Summary

Functions `isUnknown`, `unknownToNA` and `NAToUnknown` provide a useful interface to work with various representations of unknown/missing values. Their use is meant primarily for shaping the data after importing to or before exporting from R. I welcome any comments or suggestions.

Bibliography

R Development Core Team. *R Data Import/Export*, 2006. URL `http://cran.r-project.org/manuals.html`. ISBN 3-900051-10-0. 24

G. R. Warnes. *gdata: Various R programming tools for data manipulation*, 2006. URL `http://cran.r-project.org/src/contrib/Descriptions/gdata.html`. R package version 2.3.1. Includes R source code and/or documentation contributed by Ben Bolker, Gregor Gorjanc and Thomas Lumley. 24

Gregor Gorjanc
University of Ljubljana, Slovenia
`gregor.gorjanc@bfro.uni-lj.si`

A New Package for Fitting Random Effect Models

The npmlreg package

by Jochen Einbeck, John Hinde, and Ross Darnell

Introduction

Random effects have become a standard concept in statistical modelling over the last decades. They enter a wide range of applications by providing a simple tool to account for such problems as model misspecification, unobserved (latent) variables, unobserved heterogeneity, and the like. One of the most important model classes for the use of random effects is the generalized linear model. Aitkin (1999) noted that "the literature on random effects in generalized linear models is now extensive," and this is certainly even more true today.

However, most of the literature and the implemented software on generalized linear mixed models concentrates on a normal random effect distribution. An approach that avoids specifying this distribution parametrically was provided by Aitkin (1996a), using the idea of 'Nonparametric Maximum Likelihood' (NPML) estimation (Laird, 1978). The random effect distribution can be considered as an unknown mixing distribution and the NPML estimate of this is a finite discrete distribution. This can be determined by fitting finite mixture distributions with varying numbers of support points, where each model is conveniently fitted using a straightforward EM algorithm.

This approach is implemented in GLIM4 (Aitkin and Francis, 1995). Despite being a quite powerful tool, the current GLIM-based software is computationally limited and the GLIM system is no longer widely used. Though the alternatives C.A.MAN (Böhning et al., 1992) and the Stata program `gllamm` (Skrondal and Rabe-Hesketh, 2004) cover parts of GLIMs capacities (in the latter case based on Newton-Raphson instead of EM), no R implementation of NPML estimation existed. The package **npmlreg** (Einbeck et al., 2006), which we wish to introduce to the R community in this article, is designed to fill this gap.

NPML estimation

Assume there is given a set of explanatory vectors $x_1, \ldots, x_n$ and a set of observations $y_1, \ldots, y_n$ sampled from an exponential family distribution[1] $f(y_i|\beta, \phi_i)$ with dispersion[2] parameter ϕ_i. In a generalized linear model, predictors and response are assumed to be related through a link function h,

$$\mu_i \equiv E(y_i|\beta, \phi_i) = h(\eta_i) \equiv h(x_i'\beta),$$

and the variance $Var(y_i|\beta, \phi_i) = \phi_i v(\mu_i)$ depends on a function $v(\mu_i)$ which is entirely determined by the choice of the particular exponential family. However, often the actual variance in the data is larger than the variance according to this strict mean-variance relationship. This effect is commonly called overdispersion, reasons for which might be, e.g., correlation in the data or important explanatory variables not included in the model. In order to account for additional unexplained variability of the individual observations, a random effect z_i with density $g(z)$ is in-

[1] In the present implementation, Gaussian, Poisson, Binomial, and Gamma distributed responses are supported

[2] For binomial and Poisson models, $\phi_i \equiv 1$. For Gaussian and Gamma models, the dispersion may be specified as constant, i.e., $\phi_i \equiv \phi$, or as depending on the observation i. The theory in this section is provided for the most general case, i.e., variable dispersion.

cluded into the linear predictor

$$\eta_i = x_i'\beta + z_i.$$

The likelihood can be approximated by a finite mixture

$$L = \prod_{i=1}^{n} \int f(y_i|z_i, \beta, \phi_i) g(z_i)\, dz_i \approx \prod_{i=1}^{n} \left\{ \sum_{k=1}^{K} f_{ik} \pi_k \right\},$$

where $f_{ik} = f(y_i|z_k, \beta, \phi_k)$, z_k are the mass points and π_k their masses. The score equations, obtained by setting the partial derivatives of the log-likelihood $\ell = \log L$ equal to zero,

$$\frac{\partial \ell}{\partial z_k} = 0, \quad \frac{\partial \ell}{\partial \beta} = 0, \quad \frac{\partial \ell}{\partial \phi_k} = 0,$$

turn out to be weighted versions of the single-distribution score equations, with weights $w_{ik} = \pi_k f_{ik} / \sum_l \pi_l f_{il}$.

The weights w_{ik} can be interpreted as posterior probabilities that the observation y_i comes from component k. The score equation for the mixture proportions,

$$\frac{\partial \ell - \lambda(\sum \pi_k - 1)}{\partial \pi_k} = 0,$$

gives the ML estimate $\hat{\pi}_k = \frac{1}{n} \sum_i w_{ik}$, which can be nicely interpreted as the average posterior probability for component k. The parameters β, z_k and π_k can now be simultaneously estimated by the EM algorithm, whereby the missing information is the component membership of the observations:

E-Step Adjust weights $w_{ik} = P(\text{obs. } i \text{ comes from comp. } k)$

M-Step Update parameter estimates fitting a weighted GLM with weights w_{ik}.

As starting values for the EM algorithm one uses Gauss-Hermite integration points and masses. The location of these starting points can be scaled inwards or outwards by means of a tuning parameter `tol`, which is by default set to 0.5.

This procedure, and its straightforward extension to random coefficient models, is implemented in the function `alldist`, while variance component models can be fitted with `allvc`; see Aitkin et al. (2005), pp 474ff and 485ff for details.

The function `alldist`

The main functions of this package are `alldist` and `allvc`, the names of which were adapted from the homonymous macros in GLIM4. The functions can be used in a similar manner to the R function `glm`.

As an example for `alldist`, we consider data from a study on lung cancer mortality presented in Tsutakawa (1985). The data were recorded in the 84 largest Missouri cities from 1972-1981 and give the number of lung cancer deaths of males aged 45-54 as well as the city sizes[3]. The data were analyzed by Tsutakawa (1985) and Aitkin (1996b), both authors considering logit models of type

$$\log \frac{p_i}{1 - p_i} = z_i,$$

where z_i is a random effect associated with the $i-$th city and p_i is its associated mortality rate. While Tsutakawa fitted a Poisson model with a normal random effect, Aitkin opted for a binomial model with an unspecified random effect distribution. We follow Aitkin and will leave the random effect unspecified, but as lung cancer death is a rather rare event (the crude rate does not exceed 0.03 in any of the cities), it seems natural to work with Poisson models. Hence, one can write $\log(p_i) = z_i$, or equivalently, in terms of the means $\mu_i = n_i p_i$,

$$\log(\mu_i) = \log(n_i) + z_i,$$

where the logarithm of the city sizes n_i appears as an offset. The two-point solution is then obtained via the function `alldist`, using the same notation as for a usual `glm` fit, except that the random term and the number of mass points `k=2` have also to be specified. The resulting object (which we name, say, `missouri.np2`) is of class '`glmmNPML`' and its printed output is given by

```
> print(missouri.np2)

Call:  alldist(formula = Deaths ~1, random =
 ~1, family = poisson(link = "log"), data =
 missouri, k = 2, offset = log(Size))

Coefficients:
 MASS1   MASS2
-4.844  -4.232

Mixture proportions:
    MASS1      MASS2
0.8461624  0.1538376
-2 log L:          355.3
```

One minor difference to a `glm` output is that the disparity ($-2 \log L$) is displayed instead of the deviance, but the latter one is immediately obtained via

```
> missouri.np2$dev
 [1] 92.49207,
```

which is slightly better than the deviance 93.10 reported in Aitkin (1996b) for the corresponding two-mass point binomial logit model fitted with GLIM4. As also observed by Aitkin, the disparity does not

[3] The data set `missouri` is part of this R package

fall significantly when increasing the number of mass points further.

Empirical Bayes predictions (generally, $h(x_i'\hat{\beta} + \hat{z}_i)$ for the number of deaths per city can be obtained by the use of `fitted(missouri.np2)`, or equivalently, `missouri.np2$fitted.values`, or equivalently, `predict(missouri.np2, type="response")`. Dividing this quantity by `missouri$Size`, one obtains estimated or 'shrunken' rates which can be compared to the 6th column in Table 4, Aitkin (1996b). The shrunken rates are less variable than the crude rates and hence are useful for small area estimation problems. The posterior probabilities w_{ik} can be obtained from the fitted model (in analogy to the 8th and 9th column of Table 4, Aitkin, 1996b), via the component `$post.prob`. Further, 'posterior intercepts' for the construction of 'league tables' are stored in component `$post.int` — see Sofroniou et al. (2006) for an example of their application.

Methods for most generic functions that are applied to fitted model objects, such as `update()`, `coefficients()`, `residuals()`, `fitted()`, `summary()`, `predict()` and `plot()`, have been defined for the `glmmNPML` class. In some cases (the first four generic functions listed above) the default method is used; in other cases (the last three generics) explicit methods are provided. The `plot()` function offers four different plots: (i) disparity vs. EM iterations; (ii) EM trajectories; (iii) Empirical Bayes predictions vs. true response; and (iv) posterior probabilities against the residuals of the fixed part of the GLM fit in the last EM iteration. Plots (i) and (ii) are generated by default when running `alldist` and are depicted in Fig. 1 for the model fitted above.

One observes that the EM trajectories converge essentially to the fixed part residuals of cities no. 4 and 84 (in Tsutakawa's list), which have populations of 54155 and 22514, respectively, being much larger than the majority of the other cities with only several hundreds of inhabitants (For the very interested, the numerical values associated with the disparity plot and the EM trajectories, as well as the residuals plotted vertically in the latter plot, are available in component `$Misc`).

This was a simple example without any covariates. In general an arbitrary number of fixed effects can be specified, and the random component can be an intercept (∼1) or a single variable giving a model with a random slope *and* random intercept.

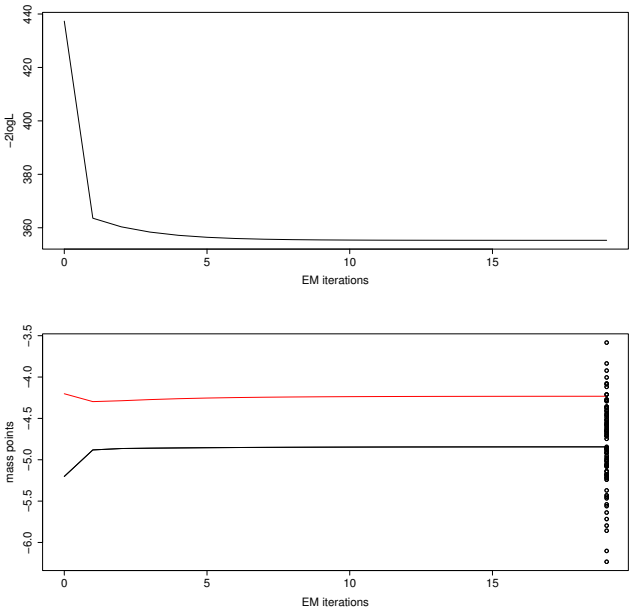

Figure 1: Top: Disparity ($-2\log L$) trend; bottom: EM trajectories. The vertical dots in the latter plot are the residuals of the fixed part of the fitted random effect model (Note that these residuals are not centered around 0, in contrast to the residuals of a simple fixed effect model. The plotted residuals, generally $h^{-1}(y_i) - x_i'\hat{\beta}$, represent the random effect distribution and are on the same scale as the mass points).

The function `allvc`

The function `alldist` is designed to fit simple overdispersion models (i.e., one has a random effect on the individual observations). However, often one wishes to introduces *shared* random effects, e.g. for students from the same class or school, for the same individual observed repeatedly over time (longitudinal data), or in small area estimation problems. This leads to variance component models, which can be fitted in **npmlreg** using the function `allvc`. As an example, let us consider the Oxford boys data from Brush and Harrison (1990), which were analyzed with NPML using GLIM4 by Aitkin et al. (2005).

The data set is part of the R package **nlme** (Pinheiro et al., 2005) and contains the heights of 26 boys, measured in Oxford on nine occasions over two years. The boys, indexed by the factor `Subject`, represent the upper level (primary sampling units, PSU), and the particular measurements at different time points correspond to the lower-level units (secondary sampling units, SSU).

As suggested by (Aitkin et al., 2005, p. 495), we fit a Gaussian model with unspecified random effect distribution and $K = 7$ mass points,

```
(Oxboys.np7 <- allvc(height ~ age, random =
~1|Subject, data = Oxboys, k=7))$disparity
[1] 1017.269
```

which confirms the GLIM4 result. Aitkin et al. state that for all models using $K > 7$ the fitted mass points 'are duplicated with the same total mass as in lower-dimension models', and hence consider this 7- mass point model as 'sufficiently complex'. However, fitting for comparison a simple linear mixed model with function `lmer` in package **lme4** (Bates and Sarkar, 2006)

```
(fm1 <- lmer(height ~ age + (1|Subject),
data = Oxboys, method = "ML"))
```

gives the `MLdeviance`, i.e., disparity, of 940.569. As this model is based on a normal random effect distribution, NPML should be superior to this, or at least competitive with it. Therefore, we went on fitting models with $K = 8$ and $K = 9$ mass points, yielding

```
 (Oxboys.np8 <- allvc(height ~ age, random =
 ~1|Subject, data = Oxboys, k=8))$disparity
 [1] 931.3752
(Oxboys.np9 <- allvc(height ~ age, random =
 ~1|Subject, data = Oxboys, k=9,
 tol=0.3))$disparity
 [1]  916.0921
```

For both models, all mass points are indeed distinct. For instance, for the 9-mass point model one has

```
       Estimate  Std. Error
age     6.523719 0.05094195
MASS1 130.200174 0.16795344
MASS2 138.416628 0.09697325
MASS3 143.382397 0.09697225
MASS4 147.350113 0.07511877
MASS5 150.954327 0.06857400
MASS6 153.869256 0.11915344
MASS7 156.178153 0.09676418
MASS8 159.521550 0.16795354
MASS9 164.883640 0.11876372
```

This increased performance compared to the GLIM code is due to the installation of a 'damping' mechanism in the first cycles of the EM algorithm; see Einbeck and Hinde (2006) for details.

Further increase of K does not yield major drops in disparity, so we continue to work with nine mass points, and extend the model by allowing the linear trend to vary across boys. The function call then takes the form

```
 (Oxboys.np9s <- allvc(height ~ age, random =
  ~age|Subject, data = Oxboys, k = 9,
  tol=0.3))
```

The difference in disparities is

```
> Oxboys.np9$disparity - Oxboys.np9s$disparity
[1] 102.1071
```

on

```
>  Oxboys.np9$df.res - Oxboys.np9s$df.res
[1] 8
```

degrees of freedom, showing clear heterogeneity in the slope.

Summary

We have introduced the R package **npmlreg**, which is in some parts a simple translation from the corresponding GLIM4 code, but, in other parts, contains substantial extensions and methodological improvements. In particular, we mention the possibility to fit Gamma models and to work with dispersion parameters varying smoothly over components, and, as already noted, the installation of a damping procedure. We note finally that for the sake of comparability all implemented features are also available for Gaussian Quadrature instead of NPML (leaving the z_k and π_k fixed and equal to Gauss-Hermite integration points and masses). The R package is available on CRAN. Future developments will include 3-level models and multicategory responses.

Acknowledgements

The work on this R package was supported by Science Foundation Ireland Basic Research Grant 04/BR/M0051. We are grateful to N. Sofroniou for sharing his GLIM code for the Missouri data and two anonymous referees for pointing at the comparison with package **lme4**.

Bibliography

M. Aitkin. A general maximum likelihood analysis of overdispersion in generalized linear models. *Statistics and Computing*, 6:251–262, 1996a. 26

M. Aitkin. Empirical Bayes shrinkage using posterior random effect means from nonparametric maximum likelihood estimation in general random effect models. In *IWSM 1996: Proceedings of the 11th International Workshop on Statistical Modelling, Orvieto*, pages 87–94, 1996b. 27, 28

M. Aitkin. A general maximum likelihood analysis of variance components in generalized linear models. *Biometrics*, 55:218–234, 1999. 26

M. Aitkin and B. Francis. Fitting overdispersed generalized linear models by nonparametric maximum likelihood. *The GLIM Newsletter*, 25:37–45, 1995. 26

M. Aitkin, B. Francis, and J. Hinde. *Statistical Modelling in GLIM 4*. Oxford Statistical Science Series, Oxford, UK., 2005. 27, 28

D. Bates and D. Sarkar. *lme4: Linear mixed-effects models using S4 classes*, 2006. R package version 0.995-2. 29

D. Böhning, P. Schlattmann, and B. Lindsey. Computer-assisted analysis of mixtures (C.A.MAN): statistical algorithms. *Biometrics*, 48: 283–303, 1992. 26

G. Brush and G. A. Harrison. Components of growth variation in human stature. *Mathematical Medicine and Biology*, 7:77–92, 1990. 28

J. Einbeck and J. Hinde. A note on NPML estimation for exponential family regression models with unspecified dispersion parameter. *Austrian Journal of Statistics*, 35:233–243, 2006. 29

J. Einbeck, R. Darnell, and J. Hinde. *npmlreg: Nonparametric maximum likelihood estimation for random effect models*, 2006. R package version 0.40. 26

N.M. Laird. Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73:805–811, 1978. 26

J. Pinheiro, D. Bates, S. DebRoy, and D. Sarkar. *nlme: Linear and nonlinear mixed effects models*, 2005. R package version 3.1-66. 28

A. Skrondal and S. Rabe-Hesketh. *Generalized Latent Variable Modelling*. Chapman & Hall/CRC, Boca Raton, 2004. 26

N. Sofroniou, J. Einbeck, and J. Hinde. Analyzing Irish suicide rates with mixture models. In *IWSM 2006: Proceedings of the 21th International Workshop on Statistical Modelling, Galway*, pages 474–481, 2006. 28

R. Tsutakawa. Estimation of cancer mortality rates: a Bayesian analysis of small frequencies. *Biometrics*, 41:69–79, 1985. 27

Jochen Einbeck
Durham University, UK
jochen.einbeck@durham.ac.uk
John Hinde
National University of Ireland, Galway, Rep. of Ireland
john.hinde@nuigalway.ie
Ross Darnell
The University of Queensland, Australia
r.darnell@uq.edu.au

Augmenting R with Unix Tools

Andrew Robinson

This article describes a small collection of Unix command-line tools that can be used to augment R. I introduce:

`make` which after judicious preparation will allow one-word execution of large-scale simulations, all the way from compilation of source code to construction of a PDF report;

`screen` which will permit remote monitoring of program execution progress with automatic protection against disconnection; and

`mail` which will allow automatic alerting under error conditions or script completion.

These programs will be part of the default installations of many flavours of Unix-like operating systems. Although these tools have many different applications, each is very useful for running R remotely on a Unix-like operating system, which is the focus of their usage in this article.

Regardless of what kind of computer is on your own desk, you may find these tools useful; they do not require you to be running a BSD or GNU/Linux on your own machine.

R and `screen`

It is sometimes necessary to run R code that executes for long periods of time upon remote machines. This requirement may be because the local computing resources are too slow, too unstable, or have insufficient memory.

For example, I have recently completed some simulations that took more than a month on a reasonably fast cluster of Linux computers. I developed, trialed, and profiled my code on my local, not-particularly-fast, computer. I then copied the scripts and data to a more powerful machine provided by my employer and ran the simulations on R remotely on that machine. To get easy access to the simulations whilst they ran, I used `screen`.

`screen` is a so-called terminal multiplexor, which allows us to create, shuffle, share, and suspend command line sessions within one window. It provides protection against disconnections and the flexibility to retrieve command line sessions remotely. `screen` is particularly useful for R sessions that are running on a remote machine.

We might use the following steps to invoke R within `screen`:

1. log in remotely via secure shell,