Note: some of these slides are out of date. Please check against the current version of the package manual (gnmOverview.pdf).

Introduction to Generalized Nonlinear Models in Social Research

David Firth and Heather Turner

ESRC National Centre for Research Methods
and
Department of Statistics
University of Warwick

ESRC Oxford Spring School, 2005–12–12/13

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. Many useful models in social science are non-linear.

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

This is a fairly specialized course. A much broader view of statistical modelling can be found in another Spring School course, *An Overview of Statistical Models and Statistical Thinking*.

Computer lab sessions will provide some familiarity with *gnm*.

Plan

Part I: Introduction
Part II: Models with multiplicative terms
Part III: Two larger examples

Part I: Introduction

Linear and generalized linear models

Generalized nonlinear models

Structured interactions

Introduction to the gnm package

Exercise

Part II: Models with multiplicative terms

Introduction

Row-column association

Rasch-type models, ideal-point models of voting

UNIDIFF (log-multiplicative) models for strength of association

Stereotype model for ordinal response

Multiplicative effects or heteroscedasticity?

Exercises

Part III: Two larger examples

Conformity to parental rules: diagonal reference models

Voting in the US House of Representatives: Logit/probit ideal-point models

Exercises

Part I

Introduction

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 model:

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

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

Examples:

- binary logistic regressions (including Rasch models, Bradley-Terry models, etc.)
- 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., 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 $\partial\eta/\partial\beta = X(\beta)$].

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 in computer language

```
> gnm(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)`
- (with categories ordered) uniform association,
  `y ~ row + col + Rscore:Cscore`
  where `Rscore` and `Cscore` are (possibly scaled versions of) the row and column index numbers.

Some applications demand more complex, subject-matter-driven interaction structures.

In social-class mobility studies various 'levels' or 'topological' association structures have been proosed. For example Xie (1992) uses, for 7 social classes, the 6-level association structure

```
2 3 4 6 5 6 6
3 3 4 6 4 5 6
4 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
```

The *gnm* package provides a special function `Topo`, in order to facilitate working with such structured interactions. See `?Topo`.

Row-column association

The uniform association model 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.

A natural generalization is to allow the *data* to determine the scores instead. 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.

Higher-order interactions

The number of parameters in an unstructured interaction term, for example 3-way γ_{rct}, can become very large.

Structured versions can help with both statistical efficiency and interpretation.

A nice example of this is the UNIDIFF model for 'similar' association in a set of 2-way tables: more tomorrow.

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 R's standard `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.)

Limitations: An important limitation of *gnm* (and indeed of the standard `glm`) is to models in which the mean-predictor function is completely determined by available explanatory variables. Latent variables (random effects) are not handled.

Non-linear model terms

The two key functions `Mult` and `Nonlin` are 'symbolic wrappers' for use inside model formulas.

A multiplicatively structured interaction is specified as `Mult(first, second)`. For example, a term of the form

$$(\alpha + \beta x)\gamma_{jk}$$

where j and k index levels of factors A and B, would be specified as `Mult(x, A:B)`.

Or, for a multiplier which depends on x but which is guaranteed positive, we can use `Mult(Exp(-1 + x), A:B)`, corresponding mathematically to $\exp(\beta x)\gamma_{jk}$.

The `Nonlin` function

Not all nonlinear terms are products of independently-specified 'constituent multipliers'.

Example: homogeneous row and column scores,

$$\alpha_r + \beta_c + \phi_r \phi_c$$

(Goodman, 1979) — `Nonlin(MultHomog(row, col))`

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

$$w_1 \gamma_r + w_2 \gamma_c$$

(Sobel, 1981, 1985) — `Nonlin(Dref(row, col))`

Any (differentiable) nonlinear term can be specified using `Nonlin`.

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').

A simple example: $\phi_r \psi_c$ is equivalent to $(2\phi_r)(\psi_c/2)$.

The *gnm* package provides various tools (`checkEstimable`, `getContrasts`, `se`) for checking the estimability of parameter combinations, and for obtaining valid standard errors for estimable combinations.

Control over the fitting process

The gnm function has various optional arguments to allow the user to control aspects of the ML fitting process. These include

- convergence criteria (`tolerance`, `iterMax`)
- starting values (`start`)
- the printing of information at each iteration (`trace`).

In many *gnm* models, random starting values are used by default. This in turn gives a random representation of the model.

Exercise

In the computer lab, your login name is ‘C*n*’, where n is the number of your terminal (e.g., C9). The password is (to be notified orally).

After login, **drag** (i.e., **copy**) the folder ‘Generalized nonlinear models’ from ‘S:\springschool05\FirthTurner’ (found via ‘My Computer’ on the Start menu) **to your Desktop**. Inside that folder — the folder now on your desktop, that is — is an R workspace icon: just double-click it to start R.

1. Load the *gnm* package, then load 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. We shall consider a number of models for these frequencies, which by default are named `Freq`.

The null association model assumes that the origin (o) and destination (d) are independent and the frequencies can be modelled by main effects only:

$$\log E(\texttt{Freq}_{od}) = \theta + \alpha_o + \beta_d$$

Fit this model using `gnm` with *family* = *poisson*, assigning the result to a suitable name. Print this object.

4. *gnm* objects inherit from *glm* and *lm* objects, i.e. the methods used by generic functions for *gnm* objects may be the same as, or based on, those for *glm* and *lm* objects. Use `apropos("^plot")` to search for help files on objects beginning with “plot” and open the one most relevant for *gnm* objects.

From the help page, find out how to use `plot` to create a plot of residuals vs. fitted values and do this for the null association model. The poor fit should be very apparent!

5. Create a vector of equally-spaced row scores for the cells in the table:

```
Rscore <- as.vector(row(occupationalStatus))
```

Create a vector of column scores named `Cscore` in a similar way (by using `col` in place of `row`).

These score vectors can be used to fit a uniform association model:

$$\log E(\texttt{Freq}_{od}) = \theta + \alpha_o + \beta_d + \gamma(\texttt{Rscore})(\texttt{Cscore})$$

again using `gnm`. The extra term can be represented in the model formula by `Rscore:Cscore`.

Fit the uniform association model, assigning the result a different name from the null model. Print the resulting object and look at a residual vs. fitted plot. Look at the effect of modelling the diagonal elements separately, by adding `Diag(origin, destination)`.

6. Keeping the `Diag` term in the model, use `gnm` to fit a model with a homogenoeous multiplicative interaction between origin and destination instead of the uniform association term (using `Nonlin(MultHomog(...))`; see p7 of *GnmR*).

7. Since `occupationalStatus` is a table, the residual component of the *gnm* object is also a table. Use `residuals` to access the deviance residuals, obtain the absolute values of these residuals using `abs`, then plot the result. Note the residuals for the diagonal elements are essentially zero because there is one `Diag` parameter for each diagonal cell.

8. Use `coef` to access the coefficients of the model and assign the result. Re-fit the model using `update` 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?

9. Use `getContrasts` to estimate simple contrasts of the parameters in the interaction term. Re-fit the model using the argument `constrain = "pick"` to set the last parameter of the interaction term to zero. Compare the parameters of the interaction term to the output of `getContrasts`. Save the coefficients and re-fit the model, this time setting a different parameter of the interaction term to zero. Compare the coefficients of the two models: which are the same in both?

10. Now fit a model with a heterogeneous multiplicative interaction (using `Mult(...)`; see *GnmR* p6), assigning the result.
11. Use `deviance` to extract the deviance from the model object. Look at the effect on the deviance when i) the intercept of the first multiplicative factor is constrained and ii) the last parameter of the first multiplicative factor is constrained. Can you explain your observations?
12. Use `anova` to compare all the models fitted to the `occupationalStatus` data. Choose the best model in terms of fit and simplicity. Use `plot` to check for any problems, e.g. outliers with high leverage, trends in the residuals, non-normal residuals, etc.

Part II

Models with multiplicative terms

Multiplicative terms, in an otherwise-additive predictor function, impose (hopefully interpretable!) structure on interactions.

Prominent examples include:

- Row-column association
- Certain Rasch models, including ideal-point models of legislator voting
- UNIDIFF-type models, e.g. as used in social mobility
- the 'stereotype' regression model of Anderson (1984), for ordered categorical response

Row-column association models

RC(1): $\alpha_r + \beta_c + \gamma_r \delta_c$,

```
row + col + Mult(row, col)
```

RC(2): $\alpha_r + \beta_c + \gamma_r^{(1)} \delta_c^{(1)} + \gamma_r^{(2)} \delta_c^{(2)}$

```
row + col + Mult(row, col, multiplicity = 2)
```

etc.

Much developed by Goodman, Clogg, Becker (1970s, 1980s), with extensions to higher-way tables, etc.

Rasch-type models, ideal-point models of voting

The 'simple' Rasch model for the binary response y_{is} of subject s to test item i is a logistic regression,

$$\text{logit}(\mu_{is}) = \gamma_s - \alpha_i$$

in which γ_s denotes the ability of subject s, and α_i the difficulty of item i.

Lots of applications, especially for more elaborate forms of the model.

In practice, it is often found that the assumption of 'equal (and without loss of generality, unit) slopes' fails to hold: some items are better at discriminating than are others.

Birnbaum's '2-parameter' version generalizes the model to address this:

$$\text{logit}(\mu_{is}) = \beta_i \gamma_s - \alpha_i$$

Application to scaling of legislative votes:

- legislator m votes yes/no on roll call r
- each legislator has a notional 'ideological position', γ_m
- for each roll call r, $\text{logit}[\text{pr}(yes)] = \alpha_r + \beta_r \gamma_m$
- generalization: position in two or more dimensions.

More in part III.

UNIDIFF-type models

High-order interactions in their 'raw' form can involve large numbers of parameters. Often more economical/interpretable summaries are possible.

The classic 'UNIDIFF' model relates to a 3-way table of counts y_{rct}, viewed as a set of T two-way tables $y_{rc1}, y_{rc2}, \dots, y_{rcT}$.

Interest is in the row-column association, and variation between tables t in the strength of that association.

The UNIDIFF model postulates a common pattern of (log) odds ratios, modulated by a constant that is specific to each table:

$$\log(\mu_{rct}) = \alpha_{rt} + \beta_{ct} + e^{\gamma_t}\delta_{rc}$$

$$\log(\mu_{rct}) = \alpha_{rt} + \beta_{ct} + e^{\gamma_t}\delta_{rc}$$

In R:

```
> gnm(y ~ row:table + col:table
      + Mult(Exp(table - 1), row:col),
      family = poisson)
```

This model has been highly influential in comparative sociological studies of class and mobility. Interest focuses on the γ_t parameters.

(Note that only the *differences* $\gamma_t - \gamma_s$ are estimable/interpretable.)

Generalizations, specializations:

- the set of tables itself is structured, e.g., arranged serially in time, or is a cross-classification of countries and years. Then γ_t may itself be related to other variables, for example

$$\gamma_t = \gamma t \quad \text{or} \quad \gamma_{tc} = \gamma_t + \gamma_c$$

- the same multipliers might be assumed to affect more than one set of associations, e.g.,

$$\log(\mu_{rclt}) = \alpha_{rt} + \beta_{ct} + \phi_{lt} + e^{\gamma_t}(\delta_{rc} + \epsilon_{cl})$$

- the assumed-common association pattern(s) may themselves be simplified, for example by a topological 'levels' structure.

etc., etc.

Stereotype Models

The stereotype model (Anderson, 1984) is suitable for ordered categorical data. It is a special case of the multinomial logistic model:

$$pr(y_i = c|\boldsymbol{x}_i) = \frac{\exp(\beta_{0c} + \boldsymbol{\beta}_c^T \boldsymbol{x}_i)}{\sum_r \exp(\beta_{0r} + \boldsymbol{\beta}_r^T \boldsymbol{x}_i)}$$

in which only the *scale* of the relationship with the covariates changes between categories:

$$pr(y_i = c|\boldsymbol{x}_i) = \frac{\exp(\beta_{0c} + \gamma_c \boldsymbol{\beta}^T \boldsymbol{x}_i)}{\sum_r \exp(\beta_{0r} + \gamma_r \boldsymbol{\beta}^T \boldsymbol{x}_i)}$$

The stereotype model can be fitted using `gnm` by re-expressing the categorical data as counts and fitting the log-linear model

$$\log \mu_{ic} = \beta_{0c} + \gamma_c \sum_r \boldsymbol{\beta}_r \boldsymbol{x}_{ir}.$$

We can look at one of the examples from Anderson's paper:

```
> data(backPain)
> backPain[1:5, ]
```

We need to express each measurement of `pain` as a set of counts, equal to 1 in the correct category and 0 elsewhere.

The counts can be obtained using `class.ind` from package *nnet*

```
> library(nnet)
> .incidence <- class.ind(backPain$pain)
> .counts <- as.vector(t(.incidence))
```

Then we need to create a factor identifying each original observation and a factor identifying the different categories:

```
> .rowID <- factor(t(row(.incidence)))
> backPain <- backPain[.rowID, ]
```

```
> backPain$pain <- C(factor(
                        rep(levels(backPain$pain),
                        nrow(.incidence)),
                        levels = levels(backPain$pain),
                        ordered = TRUE),
                    treatment)
```

Let's take a look at what all this data manipulation has achieved:

```
> cbind(.rowID[1:12], .counts[1:12],
        backPain[1:12, 4:1])
```


The sterotype model can then be fitted as follows

```
> oneDimensional <- gnm(
    .counts ~ .rowID + pain
                + Mult(pain - 1, x1 + x2 + x3 - 1),
    family = poisson, data = backPain)
> oneDimensional
```

The `.rowID` parameters are a bit of a nuisance. A better approach is to use the *eliminate* argument of `gnm` to specify that the `.rowID` parameters replace the intercept in the model. Then `gnm` will use a method exploiting the structure of these parameters in order to improve the computational efficiency of their estimation, and the parameters will be excluded from summaries of the model object.


```
> oneDimensional <- gnm(
    .counts ~ pain + Mult(pain - 1, x1 + x2 + x3 - 1),
    eliminate = ~.rowID,
    family = poisson, data = backPain)
> oneDimensional
> vcov(oneDimensional)
```

We can compare the stereotype model to the multinomial logistic model:

```
> threeDimensional <- gnm(
    .counts ~ pain + pain:(x1 + x2 + x3),
    eliminate = ~.rowID,
    family = poisson, data = backPain)
```


A note of caution on interpretation

Care is needed in interpreting apparent multiplicative effects.

For example, in political science much use has been made of generalized logit and probit models in which the standard binary-response assumption (in terms of probit) $\text{pr}(y_i = 1) = \Phi(x_i'\beta/\sigma)$ is replaced by a model which allows non-constant variance in the underlying latent regression:

$$\text{pr}(y_i = 1) = \Phi[x_i'\beta/\exp(z_i'\gamma)]$$

This clearly results in a multiplicative model for the mean: in R, the above would be specified as

```
> gnm(y ~ -1 + Mult(x, Exp(z)), family = binomial(link="probit"))
```

It is therefore impossible to distinguish, with binary data, between two distinct generative mechanisms: underlying variance depends on z; or effect of x is modulated by z.

Exercise: UNIDIFF model for social mobility

Ths exercise uses a dataset kindly provided by Louis-André Vallet, on mobility among seven social classes in France between 1970 and 1993.

1. Load the dataset *France* into your R workspace, and view it as a table:

```
> load("Data/France.RData")
> xtabs(Freq ~ orig + dest + year, France)
```

2. Fit the 'constant social fluidity' log-linear model in which `orig` and `dest` have the same association in all four survey years:

```
> CSFmodel <- gnm(
      Freq ~ orig:year + dest:year + orig:dest,
      family = poisson, data = France)
```


3. Now test whether the strength of association between `orig` and `dest` differs from year to year, using the UNIDIFF model:

```
> UNIDIFF <- update(CSFmodel, . ~ . - orig:dest
                    + Mult(Exp(-1 + year), orig:dest))
> anova(CSFmodel, UNIDIFF)
```

You should find that the UNIDIFF model is a significant improvement, but still exhibits significant lack of fit.

4. Look at the year coefficients in the `Mult` term, by picking out the four relevant coefficients from the list provided by

```
> getContrasts(UNIDIFF)
```

Interpret the estimated coefficients. (Note that the reported standard errors will be too optimistic, on account of the observed lack of fit.)

5. Since the four survey years are roughly equally spaced, it might be possible to summarize the change in mobility by a straight-line trend. We can do this by converting `year` from a 4-level factor to a quantitative variable, and then re-fitting:

```
> time <- as.numeric(France$year)
> UNIDIFFtrend <- update(UNIDIFF, . ~ .
                   - Mult(Exp(-1 + year), orig:dest)
                   + Mult(Exp(-1 + time), orig:dest))
> anova(CSFmodel, UNIDIFFtrend, UNIDIFF)
```


Exercise: Stereotype model for back pain

1. Load the `backPain` data set and work through the commands on p42 to re-express the data as counts.

2. Using `gnm`, fit the empty 'baseline' model:

```
gnm(.counts ~ pain, eliminate = .rowID,
    family = poisson, data = backpain)
```

Print the result. This model assumes that the probability of an individual experiencing a given level of pain is the same regardless of the values of the prognostic variables.

3. Use `update` to extend the null model to the stereotype model on p40 and interpret the result.

4. In order to make the category-specific multipliers (`Mult.Factor1.painworse` etc.) identifiable — so that, for example, valid standard errors can be calculated — we must constrain both the location and the scale of these parameters.

Using `getContrasts` would fix the location by setting one parameter to zero. Confirm that this constraint is insufficient by running `getContrasts` on these parameters.

5. We can constrain the scale by setting the coefficient of one of the variables in the second constituent multiplier to one. This can be achieved by treating one of the variables as an 'offset' in the second multiplier rather than a variable whose coefficient needs to be estimated.

Refit the model replacing `x1` with `offset(x1)` in the formula for the second constituent multiplier.

Use `getContrasts` to estimate simple contrasts of the category-specific multipliers in the new model.

6. The stereotype model is clearly an improvement on the null model, but is it necessary to have a separate multiplier for each category of pain? The estimates from `getContrasts` are very similiar for categories "same" and "slight.improvement". We can try fitting a common multiplier for these two categories.

Load package *car* and create a new factor from `backPain$pain`, merging the second and third categories as follows:

```
newPain <- recode(backPain$pain,
            "c('same', 'slight.improvement') =
               'same|slight.improvement'")
```

Re-fit the stereotype model using the new factor in the formula for the first constituent multiplier and look at the impact on the model deviance.

7. Using `getContrasts` to identify the group-specific multipliers, choose the two that are most similar and refit the model with a common multiplier for the corresponding groups.

Repeat until you have a model with just two group-specific multipliers.

How many different multipliers are necessary?

Part III

Two larger examples

Diagonal reference models: Conformity to parental rules

Data from van der Slik et al, (2002).

An analysis of the value that parents place on their children conforming to their rules.

Two response variables: mother's conformity score (MCFM), father's (FCFF).

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

Basic diagonal reference model for MCFM:

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

Fit this by

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



```
> coef(A)
                    AGEM                     MRMM                     FRMF
                 0.06364                 -0.32425                 -0.25324
                   MWORK                     MFCM  Dref(MOPLM, FOPLF).MOPLM
                -0.06430                 -0.06043                  0.34389
Dref(MOPLM, FOPLF).FOPLF     Dref(MOPLM, FOPLF).1     Dref(MOPLM, FOPLF).2
                 0.65611                  4.95123                  4.86328
    Dref(MOPLM, FOPLF).3     Dref(MOPLM, FOPLF).4     Dref(MOPLM, FOPLF).5
                 4.86458                  4.72342                  4.43516
    Dref(MOPLM, FOPLF).6     Dref(MOPLM, FOPLF).7
                 4.18873                  4.43379
> prop.table(exp(coef(A)[6:7]))
Dref(MOPLM, FOPLF).MOPLM Dref(MOPLM, FOPLF).FOPLF
               0.4225734                0.5774266
```

So, controlling for the other covariates, father's education is estimated to carry about 58% of the total effect of parents' education.

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 +
           Nonlin(Dref(MOPLM, FOPLF, formula = ~ 1 + MFCM)),
           family = gaussian,
           data = conformity, verbose = FALSE)
```

And so on. See Section 6.3 of *GnmR* for more details.

Logistic ideal-point models for legislator voting

Data on 20 roll calls from the US House of Representatives in 2001, each coded 0/1 such that 1 indicates liberality.

Idea: for each roll call, voting is described by a logistic regression on House members' (unknown) ideological positions.

One-dimensional model:

$$\mathrm{logit}(\mu_{rm}) = \alpha_r + \beta_r \gamma_m$$

Two dimensions:

$$\mathrm{logit}(\mu_{rm}) = \alpha_r + \beta_r^{(1)} \gamma_m^{(1)} + + \beta_r^{(2)} \gamma_m^{(2)}$$

This is a fairly large dataset/model: there are 439 House members.

Some members can be discarded as 'uninformative' (voted on fewer than 10 roll calls, or always voted the same way).

First, set up the data for our modelling:

```
data(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, ]
parties <- House2001$party[informative]
## 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)
```


For a large model such as this, we need good starting values. The utility function residSVD can be used to decompose multiplicatively the residuals from a smaller model:

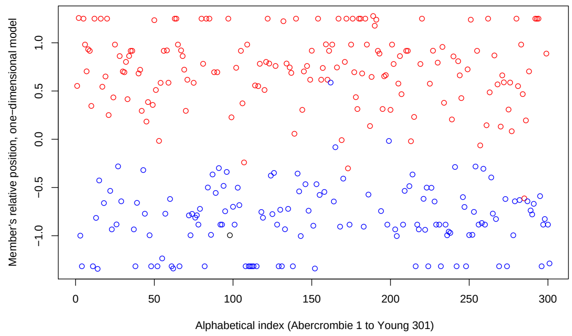
```
baseModel <- glm(vote ~ -1 + rollCall,
                 family = binomial, data = House2001f)
Start <- residSVD(baseModel, rollCall, member)
```

We will now fit the one-dimensional model. First, though, we apply some ‘flattening’ to the response variable, to reduce bias and avert numerical difficulties: 0 becomes 0.03 and 1 becomes 0.97.

```
voteAdj <- 0.5 + 0.94*(House2001f$vote - 0.5)
House2001model1 <- gnm(voteAdj ~ Mult(rollCall - 1, member - 1),
              eliminate = ~ rollCall,
              family = binomial, data = House2001f,
              na.action = na.exclude, trace = TRUE, tolerance = 1e-03,
              start = -Start)
## Deviance is 2234.847, df = 5574
```


We can plot the members’ estimated positions, coloured by party membership:

```
> partyColors <- rep("black", length(parties))
> partyColors <- ifelse(parties == "D", "red", partyColors)
> partyColors <- ifelse(parties == "R", "blue", partyColors)
##  Now make the graph
> plot(coef(House2001model1)[21:321], col = partyColors,
    xlab = "Alphabetical index (Abercrombie 1 to Young 301)",
    ylab = "Member's relative position, one-dimensional model")
```

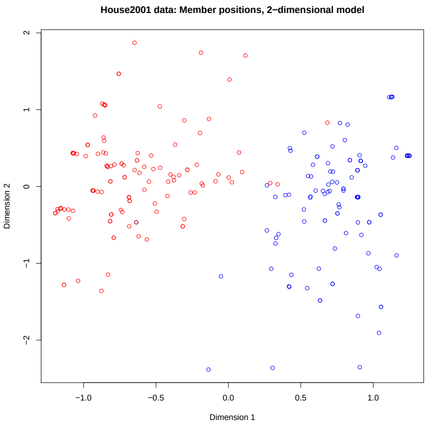



Roll call 12 appears to be different from the rest:

```
> coef(House2001model1)[1:20]
 Mult1.Factor1.rollCall1  Mult1.Factor1.rollCall2  Mult1.Factor1.rollCall3
               3.3272688                3.1187364                4.3788873
 Mult1.Factor1.rollCall4  Mult1.Factor1.rollCall5  Mult1.Factor1.rollCall6
               3.9046925                3.5438839                2.0740049
 Mult1.Factor1.rollCall7  Mult1.Factor1.rollCall8  Mult1.Factor1.rollCall9
               2.6443166                3.5397983                3.9401251
Mult1.Factor1.rollCall10 Mult1.Factor1.rollCall11 Mult1.Factor1.rollCall12
               1.5309173                4.1769722               -0.5547739
Mult1.Factor1.rollCall13 Mult1.Factor1.rollCall14 Mult1.Factor1.rollCall15
               2.0614189                4.3679824                4.4400554
Mult1.Factor1.rollCall16 Mult1.Factor1.rollCall17 Mult1.Factor1.rollCall18
               2.3971024                2.1049206                4.1963022
Mult1.Factor1.rollCall19 Mult1.Factor1.rollCall20
               4.2155210                2.0440617
```

Maybe there’s a second dimension?


```
Start2 <- residSVD(baseModel, rollCall, member, d = 2)
> House2001model2 <- gnm(
              voteAdj ~ Mult(rollCall - 1, member - 1, multiplicity = 2),
              eliminate = ~ rollCall,
              family = binomial, data = House2001f,
              na.action = na.exclude, trace = TRUE, tolerance = 1e-03,
              start = Start2)
## Deviance is 1545.166, df = 5257
```



Remarks:

1. The logistic regression model used here is an example of a *Rasch model* ('item response theory')

2. Probit gives indistinguishably similar results.

3. As is evident from the results of this small study, the choice of 'items' is crucial to the results of scaling.

4. Factor analysis would be another way to explore this, and to scale the members (using factor scores). But factor analysis gives quite different results, on account of the assumption that the unobserved positions are normally distributed.

Exercise: Diagonal reference model for data from Clifford and Heath

1. Load the `voting` data. This is a data frame of the percentage voting Labour (`percentage`) and the total number of people (`total`) in groups classified by 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 sucesses 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. Fit a diagonal reference model to these data (see p57), using `yvar` as the response, and `family = binomial`. Use `summary` to summarise the result. Evaluate the weights for the `origin` and `destination` diagonal effects, as on p58.

3. It could be that individuals which have come into or out of the salariat (class 1) vote differently from other individuals. We can define factors indicating movement in and out of class 1 as follows:

```
in <- with(voting, origin != 1 & destination == 1)
out <- with(voting, origin == 1 & destination != 1)
```

Re-fit the diagonal reference model, specifying `~ 1 + in + out` as the `formula` argument of `Dref`, so the weights are parameterised by a main effect with additional effects for `in` and `out`. Assign these effects to `base`, `in.adj` and `out.adj`. See if the fit of the model has improved.

4. Evaluate the weights for the different groups of people as below:

```
in.to.1 <- prop.table(exp(in.adj + base))
out.of.1 <- prop.table(exp(out.adj + base))
other <- prop.table(exp(base))
```

5. 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, and compare the results.

Exercise: Scaling of the US House of Representatives

We will re-run the scaling analysis of the House2001 data, with roll call 12 removed.

1. First, run through the one-dimensional scaling as shown in the lecture, by running `example(House2001)`.

2. Now re-do the analysis with roll call 12 removed from the data. (Do this by modifying commands copied and pasted from the examples in ?House2001.)

```
> House2001m <- House2001m[, -12]
> House2001v <- as.vector(House2001m)
> House2001f <- data.frame(member = rownames(House2001m),
                          party = parties,
                          rollCall = factor(rep((1:20)[-12],
                                rep(nrow(House2001m), 19))),
                          vote = House2001v)
```

```
> voteAdj <- 0.5 + 0.94*(House2001f$vote - 0.5)
> baseModel <- glm(voteAdj ~ -1 + rollCall,
    family = binomial, data = House2001f)
> Start <- residSVD(baseModel, rollCall, member)
> House2001model1 <- gnm(
              voteAdj ~ Mult(rollCall - 1, member - 1),
              eliminate = ~ rollCall,
              family = binomial, data = House2001f,
              na.action = na.exclude, trace = TRUE,
              tolerance = 1e-03,
              start = -Start)
```

3. Look at the slope coefficients for the 19 roll calls:

```
> coef(House2001model1)[1:19]
```


4. Finally, graph the members' estimated ideological positions:

```
> positions <- coef(House2001model1)[20:320]
> plot(positions, col = partyColors)
```

and find graphically the names of the more liberal among the Republicans, and the more conservative among the Democrats:

```
> identify(1:301, positions,
    labels = rownames(House2001m))
```

(right-click to stop the identify mechanism).