Introduction to Generalized Nonlinear Models

Heather Turner

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

Zurich, 2015–05–11
Updated for package changes 2018–06–20

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.

Plan

Getting Started

Association Models

Other Multiplicative Models

Other Specialized Models

Section 1

Getting Started

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.

A natural generalization is to allow the *data* to 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.

A natural generalization is to allow the *data* to 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. heterogeneous row and column scores

$$\phi_r \psi_c$$

are specified as Mult(row, col).

Other basic `"nonlin"` functions include `Exp` and `Inv`.

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').

A simple example:

$$\phi_r \psi_c$$

is equivalent to

$$\phi_r^* \psi_c^* = (2\phi_r)(\psi_c/2)$$

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').

A simple example:

$$\phi_r \psi_c$$

is equivalent to

$$\phi_r^* \psi_c^* = (2\phi_r)(\psi_c/2)$$

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').

A simple example:

$$\phi_r \psi_c$$

is equivalent to

$$\phi_r^* \psi_c^* = (2\phi_r)(\psi_c/2)$$

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 p13), 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 p15, p18). 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.

Section 2

Association Models

RC(M) Models

The row-column association models introduced in Section I are special cases of the RC(M) model:

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

We will use this class of models to further explore the issues of identifiability, parameterization and obtaining standard errors.

RC(1) with Homogeneous Scores

For the RC(1) model with homogeneous scores, we have

$$\alpha_r + \beta_c + \phi_r \phi_c = -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k)$$
$$= \alpha_r^* + \beta_c^* + \phi_r^* \phi_c^*$$

Constraining one of the ϕ_r to a constant fixes the location of the
homogeneous scores so the parameterization is unique.

Let $k = -\phi_1$, then

$$\phi_1^* = \phi_1 - \phi_1 = 0$$
$$\phi_r^* = \phi_r - \phi_1 \qquad r \neq 1$$

So constraining one parameter to zero is equivalent to fitting the
unconstrained model, then estimating simple contrasts.

RC(1) with Homogeneous Scores

For the RC(1) model with homogeneous scores, we have

$$\alpha_r + \beta_c + \phi_r\phi_c = -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k)$$
$$= \alpha_r^* + \beta_c^* + \phi_r^*\phi_c^*$$

Constraining one of the ϕ_r to a constant fixes the location of the homogeneous scores so the parameterization is unique.

Let $k = -\phi_1$, then

$$\phi_1^* = \phi_1 - \phi_1 = 0$$
$$\phi_r^* = \phi_r - \phi_1 \qquad r \neq 1$$

So constraining one parameter to zero is equivalent to fitting the unconstrained model, then estimating simple contrasts.

RC(1) with Homogeneous Scores

For the RC(1) model with homogeneous scores, we have

$$\alpha_r + \beta_c + \phi_r\phi_c = -k^2 + (\alpha_r - k\phi_r) + (\beta_c - k\phi_c) + (\phi_r + k)(\phi_c + k)$$
$$= \alpha_r^* + \beta_c^* + \phi_r^*\phi_c^*$$

Constraining one of the ϕ_r to a constant fixes the location of the homogeneous scores so the parameterization is unique.

Let $k = -\phi_1$, then

$$\phi_1^* = \phi_1 - \phi_1 = 0$$
$$\phi_r^* = \phi_r - \phi_1 \qquad r \neq 1$$

So constraining one parameter to zero is equivalent to fitting the unconstrained model, then estimating simple contrasts.

Example: Occupational Status Data

As in Practical I, we fit a homogeneous row-column association model,
separating out the diagonal effects, then use getContrasts

```
RCh <- gnm(Freq ~ origin + destination + Diag(origin, destination) +
              MultHomog(origin, destination), family = poisson,
          data = occupationalStatus, verbose = FALSE)
getContrasts(RCh, pickCoef(RCh, "MultHomog"))

##                                    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
```

Standard errors will change depending on the reference category:

```
getContrasts(RCh, pickCoef(RCh, "MultHomog"), ref = "last")

##                                  estimate     SE quasiSE quasiVar
## MultHomog(origin, destination)1    -2.589  0.189  0.1573  0.02473
## MultHomog(origin, destination)2    -2.371  0.158  0.1190  0.01416
## MultHomog(origin, destination)3    -1.773  0.120  0.0611  0.00374
## MultHomog(origin, destination)4    -1.189  0.116  0.0518  0.00269
## MultHomog(origin, destination)5    -1.171  0.131  0.0798  0.00637
## MultHomog(origin, destination)6    -0.660  0.116  0.0360  0.00129
## MultHomog(origin, destination)7    -0.244  0.158  0.0796  0.00634
## MultHomog(origin, destination)8     0.000  0.000  0.1095  0.01200
```

Quasi standard errors allow the calculation of an approximate standard error for *any* contrast; they are independent of the choice of reference category. For the theory see Firth and de Menezes (2004).

RC(1) with Heterogeneous Scores

Now since, for example,

$$\alpha_r + \beta_r + \phi_r\psi_c = \alpha_r + (\beta_r - \psi_c) + (\phi_r + 1)\psi_c$$
$$= \alpha_r + \beta_r + (2\phi_r)(\psi_c/2)$$

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(1) with Heterogeneous Scores

Now since, for example,

$$\alpha_r + \beta_r + \phi_r\psi_c = \alpha_r + (\beta_r - \psi_c) + (\phi_r + 1)\psi_c$$
$$= \alpha_r + \beta_r + (2\phi_r)(\psi_c/2)$$

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 (?).

```
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(1) 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 (?).

```
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(1) model

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

We fit the RC(1) 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
##                 -0.38132                  -0.38444
##        Mult(., MHS).SESC         Mult(., MHS).SESD
##                 -0.12790                   0.00821
##        Mult(., MHS).SESE         Mult(., MHS).SESF
##                  0.34440                   0.62060
##     Mult(SES, .).MHSwell       Mult(SES, .).MHSmild
##                 -0.82733                  -0.07881
## Mult(SES, .).MHSmoderate Mult(SES, .).MHSimpaired
##                  0.05627                   0.67798
```

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.

RC(2), RC(3), . . .

Additional multiplicative terms can be added via `instances`, e.g.

```
RC2 <- update(RC, count ~ SES + MHS + instances(Mult(SES, MHS), 2))
```

The sum of the multiplicative terms is a low rank approximation to the
full interaction matrix. For the parameters to be identified, the
multiplicative terms must be orthogonal to each other. This means
constraining the rotation as well as the location and scale.

Decomposing a matrix into a sum of separable matrices (where a
separable matrix can be written as the outer product of two vectors) is
equivalent to singular value decomposition. Incorporating marginal
probability weights is tricky. Furthermore, standard errors can not be
directly computed. The **logmult** package helps with both these issues.

RC(2), RC(3), . . .

Additional multiplicative terms can be added via `instances`, e.g.

```
RC2 <- update(RC, count ~ SES + MHS + instances(Mult(SES, MHS), 2))
```

The sum of the multiplicative terms is a low rank approximation to the full interaction matrix. For the parameters to be identified, the multiplicative terms must be orthogonal to each other. This means constraining the rotation as well as the location and scale.

Decomposing a matrix into a sum of separable matrices (where a separable matrix can be written as the outer product of two vectors) is equivalent to singular value decomposition. Incorporating marginal probability weights is tricky. Furthermore, standard errors can not be directly computed. The **logmult** package helps with both these issues.

RC(2), RC(3), . . .

Additional multiplicative terms can be added via `instances`, e.g.

```
RC2 <- update(RC, count ~ SES + MHS + instances(Mult(SES, MHS), 2))
```

The sum of the multiplicative terms is a low rank approximation to the full interaction matrix. For the parameters to be identified, the multiplicative terms must be orthogonal to each other. This means constraining the rotation as well as the location and scale.

Decomposing a matrix into a sum of separable matrices (where a separable matrix can be written as the outer product of two vectors) is equivalent to singular value decomposition. Incorporating marginal probability weights is tricky. Furthermore, standard errors can not be directly computed. The **logmult** package helps with both these issues.

Introduction to the logmult package

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

Particular features include

- ▶ wrapper functions for fitting common models, including RC(M)
- ▶ specialized print and plot methods
- ▶ jackknife and bootstrap standard errors

Example: Mental Health Data Revisited

```
MHtab <- xtabs(count ~ SES + MHS, data = mentalHealth)
rc(MHtab, verbose = FALSE)

## Call:
## rc(tab = MHtab, verbose = FALSE)
##
## Intrinsic association coefficients:
##  Dim1
## 0.166
##
## Normalized row scores:
##       A      B      C      D      E      F
## 1.112  1.121  0.371 -0.027 -1.010 -1.818
##
## Normalized column scores:
##     well    mild  moderate  impaired
##    1.678    0.140   -0.137    -1.414
##
## Normalization weights: marginal
```
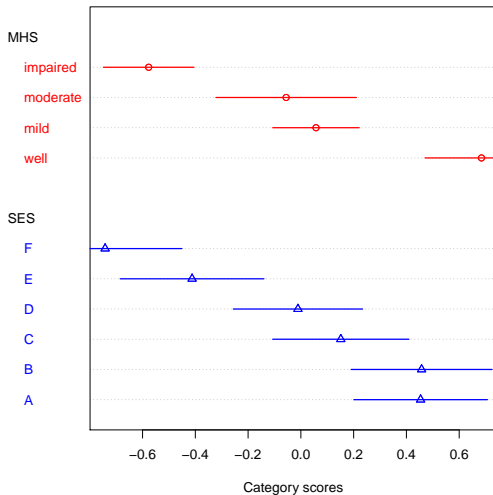
```
RC <- rc(MHtab, se = "jackknife", verbose = FALSE, ncpus = 1)
plot(RC, conf.int = 0.95)
```

```
RC2 <- rc(MHtab, nd = 2, se = "jackknife", verbose = FALSE, ncpus = 1)
plot(RC2, conf.int = 0.95)


## Loading required namespace:  ellipse
```
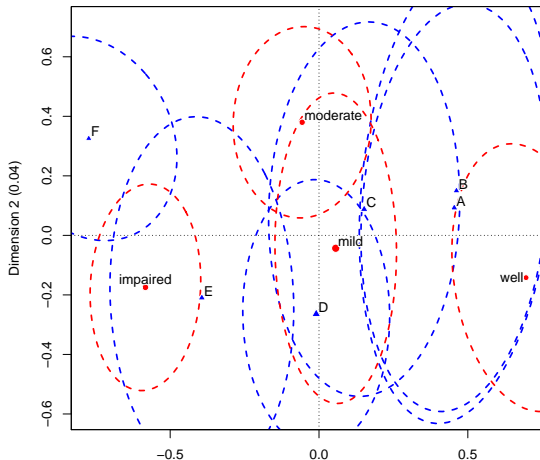
Analysis of Association

An analysis of association compares successive RC(M) models to the independence model.

```
anoas(MHtab, nd=2, verbose = FALSE)

## Fitting independence model...
## Fitting model with 1 dimension...
## Fitting model with 2 dimensions...
##         Res. Df Res. Dev Dev./Indep. (%) Dissim. (%) BIC AIC Dev. Df
## Indep.       15     47.4             100         5.5 -64  17  NA NA
## RC(1)         8      3.6               8         1.8 -56 -12 -44 -7
## RC(2)         3      0.5               1         0.7 -22  -5  -3 -5
```

Other Multiplicative Association Models

The **logmult** package also provides facilities for the following:

- ▶ Models with skew-symmetric terms, either `HMSkew`:

$$\nu_r \omega_c - \omega_r \nu_c$$

  or `YMSkew`:

$$\delta_{r<c}\, \omega_r(\omega_c - \omega_r) - \delta_{c>r}\, \omega_c(\omega_r - \omega_c)$$

  which can replace or supplement RC(M) association terms.

- ▶ RC(M)-L models, an extension of RC(M) to three-way tables
- ▶ The UNIDIFF model, also for three-way tables

Of these, the UNIDIFF model is most commonly used in practice.

UNIDIFF Model

UNIDIFF models postulate a simplified three-way interaction as follows:

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

This implies a common pattern of log ratios, modulated by a positive constant that is specific to the two-way table at each value of t.

This model can be specified in **gnm** via

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

Interest focuses of the γ_t parameters, for which simple contrasts are estimable.

UNIDIFF Model

UNIDIFF models postulate a simplified three-way interaction as follows:

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

This implies a common pattern of log ratios, modulated by a positive constant that is specific to the two-way table at each value of t.

This model can be specified in **gnm** via

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

Interest focuses of the γ_t parameters, for which simple contrasts are estimable.

UNIDIFF Model

UNIDIFF models postulate a simplified three-way interaction as follows:

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

This implies a common pattern of log ratios, modulated by a positive constant that is specific to the two-way table at each value of t.

This model can be specified in **gnm** via

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

Interest focuses of the γ_t parameters, for which simple contrasts are estimable.

Practical II

1. The `yaish` data set from **gnm** is a 3-way contingency table classified by father's social class (`orig`), son's social class (`dest`) and son's education (`educ`). Remove the last level of `dest` (due to low counts), then put educ as the third dimension (as required by `unidiff`).

```
yaish <- as.table(yaish[,,-7])
yaish <- aperm(yaish, c("orig", "dest", "educ"))
```

2. Fit the UNIDIFF model using the `unidiff` function from **logmult** and print the result. The *layer coefficients* are $\exp(\gamma_t)$, with γ_1 constrained to zero. The two-way interaction has been constrained s.t.

$$\sum_r \delta_{rc} \pi_r = \sum_c \delta_{rc} \pi_c = 0$$

where π_r and π_c are the marginal probabilities, factoring out an intrinsic association coefficient ϕ. The *layer intrinsic association coefficients* are $\phi \exp(\gamma_t)$.

3. Plot the layer coefficients with confidence intervals, using

```
plot(model, se.type = "se")
```

This computes confidence intervals for the γ_t parameters, then exponentiates to give the displayed result.

4. Re-fit the UNIDIFF model using gnm, setting γ_1 to zero with constrain = "[.]educ1" (see p40). Use pickCoef to select all the γ_t parameters. Exponentiate the result to obtain the layer coefficients returned by unidiff.

5. The confidence intervals produced in step 3 assume that the likelihood surface is approximately quadratic around the maximum likelihood solution. We can check if this is the case by profiling the likelihood. Use `profile` to profile the γ_t parameters, assigning the result. Plot this result using `plot`: if the likelihood surface is quadratic, the profile plot will show a straight line.

6. Obtain confidence intervals based on the profile likelihood by passing the profile object to `confint`, assigning the result. Re-plot the layer coefficients as in step 3. Add the confidence intervals based on profile likelihood via

```
segments(1:5+0.1, exp(conf[,1]), 1:5+0.1, exp(conf[,2]), col = "red")
```

and compare.

Section 3

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.

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.

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:

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

Poisson Trick

The stereotype model can be fitted as a GNM by re-expressing the categorical data as category counts $Y_i = (Y_{i1}, \ldots, Y_{ik})$.

Assuming a Poisson distribution for Y_{ic}, the joint distribution of Y_i is Multinomial$(N_i, p_{i1}, \ldots, p_{ik})$ conditional on the total count N_i.

The expected counts are then $\mu_{ic} = N_i p_{ic}$ and the parameters of the stereotype model can be estimated through fitting

$$\begin{aligned}
\log \mu_{ic} &= \log(N_i) + \log(p_{ic}) \\
&= \alpha_i + \beta_{0c} + \gamma_c \sum_r \beta_r x_{ir}
\end{aligned}$$

where the "nuisance" parameters α_i ensure that the multinomial denominators are reproduced exactly, as required.

Poisson Trick

The stereotype model can be fitted as a GNM by re-expressing the categorical data as category counts $Y_i = (Y_{i1}, \ldots, Y_{ik})$.

Assuming a Poisson distribution for Y_{ic}, the joint distribution of Y_i is Multinomial$(N_i, p_{i1}, \ldots, p_{ik})$ conditional on the total count N_i.

The expected counts are then $\mu_{ic} = N_i p_{ic}$ and the parameters of the stereotype model can be estimated through fitting

$$\log \mu_{ic} = \log(N_i) + \log(p_{ic})$$
$$= \alpha_i + \beta_{0c} + \gamma_c \sum_r \beta_r x_{ir}$$

where the "nuisance" parameters α_i ensure that the multinomial denominators are reproduced exactly, as required.

Poisson Trick

The stereotype model can be fitted as a GNM by re-expressing the categorical data as category counts $Y_i = (Y_{i1}, \ldots, Y_{ik})$.

Assuming a Poisson distribution for Y_{ic}, the joint distribution of Y_i is Multinomial$(N_i, p_{i1}, \ldots, p_{ik})$ conditional on the total count N_i.

The expected counts are then $\mu_{ic} = N_i p_{ic}$ and the parameters of the stereotype model can be estimated through fitting

$$\begin{aligned}
\log \mu_{ic} &= \log(N_i) + \log(p_{ic}) \\
&= \alpha_i + \beta_{0c} + \gamma_c \sum_r \beta_r x_{ir}
\end{aligned}$$

where the "nuisance" parameters α_i ensure that the multinomial denominators are reproduced exactly, as required.

Example: Back Pain Data

The backPain data are from an example in Anderson (1984). For 101
patients, 3 prognostic variables are recorded at baseline and their level of
back pain is recorded again after 3 weeks.

```
backPain[1:5,]

##   x1 x2 x3                pain
## 1  1  1  1                same
## 2  1  1  1 marked.improvement
## 3  1  1  1    complete.relief
## 4  1  2  1                same
## 5  1  2  1 slight.improvement
```

The function expandCategorical converts the categorical data into sets of counts, by default grouping individuals with common covariates.

```
backPainLong <- expandCategorical(backPain, "pain", group = TRUE)
head(backPainLong)

##   x1 x2 x3                  pain id count
## 1  1  1  1                 worse  1     0
## 2  1  1  1                  same  1     1
## 3  1  1  1    slight.improvement  1     0
## 4  1  1  1 moderate.improvement  1     0
## 5  1  1  1   marked.improvement  1     2
## 6  1  1  1      complete.relief  1     4
```

Grouping individuals reduces the size of the data set and does not affect comparisons between models.

The function expandCategorical converts the categorical data into sets of counts, by default grouping individuals with common covariates.

```
backPainLong <- expandCategorical(backPain, "pain", group = TRUE)
head(backPainLong)

## x1 x2 x3                      pain id count
## 1 1 1 1                      worse  1     0
## 2 1 1 1                       same  1     1
## 3 1 1 1     slight.improvement  1     0
## 4 1 1 1 moderate.improvement  1     0
## 5 1 1 1   marked.improvement  1     2
## 6 1 1 1       complete.relief  1     4
```

Grouping individuals reduces the size of the data set and does not affect comparisons between models.

The stereotype model can be fitted to these data as follows

```
stereotype <- gnm(count ~ pain + Mult(pain, x1 + x2 + x3),
                  eliminate = id, family = poisson,
                  data = backPainLong, verbose = FALSE)
```

The eliminate argument of gnm is used to specify that the id
parameters replace the intercept in the model. This has two benefits:

- gnm exploits the structure of these parameters to estimate them
  more efficiently
- these nuisance parameters are excluded from summaries of the
  model object

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

```
logistic <- gnm(count ~ pain + pain:(x1 + x2 + x3),
                eliminate = id, family = poisson, data = backPainLong)
anova(stereotype, logistic)

## Analysis of Deviance Table
##
## Model 1: count ~ pain + Mult(pain, x1 + x2 + x3) - 1
## Model 2: count ~ pain + pain:x1 + pain:x2 + pain:x3 - 1
##   Resid. Df Resid. Dev Df Deviance
## 1        48       55.9
## 2        40       51.8  8     4.09
```

In order to make the category-specific multipliers identifiable, we must constrain both the location and scale.

One way to do this is to set the last multiplier to one and fix the coefficient of the first covariate to one. We can do this using offset along with the constrain and constrainTo arguments of gnm:

```
stereotype <- update(stereotype,
                     . ~ pain + Mult(pain, offset(x1) + x2 + x3),
                     constrain = "[.]paincomplete.relief",
                     constrainTo = 1)
```

We can (re-)define the coefficients of interest after fitting the model

```
ofInterest(stereotype) <- pickCoef(stereotype, "Mult")
```

```
## Mult(., x2 + x3 + offset(x1)).painworse, Mult(., x2 + x3 +
## offset(x1)).painsame, Mult(., x2 + x3 +
## offset(x1)).painslight.improvement, Mult(., x2 + x3 +
## offset(x1)).painmoderate.improvement, Mult(., x2 + x3 +
## offset(x1)).painmarked.improvement, Mult(., x2 + x3 +
## offset(x1)).paincomplete.relief, Mult(pain, . + x3 +
```

In order to make the category-specific multipliers identifiable, we must
constrain both the location and scale.

One way to do this is to set the last multiplier to one and fix the
coefficient of the first covariate to one. We can do this using `offset`
along with the `constrain` and `constrainTo` arguments of `gnm`:

```
stereotype <- update(stereotype,
                     . ~ pain + Mult(pain, offset(x1) + x2 + x3),
                     constrain = "[.]paincomplete.relief",
                     constrainTo = 1)
```

We can (re-)define the coefficients of interest after fitting the model

```
ofInterest(stereotype) <- pickCoef(stereotype, "Mult")

## Mult(., x2 + x3 + offset(x1)).painworse, Mult(., x2 + x3 +
## offset(x1)).painsame, Mult(., x2 + x3 +
## offset(x1)).painslight.improvement, Mult(., x2 + x3 +
## offset(x1)).painmoderate.improvement, Mult(., x2 + x3 +
## offset(x1)).painmarked.improvement, Mult(., x2 + x3 +
## offset(x1)).paincomplete.relief, Mult(pain, . + x3 +
```

```
parameters(stereotype)

## Coefficients of interest:
##               Mult(., x2 + x3 + offset(x1)).painworse
##                                                  6.372
##                 Mult(., x2 + x3 + offset(x1)).painsame
##                                                  2.662
##    Mult(., x2 + x3 + offset(x1)).painslight.improvement
##                                                  2.862
## Mult(., x2 + x3 + offset(x1)).painmoderate.improvement
##                                                  3.739
##    Mult(., x2 + x3 + offset(x1)).painmarked.improvement
##                                                  1.760
##      Mult(., x2 + x3 + offset(x1)).paincomplete.relief
##                                                  1.000
##                    Mult(pain, . + x3 + offset(x1)).x2
##                                                  0.574
##                    Mult(pain, x2 + . + offset(x1)).x3
##                                                  0.505
```

Is the scale of the relationship with the covariates really different for each
category?

Consider common multiplier for `same` and `slight.improvement`

```
.pain <- backPainLong$pain
levels(.pain)[2:3] <- paste(levels(.pain)[2:3], collapse = " | ")
stereotype5 <- update(stereotype,
                        ~ pain + Mult(.pain, x1 + x2 + x3))
anova(stereotype, stereotype5)

## Analysis of Deviance Table
##
## Model 1: count ~ pain + Mult(pain, offset(x1) + x2 + x3) - 1
## Model 2: count ~ pain + Mult(.pain, x1 + x2 + x3) - 1
##   Resid. Df Resid. Dev Df Deviance
## 1        48       55.9
## 2        49       56.0 -1   -0.118
```

In fact, only three different multipliers are necessary

```
## Analysis of Deviance Table
##
## Model 1: count ~ pain + Mult(pain, offset(x1) + x2 + x3) - 1
## Model 2: count ~ pain + Mult(.pain, x1 + x2 + x3) - 1
## Model 3: count ~ pain + Mult(.pain, x1 + x2 + x3) - 1
## Model 4: count ~ pain + Mult(.pain, x1 + x2 + x3) - 1
## Model 5: count ~ pain + Mult(.pain, x1 + x2 + x3) - 1
##   Resid. Df Resid. Dev Df Deviance
## 1        48       55.9
## 2        49       56.0 -1    -0.12
## 3        50       58.4 -1    -2.35
## 4        51       61.6 -1    -3.19
## 5        52       83.4 -1   -21.79
```

Rasch Models

Rasch models are used in Item Response Theory to model the binary responses of *subjects* over a set of *items*.

The simplest one parameter logistic (1PL) model has the form

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \gamma_s$$

The one-dimensional Rasch model extends the 1PL as follows:

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \beta_i \gamma_s$$

where β_i measures the discrimination of item i: the larger β_i the steeper the item-response function that maps γ_s to π_{is}.

Rasch Models

Rasch models are used in Item Response Theory to model the binary responses of *subjects* over a set of *items*.

The simplest one parameter logistic (1PL) model has the form

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \gamma_s$$

The one-dimensional Rasch model extends the 1PL as follows:

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \beta_i \gamma_s$$

where β_i measures the discrimination of item i: the larger β_i the steeper the item-response function that maps γ_s to π_{is}.

Rasch Models

Rasch models are used in Item Response Theory to model the binary responses of *subjects* over a set of *items*.

The simplest one parameter logistic (1PL) model has the form

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \gamma_s$$

The one-dimensional Rasch model extends the 1PL as follows:

$$\log \frac{\pi_{is}}{1 - \pi_{is}} = \alpha_i + \beta_i \gamma_s$$

where β_i measures the discrimination of item i: the larger β_i the steeper the item-response function that maps γ_s to π_{is}.

Example: US House of Representatives

Votes on 20 roll calls selected by Americans for Democratic Action (ADA)

The data are prepared for modelling as shown in `?House2001`,
specifically

- ▶ removing uninformative House members (too many `NA`s)
- ▶ creating a data frame with factors `member` and `rollCall`.

For representatives that always vote "For" or "Against" the ASA position,
maximum likelihood will produce infinite γ_s estimates, so that the fitted
probabilities are 0 or 1 (*complete separation*).

To mitigate this, the response is "flattened": 0 becomes 0.03 and 1
becomes 0.97.

The data are prepared for modelling as shown in `?House2001`, specifically

- ▶ removing uninformative House members (too many `NA`s)
- ▶ creating a data frame with factors `member` and `rollCall`.

For representatives that always vote "For" or "Against" the ASA position, maximum likelihood will produce infinite γ_s estimates, so that the fitted probabilities are 0 or 1 (*complete separation*).

To mitigate this, the response is "flattened": 0 becomes 0.03 and 1 becomes 0.97.

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)
```

The one-dimensional Rasch model is then fitted via

```
rasch1 <- gnm(voteAdj ~ Mult(rollCall, member),
              eliminate = rollCall,
              family = binomial, data = House2001f,
              na.action = na.exclude, tolerance = 1e-03,
              start = -Start, verbose = FALSE)
```

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)
```

The one-dimensional Rasch model is then fitted via

```
rasch1 <- gnm(voteAdj ~ Mult(rollCall, member),
              eliminate = rollCall,
              family = binomial, data = House2001f,
              na.action = na.exclude, tolerance = 1e-03,
              start = -Start, verbose = FALSE)
```

```
plot(pickCoef(rasch1, "[.]member", value = TRUE),
     col = c("red", "black", "black", "blue")[parties],
     xlab = "Alphabetical index", ylab = "Member's relative position")
dotchart(pickCoef(rasch1, "[.]rollCall", value = TRUE),
         paste0("Roll call ", 1:20))
```
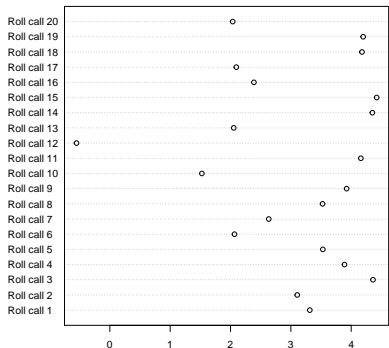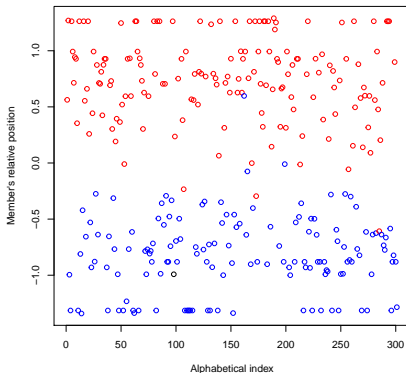
Lee-Carter model for mortality trends

For the study and projection of age-specific population mortality rates, Lee and Carter (1992) proposed a model that has been the basis of many subsequent analyses.

Suppose that death count D_{ay} for individuals of age a in year y has mean μ_{ay} and *quasi-Poisson* variance $\phi\mu_{ay}$.

Lee-Carter model:

$$\log(\mu_{ay}/e_{ay}) = \alpha_a + \beta_a\gamma_y\,,$$
$$\Rightarrow \log(\mu_{ay}) = \log(e_{ay}) + \alpha_a + \beta_a\gamma_y$$

where e_{ay} is the *exposure* (number of lives at risk).

The Lee-Carter model can be fitted with `gnm` as follows

```
LCmodel <- gnm(Deaths ~ Mult(Exp(Age), Year),
               eliminate = Age, offset = log(Exposure),
               family = "quasipoisson")
```

where

- `Exp(Age)` is used to constrain the parameters representing the "sensitivity" of age group a to have the same sign,
- `Age` is "eliminated" since it will typically have many levels,
- `offset` is used to add the log exposure with a coefficient of 1,
- the quasipoisson family is used so that the dispersion parameter ϕ is estimated rather than fixed to 1.

Practical III

1. Use `read.table` to read the file `Canada.txt` provided on your USB
stick: data from the Human Mortality Database on male deaths in
Canada between 1921 and 2003. Convert `Year` and `Age` to factors and fit
a simplified version of the Lee-Carter model in which the multiplicative
term is replaced by a linear `Year` effect. Eliminate `Age` as on p62.

2. Extract the eliminated age coefficients via

```
AgeCoef <- attr(coef(model1), "eliminated")
```

Fit the Lee-Carter model using `update`, specifying the new formula and

```
start = c(AgeCoef, rep(0, length(AgeCoef)), 0, coef(model1))
```

This starts the gnm with the linear effects of age set to their estimates
from the first model, the age multipliers set to one $(= \exp(0))$, and the
year multipliers set to their linear effects from the first model, including
the effect for the first year which was constrained to zero by default.

3. Use `deviance` and `df.residual` to compare the deviance of the Lee-Carter model relative to the degrees of freedom. You should notice severe overdispersion (deviance ≫ df).

4. Use `residuals` to obtain the Pearson residuals from the Lee-Carter model. Plot the residuals vs. `Age`. The overdispersion is not evenly spread through the data, but is largely concentrated in two age groups, roughly ages 25–35 and 50–65. Plot the residuals vs. `Year` for each of these subgroups, in separate windows. There is a clear (and roughly cancelling) dependence on year, indicating that the assumed bilinear interaction between age and year does not hold for the full range of ages and years considered here.

5. Use `update` to re-fit the Lee-Carter model for males aged 45 or over. Look at the deviance and degrees of freedom: the over-dispersion should be much reduced. Plot the residuals against `Age` and `Year` to look for departures from the assumed bilinear structure.

6. Use `getContrasts` to compute quasi-standard errors for the logarithms of β_a – this takes several seconds! Use `plot` to plot the resulting qv object, using `levelNames` = 45:98 to name the levels (the parameter for age 99 is unestimable). Consider the expected and unexpected features of this plot.

Section 4

Other Specialized Models

Other Specialized Models

We have already seen an example of a specialized `"nonlin"` term,
`MultHomog`. A specialized term is required if it

- requires common parameters to be estimated across different factors,
- cannot be expressed as a function of existing `"nonlin"` terms.

In this section we look at the other specialized `"nonlin"` term provided
by **gnm**, `Dref`, and some examples of custom **"nonlin"** functions
encountered in practice.

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.

The weights are constrained to be non-negative and to sum to one so that γ_l is the value for observations with level l across all the factors.

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

Dref specifies the constraints on the weights by defining them as

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

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.

The weights are constrained to be non-negative and to sum to one so that γ_l is the value for observations with level l across all the factors. Unlike the GNMs models considered so far, which structure interaction terms, this structures the main effects of the corresponding factors.

Dref specifies the constraints on the weights by defining them as

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

Example: 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:

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

Note that including an intercept in the model would require one of the diagonal effects to be set to zero for identifiability – the intercept would then be γ_1 and the other diagonal effects would be $\gamma_r - \gamma_1$.

If the model included another factor, say nationality, a reference level must be set, e.g.

  ▶ setting first level of nationality to zero

  ▶ setting first diagonal effect to zero

  ▶ adding intercept and setting first diagonal effect to zero

Basic diagonal reference model for MCFM:

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

Note that including an intercept in the model would require one of the diagonal effects to be set to zero for identifiability – the intercept would then be γ_1 and the other diagonal effects would be $\gamma_r - \gamma_1$.

If the model included another factor, say nationality, a reference level must be set, e.g.

- ▶ setting first level of nationality to zero
- ▶ setting first diagonal effect to zero
- ▶ adding intercept and setting first diagonal effect to zero

```
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

```r
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

```r
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
```

More Complex "nonlin" Terms

The "nonlin" features introduced in the last example will be sufficient in many cases. The **gnm** vignette gives further details on

- ▶ specifying homologous parameters
- ▶ implementing *dot-style* parameter labelling
- ▶ implementing term-specific starting values
- ▶ enabling multiple instances

Practical IVa

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 p71), 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 IVb

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` on your USB stick. 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.

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

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

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

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

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

Sobel, M. E. (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, F. W. P., N. D. de Graaf, and J. R. M. Gerris (2002). Conformity to Parental Rules: Asymmetric Influences of Father's and Mother's Levels of Education. *Europ. Soc. Rev. 18*, 489–502.