gnm: A Package for Generalized Nonlinear Models

by Heather Turner and David Firth

In a generalized nonlinear model, the expectation of a response variable Y is related to a predictor η — a possibly nonlinear function of parameters — via a link function g:

$$g[E(Y)] = \eta(\beta)$$

and the variance of Y is equal to, or proportional to, a known function $v[E(Y)]$. This class of models may be thought of as extending the class of generalized linear models by allowing nonlinear terms in the predictor, or extending the class of nonlinear least squares models by allowing the variance of the response to depend on the mean.

The **gnm** package provides facilities for the specification, estimation and inspection of generalized nonlinear models. Models are specified via symbolic formulae, with nonlinear terms specified by functions of class `"nonlin"`. The fitting procedure uses an iterative weighted least squares algorithm, adapted to work with over-parameterized models. Identifiability constraints are not essential for model specification and estimation, and so the difficulty of automatically defining such constraints for the entire family of generalized nonlinear models is avoided. Constraints may be pre-specified, if desired; otherwise, functions are provided in the package for conducting inference on identifiable parameter combinations after a model in an over-parameterized representation has been fitted.

Specific models which the package may be used to fit include models with multiplicative interactions, such as row-column association models (Goodman, 1979), UNIDIFF (uniform difference) models for social mobility (Xie, 1992; Erikson and Goldthorpe, 1992), GAMMI (generalized additive main effects and multiplicative interaction) models (e.g. van Eeuwijk, 1995), and Lee-Carter models for trends in age-specific mortality (Lee and Carter, 1992); diagonal-reference models for dependence on a square or hyper-square classification (Sobel, 1981, 1985); Rasch-type logit or probit models for legislative voting (e.g. de Leeuw, 2006); and stereotype multinomial regression models for ordinal response (Anderson, 1984). A comprehensive manual is distributed with the package (see `vignette("gnmOverview", package = "gnm")`) and this manual may also be downloaded from `http://go.warwick.ac.uk/gnm`. Here we give an introduction to the key functions and provide some illustrative applications.

Key Functions

The primary function defined in package **gnm** is the model-fitting function of the same name. This function is patterned after `glm` (the function included in the standard **stats** package for fitting generalized linear models), taking similar arguments and returning an object of class `c("gnm", "glm", "lm")`.

A model formula is specified to `gnm` as the first argument. The conventional symbolic form is used to specify linear terms, whilst nonlinear terms are specified using functions of class `"nonlin"`. The `nonlin` functions currently exported in **gnm** are summarized in Figure 1. These functions enable the specification of basic mathematical functions of predictors (`Exp`, `Inv` and `Mult`) and some more specialized nonlinear terms (`MultHomog`, `Dref`). Often arguments of `nonlin` functions may themselves be parametric functions described in symbolic form. For example, an exponential decay model with additive errors

$$y = \alpha + \exp(\beta + \gamma x) + e \qquad (1)$$

is specified using the `Exp` function as

```
gnm(y ~ Exp(1 + x))
```

These "sub-formulae" (to which intercepts are *not* added by default) can include other `nonlin` functions, allowing more complex nonlinear terms to be built up. Users may also define custom `nonlin` functions, as we shall illustrate later.

`Dref`	to specify a diagonal reference term
`Exp`	to specify the exponential of a predictor
`Inv`	to specify the reciprocal of a predictor
`Mult`	to specify a product of predictors
`MultHomog`	to specify a multiplicative interaction with homogeneous effects

Figure 1: `"nonlin"` functions in the **gnm** package.

The remaining arguments to gnm are mainly control parameters and are of secondary importance. However, the `eliminate` argument — which implements a feature seen previously in the *GLIM 4* statistical modelling system — can be extremely useful when a model contains the additive effect of a (typically 'nuisance') factor with a large number of levels; in such circumstances the use of `eliminate` can substantially improve computational efficiency, as well as allowing more convenient model summaries with the 'nuisance' factor omitted.

Most of the methods and accessor functions implemented for `"glm"` or `"lm"` objects are also implemented for `"gnm"` objects, such as `print`, `summary`, `plot`, `coef`, and so on. Since `"gnm"` models may be

over-parameterized, care is needed when interpreting the output of some of these functions. In particular, for parameters that are not identified, an arbitrary parameterization will be returned and the standard error will be displayed as `NA`.

For estimable parameters, methods for `profile` and `confint` enable inference based on the profile likelihood. These methods are designed to handle the asymmetric behaviour of the log-likelihood function that is a well-known feature of many nonlinear models.

Estimates and standard errors of simple "sum-to-zero" contrasts can be obtained using `getContrasts`, if such contrasts are estimable. For more general linear combinations of parameters, the lower-level `se` function may be used instead.

Example

We illustrate here the use of some of the key functions in **gnm** through the analysis of a contingency table from Goodman (1979). The data are a cross-classification of a sample of British males according to each subject's occupational status and his father's occupational status. The contingency table is distributed with **gnm** as the data set `occupationalStatus`.

Goodman (1979) analysed these data using row-column association models, in which the interaction between the row and column factors is represented by components of a multiplicative interaction. The log-multiplicative form of the RC(1) model — the row-column association model with one component of the multiplicative interaction — is given by

$$\log \mu_{rc} = \alpha_r + \beta_c + \gamma_r \delta_c, \qquad (2)$$

where μ_{rc} is the cell mean for row r and column c. Before modelling the occupational status data, Goodman (1979) first deleted cells on the main diagonal. Equivalently we can separate out the diagonal effects as follows:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \delta_c \qquad (3)$$

where

$$\theta_{rc} = \begin{cases} \phi_r & \text{if } r = c \\ 0 & \text{otherwise.} \end{cases}$$

In addition to the `"nonlin"` functions provided by **gnm**, the package includes a number of functions for specifying structured linear interactions. The diagonal effects in model 3 can be specified using the **gnm** function `Diag`. Thus we can fit model 3 as follows:

```
> set.seed(1)
> RC <- gnm(Freq ~ origin + destination +
+     Diag(origin, destination) +
+     Mult(origin, destination),
+     family = poisson,
+     data = occupationalStatus)
```

An abbreviated version of the output given by `summary(RC)` is shown in Figure 2. Default constraints (as specified by `options("contrasts")`) are applied to linear terms: in this case the first level of the `origin` and `destination` factors have been set to zero. However the "main effects" are not estimable, since they are aliased with the unconstrained multiplicative interaction. If the `gnm` call were re-evaluated from a different random seed, the parameterization for the main effects and multiplicative interaction would differ from that shown in Figure 2. On the other hand the diagonal effects, which are essentially contrasts with the off-diagonal elements, are identified here. This is immediately apparent from Figure 2, since standard errors are available for these parameters.

One could consider extending the model by adding a further component of the multiplicative interaction, giving an RC(2) model:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \delta_c + \theta_r \phi_c. \qquad (4)$$

Most of the `"nonlin"` functions, including `Mult`, have an `inst` argument to allow the specification of multiple instances of a term, as here:

```
Freq ~ origin + destination +
    Diag(origin, destination) +
    Mult(origin, destination, inst = 1) +
    Mult(origin, destination, inst = 2)
```

Multiple instances of a term with an `inst` argument may be specified in shorthand using the `instances` function provided in the package:

```
Freq ~ origin + destination +
    Diag(origin, destination) +
    instances(Mult(origin, destination), 2)
```

The formula is then expanded by `gnm` before the model is fitted.

In the case of the occupational status data however, the RC(1) model is a good fit (a residual deviance of 29.149 on 28 degrees of freedom). So rather than extending the model, we shall see if it can be simplified by using homogeneous scores in the multiplicative interaction:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \gamma_c \qquad (5)$$

This model can be obtained by updating `RC` using `update`:

```
> RChomog <- update(RC, . ~ .
+     - Mult(origin, destination)
+     + MultHomog(origin, destination))
```

We can compare the two models using `anova`, which shows that there is little gained by allowing heterogeneous row and column scores in the association of the fathers' occupational status and the sons' occupational status:

```
Call:
gnm(formula = Freq ~ origin + destination + Diag(origin, destination) +
    Mult(origin, destination), family = poisson, data = occupationalStatus)

Deviance Residuals: ...

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                   0.11881         NA      NA       NA
origin2                       0.49005         NA      NA       NA
...
origin8                       1.60214         NA      NA       NA
destination2                  0.95334         NA      NA       NA
...
destination8                  1.42813         NA      NA       NA
Diag(origin, destination)1    1.47923    0.45401   3.258  0.00112
...
Diag(origin, destination)8    0.40731    0.21930   1.857  0.06327
Mult(., destination).origin1 -1.74022         NA      NA       NA
...
Mult(., destination).origin8  1.65900         NA      NA       NA
Mult(origin, .).destination1 -1.32971         NA      NA       NA
...
Mult(origin, .).destination8  0.66730         NA      NA       NA

...
Residual deviance: 29.149 on 28 degrees of freedom
...
```

Figure 2: Abbreviated summary of model `RC`.

```
> anova(RChomog, RC)
Analysis of Deviance Table

Model 1: Freq ~ origin + destination +
    Diag(origin, destination) +
    MultHomog(origin, destination)
Model 2: Freq ~ origin + destination +
    Diag(origin, destination) +
    Mult(origin, destination)
  Resid. Df Resid. Dev Df Deviance
1        34     32.561
2        28     29.149  6    3.412
```

All of the parameters in model 5 can be made identifiable by constraining one level of the homogeneous multiplicative factor to zero. This can be achieved by using the `constrain` argument to `gnm`, but this would require re-fitting the model. As we are only really interested in the parameters of the multiplicative interaction, we investigate simple contrasts of these parameters using `getContrasts`. The **gnm** function `pickCoef` enables us to select the parameters of interest using a regular expression to match against the parameter names:

```
> contr <- getContrasts(RChomog,
+     pickCoef(RChomog, "MultHomog"))
```

A summary of `contr` is shown in Figure 3. By default, `getContrasts` sets the first level of the homogeneous multiplicative factor to zero. The quasi standard errors and quasi variances are independent of parameterization; see Firth (2003) and Firth and de Menezes (2004) for more detail. In this example the reported relative-error diagnostics indicate that the quasi-variance approximation is rather inaccurate — in contrast to the high accuracy found in many other situations by Firth and de Menezes (2004).

Users may run through the example covered in this section by calling `demo(gnm)`.

Custom `"nonlin"` functions

The small number of `"nonlin"` functions currently distributed with **gnm** allow a large class of generalized nonlinear models to be specified, as exemplified through the package documentation. Nevertheless, for some models a custom `"nonlin"` function may be necessary or desirable. As an illustration, we shall consider a logistic predictor of the form given by `SSlogis` (a selfStart model for use with the **stats** function `nls`),

$$\frac{Asym}{1 + \exp((xmid - x)/scal)}. \tag{6}$$

This predictor could be specified to `gnm` as

```
~ -1 + Mult(1, Inv(Const(1) +
    Exp(Mult(1 + offset(-x), Inv(1)))))
```

```
Model call:  gnm(formula = Freq ~ origin + destination + Diag(origin, destination)
    + MultHomog(origin, destination), family = poisson, data = occupationalStatus)
                                  estimate    SE quasiSE quasiVar
    MultHomog(origin, destination)1    0.000 0.000  0.1573  0.02473
    MultHomog(origin, destination)2    0.218 0.235  0.1190  0.01416
    MultHomog(origin, destination)3    0.816 0.167  0.0611  0.00374
    MultHomog(origin, destination)4    1.400 0.160  0.0518  0.00269
    MultHomog(origin, destination)5    1.418 0.172  0.0798  0.00637
    MultHomog(origin, destination)6    1.929 0.157  0.0360  0.00129
    MultHomog(origin, destination)7    2.345 0.173  0.0796  0.00634
    MultHomog(origin, destination)8    2.589 0.189  0.1095  0.01200
Worst relative errors in SEs of simple contrasts (%):  -19 4.4
Worst relative errors over *all* contrasts (%):  -17.2 51.8
```

Figure 3: Simple contrasts of homogeneous row and column scores in model `RChomog`.

where `Mult`, `Inv` and `Exp` are `"nonlin"` functions, `offset` is the usual **stats** function and `Const` is a **gnm** function specifying a constant offset. However, this specification is rather convoluted and it would be preferable to define a single `"nonlin"` function that could specify a logistic term.

Our custom `"nonlin"` function only needs a single argument, to specify the variable x, so the skeleton of our definition might be as follows:

```
Logistic <- function(x){
    }
class(Logistic) <- "nonlin"
```

Now we need to fill in the body of the function. The purpose of a `"nonlin"` function is to convert its arguments into a list of arguments for the internal function `nonlinTerms`. This function considers a nonlinear term as a mathematical function of *predictors*, the parameters of which need to be estimated, and possibly also *variables*, which have a coefficient of 1. In this terminology, *Asym*, *xmid* and *scal* in Equation 6 are parameters of intercept-only predictors, whilst x is a variable. Our `Logistic` function must specify the corresponding `predictors` and `variables` arguments of `nonlinTerms`. We can define the `predictors` argument as a named list of symbolic expressions

```
predictors = list(Asym = 1, xmid = 1,
    scal = 1)
```

and pass the user-specified variable `x` as the `variables` argument:

```
variables = list(substitute(x))
```

We must then define the `term` argument of `nonlinTerms`, which is a function that creates a deparsed mathematical expression of the nonlinear term from labels for the predictors and variables. These labels should be passed through arguments `predLabels` and `varLabels` respectively, so we can specify the `term` argument as

```
term = function(predLabels, varLabels){
    paste(predLabels[1], "/(1 + exp((",
    predLabels[2], "-", varLabels[1], ")/",
    predLabels[3], "))")
    }
```

Our `Logistic` function need not specify any further arguments of `nonlinTerms`. However, we do have an idea of useful starting values, so we can also specify the `start` argument. This should be a function which takes a named vector of the parameters in the term and returns a vector of starting values for these parameters. The following function will set the initial scale parameter to one:

```
start = function(theta){
    theta[3] <- 1
    theta
    }
```

Putting these components together, we have:

```
Logistic <- function(x){
    list(predictors =
        list(Asym = 1, xmid = 1, scal = 1),
        variables = list(substitute(x)),
        term =
        function(predLabels, varLabels){
            paste(predLabels[1],
            "/(1 + exp((", predLabels[2],
            "-", varLabels[1], ")/",
            predLabels[3], "))")
        },
        start = function(theta){
            theta[3] <- 1
            theta
        })
}
class(Logistic) <- "nonlin"
```

Having defined this function, we can reproduce the example from `?SSlogis` as shown below

```
> Chick.1 <-
+ ChickWeight[ChickWeight$Chick == 1, ]
```

```
> fm1gnm <- gnm(weight ~ Logistic(Time) - 1,
+    data = Chick.1, trace = FALSE)
> summary(fm1gnm)

Call:
gnm(formula = weight ~ Logistic(Time) - 1,
   data = Chick.1)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-4.154  -2.359   0.825   2.146   3.745

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
Asym 937.0205   465.8569   2.011  0.07516
xmid  35.2228     8.3119   4.238  0.00218
scal  11.4052     0.9052  12.599 5.08e-07

(Dispersion parameter for gaussian family
   taken to be 8.51804)

Residual deviance: 76.662 on 9 degrees of
   freedom
AIC: 64.309

Number of iterations: 14
```

In general, whenever a nonlinear term can be represented as an expression that is differentiable using `deriv`, it should be possible to define a `"nonlin"` function to specify the term. Additional arguments of `nonlinTerms` allow for the specification of factors with homologous effects and control over the way in which parameters are automatically labelled.

Summary

The functions distributed in the **gnm** package enable a wide range of generalized nonlinear models to be estimated. Nonlinear terms that are not (easily) represented by the `"nonlin"` functions provided may be implemented as custom `"nonlin"` functions, under fairly weak conditions.

There are several features of **gnm** that we have not covered in this paper. Some facilitate model inspection, such as the `ofInterest` argument to `gnm` which allows the user to customize model summaries and simplify parameter selection. Other features are designed for particular models, such as the `residSVD` function for generating good starting values for a multiplicative interaction. Still others relate to particular types of data, such as the `expandCategorical` function for expressing categorical data as counts. These features complement the facilities we have already described, to produce a substantial and flexible package for working with generalized nonlinear models.

Acknowledgments

This work was supported by the Economic and Social Research Council (UK) through Professorial Fellowship RES-051-27-0055.

Bibliography

J. A. Anderson. Regression and ordered categorical variables. *J. R. Statist. Soc. B*, 46(1):1–30, 1984.

J. de Leeuw. Principal component analysis of binary data by iterated singular value decomposition. *Comp. Stat. Data Anal.*, 50(1):21–39, 2006.

R. Erikson and J. H. Goldthorpe. *The Constant Flux*. Oxford: Clarendon Press, 1992.

D. Firth. Overcoming the reference category problem in the presentation of statistical models. *Sociological Methodology*, 33:1–18, 2003.

D. Firth and R. X. de Menezes. Quasi-variances. *Biometrika*, 91:65–80, 2004.

L. A. Goodman. Simple models for the analysis of association in cross-classifications having ordered categories. *J. Amer. Statist. Assoc.*, 74:537–552, 1979.

R. D. Lee and L. Carter. Modelling and forecasting the time series of US mortality. *Journal of the American Statistical Association*, 87:659–671, 1992.

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

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

F. A. van Eeuwijk. Multiplicative interaction in generalized linear models. *Biometrics*, 51:1017–1032, 1995.

Y. Xie. The log-multiplicative layer effect model for comparing mobility tables. *American Sociological Review*, 57:380–395, 1992.

Heather Turner
University of Warwick, UK
`Heather.Turner@warwick.ac.uk`

David Firth
University of Warwick, UK
`David.Firth@warwick.ac.uk`