination table. Since the data were not provided to gnm as a table, you will need to provide a formula with the cross-classifying factors.
3. It could be that the diagonal weights should be different for the upwardly mobile. Define a variable to indicate this group as follows:

```
origin <- as.numeric(as.character(voting$origin))
destination <- as.numeric(as.character(voting$destination))
upward <- origin > destination
```

Using this variable, refit the diagonal reference model to have separate weights for the upwardly and downwardly mobile (note the stable are modelled by the diagonal effects). Do the weights differ between the two groups?

4. It could be that individuals which have come into or out of the salariat (class 1) vote differently from other individuals. Define variables indicating movement in and out of class 1 as follows:

```
in1 <- origin != 1 & destination == 1
out1 <- origin == 1 & destination != 1
```

Re-fit the diagonal reference model, specifying `~ 1 + in1 + out1` as the `formula` argument of `Dref`, so the weights are parameterized by a main effect with additional effects for `in1` and `out1`.

5. Evaluate the weights under the new model. The weights for groups that have moved in to the salariat are similar to the general weights. Fit a model that only has separate weights for the groups moving out of the salariat. Is this model a significant improvement on the standard diagonal reference model?

Practical IIb

1. The generalized logistic function or Richard's curve is defined as

$$y(t) = A + \frac{K - A}{(1 + \exp(-B(t - M))^{1/v}}$$

where

- A is the lower asymptote
- K is the upper asymptote
- B is the growth rate
- v affects near which asymptote the growth rate is at its maximum
- M is the time at which the growth rate is at its maximum

Create a custom `""nonlin""` function to fit this model.

2. Some test data are provided in the file `Richard.txt` in the data folder of the course materials. Plot `y` against `t`. Since `y` decreases as `t` increases (i.e. the growth rate is negative) the upper asymptote is the value as $t \to -\infty$ and the lower asymptote is the value as $t \to \infty$. Given that if $v = 1$, M is the value of t at which y is half-way between the lower and upper asymptotes, make reasonable guesses for starting values of A, K, B, v and M. Use `gnm` to fit the Richard's curve to these data, with `family = gaussian` and `start` set to your guessed values. Note the starting values must be in the same order as specified by the `predictors` element of your `"nonlin"` term.

3. Add the fitted line to your plot. Does the model fit well? Look at the summary for your fitted model. Are all the parameters significant? Investigate whether one or more of the following simplifications is reasonable:

- $v = 1$
- $A = 0$
- $K = 100$

Concluding Remarks

Many frequently-used GNMs can be handled by `gnm` and convenience functions for association models are available in **logmult**.

Other examples in the **gnm** vignette/documentation include

- GAMMI models (RC(M) models for a general response)
- biplot models for two-way data
- compound exponential decay curves
- double UNIDIFF model for 4-way table `?cautres`

Formula interface to `gnm` encourages experimentation and uninhibited modelling.

Agresti A (2013). *Categorical data analysis*. 3rd edition. Wiley. ISBN 9780470463635. URL https://www.wiley.com/en-gb/Categorical+Data+Analysis,+3rd+Edition-p-9780470463635.

Anderson JA (1984). "Regression and Ordered Categorical Variables." *J. R. Statist. Soc. B*, **46**(1), 1–30.

Goodman LA (1979). "Simple models for the analysis of association in cross-classifications having ordered categories." *J. Amer. Statist. Assoc.*, **74**, 537–552.

Lee RD, Carter L (1992). "Modelling and forecasting the time series of {US} mortality." *Journal of the American Statistical Association*, **87**, 659–671.

Sobel ME (1981). "Diagonal mobility models: A substantively motivated class of designs for the analysis of mobility effects." *Amer. Soc. Rev.*, **46**, 893–906.

Sobel ME (1985). "Social mobility and fertility revisited: Some new models for the analysis of the mobility effects hypothesis." *Amer. Soc. Rev.*, **50**, 699–712.

van der Slik FWP, de Graaf ND, Gerris JRM (2002). "Conformity to Parental Rules: Asymmetric Influences of Father's and Mother's Levels of Education." *Europ. Soc. Rev.*, **18**, 489–502.