Paper 902-2017

# Fitting Statistical Models with PROCs NLMIXED and MCMC

Robin High and Wael ElRayes, University of Nebraska Medical Center

SAS®/Stat software has several procedures which estimate parameters from generalized linear models designed for both continuous and discrete response data (including proportions and counts). SAS procedures such as GENMOD, GLIMMIX, LIFEREG, and FMM, among others, offer a flexible range of analysis options to analyze data from a variety of distributions and also with correlated or clustered data. GENMOD can model zero-inflated count distributions and FMM a more extensive list of zero-inflated and truncated distributions. This paper demonstrates how statements from NLMIXED can be written to match the output results from these procedures. Situations arise where the flexible programming statements of NLMIXED are needed for situations such as zero inflated, hurdle models, truncated counts, and continuous data (including proportions) which have random effects and for probability distributions not available as a model option. A useful application of these coding techniques is that programming statements from NLMIXED can often be directly transferred into the SAS procedure MCMC with little or no modification to perform analyses from a Bayesian perspective with these various types of complex models.

## INTRODUCTION

Statistical modeling of data can be a complex process such that the commonly used SAS/STAT procedures may not have options to deal with important components of the analysis. Examples include adding random effects to zero-inflated or hurdle models, modeling the precision with data bounded between 0 and 1, specifying boundary values on the coefficients to model the presence of 0s, or with probability distributions not available with the MODEL statement `distribution=<option>`. A procedure which can deal with these somewhat infrequent, yet important, modeling situations is PROC NLMIXED; however, programming statements are needed within the SAS code to do so.

The objective of this paper is to demonstrate how to write code in the NLMIXED procedure that matches the parameter estimates from the GLIMMIX, GENMOD, or FMM procedures and also to describe NLMIXED statements that provide features not available with them. Another useful aspect of this approach is many of the NLMIXED statements described here can be copied directly into the MCMC procedure to produce results from a Bayesian perspective.

## MOTIVATING EXAMPLE

Assume a data set contains a response variable (notated throughout this document as y) which comes from a continuous or discrete distribution. The objective is to examine the relationship of the response to three categorical variables and one continuous variable, the fixed effects of the model:

| Variable Name | Model Coef | Levels | Example Values |
|---|---|---|---|
| S_ | b1 | 3 | Size: L=Large / M=Medium / S=Small |
| G_ | b2 | 2 | Gender: F=Female / M=Male |
| L_ | b3 | 3 | Level: 1/2/3 |
| x | b4 | | Continuous |
| y | | | Continuous or Discrete, Bounded or Unbounded |

**Table 1. Factors and characteristics of values of the motivating example**

The variable names for the three classification factors in the first column of the table all end with the underscore for reasons that will become evident in the coding examples. In general, whenever identifying variables to be written into programming statements in GLIMMIX (and especially PROC NLMIXED), avoid variable names that begin with an underscore _ since the name may conflict with

internal variables within the procedure. Also, names for the parameter estimates will generally start with the letter b followed by a number, as shown in the column labeled “Model Coef,” the variable names in the linear predictor equation for the model with both PROC NLMIXED and PROC MCMC.

## ANALYSIS WITH PROC GLIMMIX

With one observation for each subject, the GLIMMIX, GENMOD, or FMM procedures are choices to model data from a variety of distributions. For example, with data bounded by 0 and 1, to compare means across levels of one or more within-subject factors, PROC GLIMMIX would be a choice with the RANDOM statement to account for clustering of data within each subject:

```
PROC GLIMMIX DATA=dt1 method=quad;
CLASS S_ G_ L_ subj ;
MODEL y = S_  G_  S_*G_   L_ x / dist=beta solution;
RANDOM intercept / subject=subj;
LSMEANS S_*G_   L_ / ilink cl at x = 5 e;
TITLE1 "Response";
RUN;
```

For data bounded between 0 and 1 which also have a substantial number of observations with responses of 0, PROC FMM offers a zero-inflated (altered) beta model whereas GLIMMIX does not. Also, both procedures compute only one precision parameter. However, with responses of 0, repeated measurements from the same subject, and the capability to model the precision parameter, the NLMIXED and MCMC procedures can be utilized.

## ANALYSIS WITH PROC NLMIXED

The following sections illustrate the basic statements that generate data analysis results from various response distributions with NLMIXED and subsequently shown how to transfer them into PROC MCMC. The framework for coding linear statistical models in NLMIXED consists of these statements:

```
PROC NLMIXED DATA=indata < options > ;
PARMS < initial parameter values > ;
* enter programming statements;
eta = < linear predictors for the probability model, the mean, and precision > ;
mu = G(eta) ;   * G is the inverse link function;
lglk = LOG(f(x|mu,phi)); * Log-Likelihood of the probability density function;
MODEL y ~ GENERAL(LgLk);
< RANDOM, CONTRAST, ESTIMATE, and OUTPUT statements as needed >
RUN;
```

Many programming statements as they would be written within a DATA step can also be entered into NLMIXED. For example, ARRAY statements work much the same way as they do in a DATA step, although the indices must be explicitly defined. The equations defining the linear predictor, inverse link, and log likelihood statements all have the same programming syntax. Brief descriptions of these programming statements follow.

### THE MODEL STATEMENT

The MODEL statement defines the data distribution based on the characteristics of the data to be analyzed. Several choices are directly available; others can be coded through their log likelihood equations. In NLMIXED, six response distributions can be specified with the MODEL statement. Enter the name of the desired distribution with its parameters to model the response:

```
MODEL y ~ distribution(parameters);
```

| Distribution(parms) | Description with necessary parameters |
|---|---|
| NORMAL(mu,vr) | normal with mean mu and variance vr |
| BINARY(p) | binary (Bernoulli) with probability p |
| BINOMIAL(n,p) | binomial with count n and probability p |
| GAMMA(a,b) | gamma with shape a and scale b |
| NEGBIN(n,p) | negative binomial with count n and probability p |
| POISSON (mu) | Poisson with mean mu |

**Table 2. Probability distributions available in NLMIXED**

Since the objective of this paper is to describe how NLMIXED can be utilized for complex modeling situations, such as mixtures of distributions, truncated or censored distributions, or for other types of distributions not available from this list, the seventh type of MODEL statement will appear in all examples:

```
MODEL y ~ GENERAL(lglk);
```

The objective is to compute estimates for the model parameters that maximize the log-likelihood function identified with the variable LgLk computed with SAS programming statements. The various types of continuous and discrete distributions which can be modeled through programming statements in both NLMIXED and MCMC include:

| Continuous | Model | Discrete | Model |
|---|---|---|---|
| Normal | * | Binary | * |
| Beta | | Binomial | * |
| Exponential | | Beta Binomial | |
| Gamma | * | Binomial Cluster | |
| Log-Logistic | | Multinomial<br>- Ordinal (3 types)<br>- Generalized Logit | |
| Weibull | | | |
| | | | |
| | | Poisson | * |
| | | Negative Binomial | * |
| **Distributions can incorporate these features** | | | |
| - Zero altered<br>- Censoring<br>- Truncation | | - Zero inflated (altered)<br>- Hurdle<br>- Truncation | |

**Table 3. Probability distributions programmable in NLMIXED where * indicates available with MODEL**

Statements to implement these log-likelihood equations in NLMIXED and MCMC are provided in file with the sample SAS code (its contents and directions for use is described in Appendix B) which can be adapted for other data sets.

## THE PARMS STATEMENT

In NLMIXED, initial values of the parameter estimates are set to 1 by default (any name within the code which is not included as a variable in the input data set or the result of a computation). The PARMS statement sets initial values for the estimates:

```
PARMS b0 -4 b1 .1 b2 .1 b3 .1 b4 .1;
```

Values should always be chosen within a feasible region of the parameter space or if unknown, enter a reasonable estimate. An initial value of 0 may not be a good choice if division occurs anywhere in the code or a non-positive value if it will be entered into a log function in a programming statement. An "equal to" sign between the variable name and its initial value is permissible. However, with one initial value for each parameter, the equals sign can be omitted, since the logic, syntax, and values computed with programming statements in the initial iteration can easily be examined by copying the computational statements into a DATA step and replacing the keyword PARMS with RETAIN:

```
DATA test;
SET indata;
RETAIN b0 -4 b1 .1 b2 .1 b3 .1 b4 .1;
< NLMIXED programming statements > ;
PROC PRINT; RUN;
```

Initial parameter can also be entered from a SAS data set, to be illustrated with a later example.

## CONSTRUCTING LINEAR PREDICTORS

The underlying equation that relates the explanatory data to the mean of the response is given with a linear predictor called eta consisting of p explanatory variables (x1–xp) with coefficients b0 for the intercept and b1–bp for the predictor variables:

```
eta = b0 + b1*x1 + b2*x2 + .. + bp*xp
```

The $x_i$ variables refer to dummy coded classification variables, their interactions, along with names for continuous data. For consistency and ease of programming, parameter names will appear with the letter b followed by a number and in cases of factors with multiple levels or interactions, other letters/numbers to associate it with respective levels of the explanatory data. Although the letters NL of the procedure name NLMIXED imply it is designed to work with non-linear equations (see Kurada and the NLMIXED documentation for examples), all of the models demonstrated here will have a linear equation for parameter estimation. One important characteristic of the linear predictor is that the values computed from the coefficients applied to the input data should freely cover the practical range of real numbers, both negative and positive, a feature that an appropriate link function provides.

For categorical explanatory data a CLASS statement is available with many SAS procedures. However, both NLMIXED and MCMC do not have a CLASS statement, which can be one of the more challenging aspects of writing code, especially constructing linear predictors with several categorical variables and their interactions, resulting with the need to compute dummy coded variables. Despite this limitation, the equation for the linear predictor(s) needs to be written out with programming statements, just as it would be written in a DATA step, with dummy (0/1) coding for levels of categorical explanatory data.

For categorical variables which have no missing data, the linear predictor eta may be computed with the parameter estimates multiplied by indicator variables for the categorical explanatory data (main effects and interactions) without any modifications to the input data set:

```
eta = b0 + b1a*(S_ = 'L') + b1b*(S_ = 'M')
         + b2*(G_ = 'F')
         + b1a_b2*(S_ = 'L')*(G_ = 'F') + b1b_b2*(S_ = 'M')*(G_ = 'F')
         + b3a*(L_ = '1') + b3b*(L_ = '2') + b4*x;
```

Dummy variables for the classification factors can also be computed externally within a DATA step. However, for models with several categorical predictor variables, including interactions, dummy coded categorical data can be produced more efficiently. PROC GLMMOD converts categorical data into dummy coded variables for fixed effects models. PROC GLIMMIX has the outdesign=(<data set name>) option which produces variables with names _X1, _X2, _X3, etc. Its use requires extra care in matching these variables to parameter names. However, for models computed with PROCs NLMIXED and MCMC, the conversion of categorical data into dummy codes may be done most efficiently with PROC TRANSREG (illustrated for the zero-inflated beta model in the example included in the SAS code file) which produces new variable names combining the variable name(s) and their coded levels (thus, the reason for the trailing _ entered on the variable names in Table 1):

```
PROC TRANSREG DATA=dt1 design;
MODEL CLASS(S_  G_  S_*G_  L_ / ZERO=last);
ID subj x y;
OUTPUT out=dt1I(drop=_type_ intercept ); * Output data set for NLMIXED/MCMC;
RUN;
```

The main effects and interactions desired for the model are defined with a CLASS statement just as they would be written in a MODEL statement in GLIMMIX or GENMOD where the option ZERO=last indicates the highest sorted value is the reference category. Other data such as the subject identification codes and numerical values (including the response variable and continuous covariates) are placed on the ID statement. The dummy coded categorical data and original data are saved in the output data set `dt1I`. A helpful feature is the procedure produces the macro variable `&_trgind` which contains the names and specific levels of each dummy coded categorical variable. The names of the dummy variables can be extracted by printing the contents of the macro variable in the LOG window:

```
%PUT &_trgind;

S_L S_M  G_F  S_LG_F S_MG_F   L_1 L_2
```

In these new variable names, the first part of the name for Size (S_) is the original name immediately followed by the dummy coded value (L or M; level S is the reference category and is not provided a dummy coded column). Interactions concatenate both the respective variable names and levels (S_L and G_F are combined for size=Large and gender=Female called `S_LG_F`). The macro variable `&_trgind` can be entered or elements from it extracted as needed.

```
PROC PRINT DATA=dt1I(obs=6) NOObs;
VAR subj S_ G_ L_ &_trgind x y;
RUN;
```

| | -orig data- | | | ---------- dummy coded values ---------- | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Subj | S_ | G_ | L_ | S_L | S_M | G_F | S_LG_F | S_MG_F | L_1 | L_2 | x | y |
| 001 | L | M | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 3.7 | 0.040 |
| 001 | L | M | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 5.1 | 0.046 |
| 001 | L | M | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5.3 | 0.031 |
| 002 | M | F | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 3.7 | 0.036 |
| 002 | M | F | 2 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 5.1 | 0.046 |
| 002 | M | F | 3 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 5.3 | 0.026 |
| Coef: | | | | b1a | b1b | b2 | b1a_b2 | b1b_b2 | b3a | b3b | b4 | y |

An efficient way to enter the equation for the linear predictor into the code for NLMIXED and MCMC is with two ARRAY statements, one for the dummy coded variable names (the first array called dt) and the other for the names of the parameter estimates for each variable (the second array called bt) which are then multiplied and summed.

```
ARRAY dt[8] &_trgind. x;
ARRAY bt[8] b1a b1b b2 b1a_b2 b1b_b2 b3a b3b b4 ;
eta = b0; DO i = 1 to 8; eta = eta + bt{i}*dt{i}; end;
eta = eta + v; * included with random effect models;
```

The value of eta is initially set to the intercept (b0) followed by the accumulation of the sum of the product of the dummy variables and continuous data (in the array dt) multiplied by the values of the respective parameter estimates (in the array bt). For random effect models, a component v is also added to eta (which is defined as having mean 0 and variance sv2 by a RANDOM statement). The SAS documentation for NLMIXED (PDF version 14.1, pdf file, p. 6545) indicates that ARRAY statement names should not exceed 8 characters; however, like the DATA step, longer variable names are acceptable.

## INITIAL PARAMETER VALUES

A PARMS statement in PROC NLMIXED is optional (though often necessary); with MCMC they are mandatory. However, computational errors and non-convergence may occur because the default starting values of 1 (NLMIXED) are too far away from the optimal solution or they may give a computational error (a reason to check computations in a DATA step). In complex statistical models, entering initial coefficient estimates that approximate the optimal solution in sign and magnitude may

help the model to converge to a global maximum, as well as reduce the amount of time required to do so. The ODS OUTPUT file for parameter estimates in NLMIXED has the necessary variable names and contents that can be sent to more complex NLMIXED code. A model may be written with the linear predictor eta (which includes a two-level class variable called group coded as 0/1) and the inverse link along with the scale and shape parameters such as the negative binomial, but instead treating them as the mean and variance from a normal distribution:

```
ODS OUTPUT parameterestimates=prmsI(keep=parameter estimate);

PROC NLMIXED data=negbin_data ;
eta = b0 + b1*(group=0);
mu = exp(eta);
MODEL y ~ NORMAL(mu,(1/phi)**2);
TITLE 'NLMIXED: Initial parameter estimates based on normal distribution';
RUN;
```

Parameter estimates are saved in the ODS output data set `prmsI` which is then entered as an option following a / in the PARMS statement as the initial values for a negative binomial count model:

```
PROC NLMIXED DATA=negbin;
PARMS / data=prmsI;
eta = b0 + b1*(group=0);
mu = exp(eta);
LgLk = y*log(phi*mu) - (y+(1/phi))*log(1+(phi*mu))
     + lgamma(y+(1/phi)) - lgamma(1/phi) - lgamma(y+1) ;
MODEL y ~ general( LgLk ) ;
TITLE 'NLMIXED: Negative Binomial';
RUN ;
```

The PARMS statement should also include initial estimates for coefficients of any new variables not in the PARMS input data file (e.g., precision parameters or a random effect):

```
PARMS logsig -1 / data=init_prmsI;
```

The initial parameter estimates are entered here in the narrow or univariate format with one record for each parameter defined by the variable names “parameter” and “estimate.” Getting initial estimates from other procedures usually requires recoding data or renaming variables, whereas the ODS OUTPUT statement from NLMIXED provides the file structure needed. A multivariate layout of the data set with one record consists of the estimates defined by the parameters as the variable names. PROC MCMC requires a PARMS statement; however, it does not have this method of defining initial parameter estimates, though it does have several other ways to do so not available with NLMIXED.

## ANOVA TABLE AND LSMEANS

In NLMIXED, pvalues for factors and interactions available in ANOVA tables can be produced with CONTRAST statements; LS-means can be computed with ESTIMATE statements. Examples of how to write these statements in NLMIXED are shown in Appendix A and also the Zero-inflated beta model in the SAS code file. PROC MCMC does not have CONTRAST or ESTIMATE statements, yet these computations are available by entering the equations within the MCMC procedure and including the `monitor=(< var1 var2 .. >)` option on the PROC statement (illustrated for many of the examples in the SAS code file) or by processing the post fitted output file.

## RANDOM SUBJECT EFFECTS

Models with random subject effects in NLMIXED are defined with a RANDOM statement:

```
RANDOM v ~ NORMAL(0,sv2) subject=subj;
```

where v (added to the linear predictor) is the deviation of the classification factor identified with `subject=<cluster>` specification. Versions of SAS/STAT prior to 13.1 required the input data file to be sorted by this factor. Subsequent versions of SAS do not require this sorting; they also provide

computations for nested random effects, two random variables such as subject=subj(trt) (see Kurada, 2016). Since variances are always positive, bounded from below by 0, when the response or subject variances are expected to be small it may be helpful to model the log of the variance, rather than the value itself which must remain positive.

```
RANDOM v ~ NORMAL(0,exp(2*logsig)) subject=subj;
```

The estimated variance for subject can then be computed from an ESTIMATE statement:

```
ESTIMATE 'Subject Variance' exp(2*logsig);
```

## Examples: Assembling the Components

The examples presented in the accompanying SAS file demonstrate the how to write NLMIXED code for data analyses that would be typically run with the procedures GLIMMIX, GENMOD, LIFEREG, or FMM. In many of these examples the NLMIXED statements can be copied into MCMC. The primary differences are the options placed on the PROC MCMC statement and the addition of non-informative PRIOR statements for each model parameter (note of simplicity of this statement when parameter names start with the same letter). Initial values must be specified in a PARMS statement in MCMC (which has several alternative methods not provided with NLMIXED, however, the input data=<file> option for initial estimates is not available).

### Truncated Negative Binomial Distribution

A truncated negative binomial model has 0s excluded from the data analysis as a possible outcome. It is often run with GENMOD or GLIMMIX without concern for how the absence of 0s may affect results. Application of a more appropriate truncated count data model can be run with PROCS FMM and NLMIXED with these statements:

```
PROC FMM DATA = negbin_data;
CLASS group;               * a two-level class variable coded as 0/1 ;
MODEL y = group / DIST=tnegbin link=log;
TITLE 'FMM: Zero Truncated Negative Binomial';
RUN;
```

```
PROC NLMIXED DATA = negbin_data;
PARMS b0 .1  b1 .1 phi .1;
eta = b0 + b1*(group=0);    * group=1 is the reference category ;
mu = exp(eta);
lglk = y*LOG(phi*mu) - (y+(1/phi))*LOG(1+(phi*mu))
       + lgamma(y+(1/phi)) - (lgamma(1/phi) + lgamma(y+1)) /*neg bin loglikelihood*/
       - LOG(1-(((phi*mu)+1)**(-1/phi)) );                 /* divide by (1-P(y=0))*/
MODEL y ~ general(lglk);
TITLE 'NLMIXED: Zero Truncated Negative Binomial';
RUN;
```

The NLMIXED parameter estimates match results from PROC FMM (see SAS code). Note if there happen to be any stray 0s in the data set, FMM automatically excludes them, whereas the NLMIXED and MCMC code below will still include them in the calculations. Much of this NLMIXED code can be copied directly into PROC MCMC along with two PRIOR statements:

```
PROC MCMC DATA = negbin_data seed=341925 nmc=40000 opti=quanew;
PARMS b0 .1  b1 .1 phi .1;
PRIOR b: ~ normal(0, var=1e6);
PRIOR phi  ~ normal(0, var=1e6);
eta = b0 + b1*(group=0);
mu = exp(eta);
lglk = y*LOG(phi*mu) - (y+(1/phi))*LOG(1+(phi*mu))
     + lgamma(y+(1/phi)) - (lgamma(1/phi) + lgamma(y+1)) /*neg bin loglikelihood*/
     - LOG(1-(((phi*mu)+1)**(-1/phi)) );                 /* divide by (1-P(y=0))*/
MODEL y ~ general(lglk);
TITLE 'MCMC: Zero Truncated Negative Binomial';
RUN;
```

The MCMC parameter estimates will have small differences compared with those from FMM and NLMIXED because everything in MCMC is random. Adding a RANDOM statement in both procedures to work with multiple measurements from each subject greatly increases the computational resources needed, especially for PROC MCMC.

**Censored Weibull Distribution**

An example with continuous time to event data computes estimates with the response y has right-censored observations coded alongside a second variable having the value cens=0 for the final measurement and cens=1 for when y is right-censored. In this case, the outcome y is the maximum time observed, yet the event of interest has not yet occurred or is the last known value. Right-censoring can also occur when a data collection device cannot reliably record measurements above a fixed value.

```
PROC LIFEREG DATA=trc;
CLASS group;
MODEL y*cens(1) = group / dist=weibull;
TITLE "LIFEREG: Right Censored Weibull";
RUN;

PROC NLMIXED DATA=trc;
PARMS b0 .1 b1 .1 phi .5;
eta  = b0 + b1*(group=0); * group=1 is the reference category ;
mu = EXP(eta);
lglk = (cens=0) * (LOG(phi/mu) + (phi-1)*LOG(y/mu) + (-(y/mu)**phi) )
     + (cens=1) * (-(y/mu)**phi);
MODEL y ~ GENERAL(lglk);
TITLE "NLMIXED: Right Censored Weibull";
RUN;
```

As demonstrated in several examples in the file with SAS code, the **LOGSDF** function (log of the survival) could be entered for the log of the upper tail probability in the log-likelihood equation with censored observations:

```
lglk = (cens=0) * (LOG(phi/mu) + (phi-1)*LOG(y/mu) + (-(y/mu)**phi) )
     + (cens=1) * LOGSDF('WEIBULL', y, phi, mu);
```

These NLMIXED statements can be copied directly into the MCMC step, with the addition of the two PRIOR statements:

```
PROC MCMC DATA=trc seed=872695 nmc=40000 opti=quanew;
PARMS b0 .1 b1 .1 phi .5;
PRIOR b: ~ normal(0, var=1e6);
PRIOR phi  ~ normal(0, var=1e6);
eta  = b0 + b1*(group=0);
mu = EXP(eta);
lglk = (cens=0) * (LOG(phi/mu) + (phi-1)*LOG(y/mu) + (-(y/mu)**phi) )
     + (cens=1) * (-(y/mu)**phi);
MODEL y ~ general(lglk);
TITLE "MCMC: Right Censored Weibull";
RUN;
```

Computational problems may occur with both NLMIXED and MCMC for complex statistical models. Getting the code entered accurately is essential. In order to compute a maximum likelihood solution for a given model, variation in the data must be present. Read the documentation section regarding choice of optimization method considering the size of the data set and model complexity. A few of the MCMC models based on the NLMIXED code do not converge to reasonable estimates, as noted in the examples in the SAS file. With both procedures, always check for model convergence, especially the notes written in the LOG window and the graphical displays and diagnostic tables produced with PROC MCMC. The respective documentation for both procedures have methods to assist with model convergence, yet may greatly increase the processing time.

## References

Chen, F., Brown, G., and Stokes, M., Fitting Your Favorite Mixed Models with PROC MCMC (2016), SAS Global Forum, Paper SAS5601-2016.

High, R., Models for Ordinal Response Data (2013). SAS Global Forum, Paper 445-2013.

Kurada, R. R., Fitting Multilevel Hierarchical Mixed Models Using PROC NLMIXED (2016). SAS Global Forum, Paper SAS4720-2016.

Lemus, Hector, Demystifying the CONTRAST and ESTIMATE Statement (2016). Western SAS Users Regional Conference.

Morel, Jorge G. and Nagaraj K. Neerchal. 2012. Overdispersion Models in SAS®. Cary, NC: SAS Institute Inc.

Your comments and questions are valued and encouraged. Contact the author at:

Robin High
Department of Biostatistics
University of Nebraska Medical Center
984375 Nebraska Medical Center
Omaha, NE 68198-4375
email: rhigh@unmc.edu

Wail ElRayes
Environmental, Agricultural and Occupational Health
University of Nebraska Medical Center
984375 Nebraska Medical Center
Omaha, NE 68198-4375
email: wael.elrayes@unmc.edu



## APPENDIX A

## Computing the ANOVA Table

In NLMIXED, CONTRAST statements produce pvalues for specific contrasts, in this case testing the overall significant of the fixed effects:

```
CONTRAST "   b1: Size"          b1a - 0, b1b – 0;
CONTRAST "   b2: Gender"        b2  – 0;
CONTRAST "b1_b2: Size x Gender" b1a_b2 - 0, b1b_b2 – 0;
CONTRAST "   b4: x"             b4 – 0;
```

The descriptive content of each effect can be formed by extracting the variable names from the &_trgind macro variable produced with TRANSREG and placing the respective coefficient b# at the beginning, as show above. The printed pvalue is the joint test that all coefficients which belong to the term equal 0 (with only one coefficient, the pvalue here will equal the pvalue from the parameter estimate table). Note that size has three levels, so it is coded with two dummy variables with two degrees of freedom. Also the two-factor interaction consists of 2 additional coefficients to be estimated. Note that with more complex situations, such as models with random effects, these tests will differ from GLIMMIX since the more advanced methods of computing F values and degrees of freedom are not available in NLMIXED.

## Computing LSMEANS

With the GLIMMIX procedure, to compute LsMeans for the levels of the categorical fixed effects only needs to have the name of the effect of interest on the LSMEANS statement. The resulting values are a linear function of the estimated model coefficients notated as **L*ß**. Replication of the LsMeans in PROC NLMIXED is assisted with components of **L** matrix which can be produced in GLIMMIX with the e option:

```
LSMEANS S_*G_ L_ / ilink cl at x=5 e;
```

The coefficient matrix is saved with ODS:

```
ODS OUTPUT coef=lsmcoef;
```

The content of the file `lsmcoef` includes a variable called Lmatrix, an increasing integer for each factor for which Ls-Means are requested. In this case there are one set of coefficients for the S_*G_ interaction and a second second set of LsMeans for the three-level factor L_. In order to utilize these coefficients in NLMIXED, the file can be transposed and fractional values converted to integers. The file contains several variable names that begin with "Row" followed by a number. The largest value of this index number is the specific factor or interaction that contains the largest number of levels. In this case, the S_*G_ interaction has 6 combinations, so the macro variable below, `&nrws`, is set to 6. (If the main effect L_ had 7 levels, then the number of rows would be 7.) Also, the least common denominator for the 3-level variables S_ and L_ and the 2-level variable G_ is **6** (the macro variable where this multiplication is placed is emphasized in bold print):

```
%LET nrws = 6; * largest number of factor or interaction levels;
%LET mltp = 6; * the least common denominator to produce integer coefficients;

DATA lsmcoef;
SET lsmcoef;
LENGTH effct $20; DROP i;
effct=COMPRESS(CATT(effect,S_,G_,L_),'*'); * Enter the names of the fixed effects;
ARRAY rw{&nrws.} row1 - row&nrws.;
DO i = 1 to &nrws.; IF rw{i} = . then rw{i} = 0; rw{i} = rw{i}*&mltp.; end;

PROC TRANSPOSE DATA=lsmcoef out=tlsmcoef prefix=cf_;
BY lmatrix; VAR row: ; ID effct;
PROC PRINT DATA=lsmcoef NOObs; run;
```

The long variable names are not printed; the column headers below refer to the factors for the main effects, interaction, and covariate in the model. The first Lmatrix for the `S_*G_` interaction is:

| | Intcp | S_ | | G_ | | S_*G_ | | | | L_ | | | x |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Row1 | 6 | 6 | 0 | 6 | 0 | 6 | 0 | 0 | 0 | 2 | 2 | 2 | 3 |
| Row2 | 6 | 6 | 0 | 0 | 6 | 0 | 6 | 0 | 0 | 2 | 2 | 2 | 3 |
| Row3 | 6 | 0 | 6 | 6 | 0 | 0 | 0 | 6 | 0 | 2 | 2 | 2 | 3 |
| Row4 | 6 | 0 | 6 | 0 | 6 | 0 | 0 | 0 | 6 | 2 | 2 | 2 | 3 |
| Row5 | 6 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 3 |
| Row6 | 6 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 2 | 2 | 2 | 3 |

The order of the factor names and ordering of their levels, the coefficients printed in the ESTIMATE statements below show how they are written in GLIMMIX to match the LSMEANS:

```
ESTIMATE 'Size=L /Gnd=F' intercept 6 S_ 6 0 0  G_ 6 0   S_*G_ 6 0  0 0  0 0   L_ 2 2 2   x 30 / divisor=6;
ESTIMATE 'Size=L /Gnd=M' intercept 6 S_ 6 0 0  G_ 0 6   S_*G_ 0 6  0 0  0 0   L_ 2 2 2   x 30 / divisor=6;
ESTIMATE 'Size=M /Gnd=F' intercept 6 S_ 0 6 0  G_ 6 0   S_*G_ 0 0  6 0  0 0   L_ 2 2 2   x 30 / divisor=6;
ESTIMATE 'Size=M /Gnd=M' intercept 6 S_ 0 6 0  G_ 0 6   S_*G_ 0 0  0 6  0 0   L_ 2 2 2   x 30 / divisor=6;
ESTIMATE 'Size=S /Gnd=F' intercept 6 S_ 0 0 6  G_ 6 0   S_*G_ 0 0  0 0  6 0   L_ 2 2 2   x 30 / divisor=6;
ESTIMATE 'Size=S /Gnd=M' intercept 6 S_ 0 0 6  G_ 0 6   S_*G_ 0 0  0 0  0 6   L_ 2 2 2   x 30 / divisor=6;

ESTIMATE 'Level 1'   intercept 6 S_ 2 2 2  G_ 3 3   S_*G_ 1 1  1 1  1 1   L_ 6 0 0   x 30 / divisor=6;
ESTIMATE 'Level 2'   intercept 6 S_ 2 2 2  G_ 3 3   S_*G_ 1 1  1 1  1 1   L_ 0 6 0   x 30 / divisor=6;
ESTIMATE 'Level 3'   intercept 6 S_ 2 2 2  G_ 3 3   S_*G_ 1 1  1 1  1 1   L_ 0 0 6   x 30 / divisor=6;
```

These six lines can then be translated into NLMIXED code (categories which are the reference levels for each effect are assigned a coefficient of 0 and can be omitted):

```
ESTIMATE 'Size=L /Gnd=F' (6*b0 + 6*b1a         + 6*b2 + 6*b1a_b2           + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Size=L /Gnd=M' (6*b0 + 6*b1a                                     + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Size=M /Gnd=F' (6*b0         + 6*b1b + 6*b2            + 6*b1b_b2 + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Size=M /Gnd=M' (6*b0         + 6*b1b                             + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Size=S /Gnd=F' (6*b0                 + 6*b2                      + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Size=S /Gnd=M' (6*b0                                             + 2*b3a + 2*b3b + b4*30)/6;
ESTIMATE 'Level 1'       (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2 + 6*b3a         + b4*30)/6;
ESTIMATE 'Level 2'       (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2         + 6*b3b + b4*30)/6;
ESTIMATE 'Level 3'       (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2                 + b4*30)/6;
```

To produce the means on the scale of the original data, the linear predictor `lp` within the () in each ESTIMATE statement is entered into the inverse link function (the results produced with the `ilink` option on the LSMEAN statement):

```
ESTIMATE 'Mean: Size L / G=F' 1 / (1 + EXP(-(lp)));
```

A DATA _null_ step can also write the equations to enter into NLMIXED/MCMC by matching the respective coefficient names with the contents of the LMatrix file:

```
DATA _null_;
SET tlsmcoef;
PUT cf_Intercept '*b0 + ' cf_S_L '*b1a + ' cf_S_M '*b1b + '
    cf_G_F '*b2 + '
    cf_S_G_LF '*b1a_b2 +' cf_S_G_MF '*b1b_b2 + '
    cf_L_1 '*b3a + ' cf_L_2 '*b3b + '
    cf_x '*b4'
;
RUN;
```

PROC MCMC does not have an ESTIMATE statement, yet computations can be made in this manner either by entering the linear predictor equations within the procedure and include the monitor option on the PROC statement or by processing the post fitted output file.

For truncated, hurdle, and zero-inflated distributions, computing the LsMeans requires an adjustment factor, as illustrated in the SAS example file for many of the example distributions where the PROC NLMIXED means match the predicted values from PROC FMM.

## APPENDIX B

## Link Functions

The link function connects the mean of the probability distribution with the linear predictor. Although a link with a specific name is specified for a GLM, in PROC NLMIXED the function itself is not entered into the model; rather, it is the inverse link that is required since the linear predictor computes values on the scale of the specified link function. A function G is applied to each component of the mean, E(y), that relates it to the linear predictor:

| **Function** | **Link: eta=G(E(y))** | **Inverse Link** (entered in NLMIXED/MCMC) |
|---|---|---|
| Response Bounded between 0 and 1 | | |
| Logit | eta = LOG(p/(1-p) | p = 1 / (1 + EXP(-eta)) |
| Probit | eta = $PHI^{-1}$(mu) | p = CDF("normal", eta) |
| Log-Log | eta = -(LOG(-LOG(p))) | p = EXP(-EXP(-eta)) |
| | eta = LOG(-LOG(p)) | p = EXP(-EXP(eta)) |
| Complimentary Log-Log | eta=LOG(-LOG(1-p)) | p = 1 - EXP(-EXP(eta)) |
| Continuous Responses | | |
| Identity | eta | mu = eta |
| Log | eta = LOG(mu) | mu = EXP(eta) |
| Reciprocal | eta =1 / mu | mu = 1 / eta |

The linear predictor for each link is the variable named eta (or an equation for eta_zr with models having excess 0s), a function of the model coefficients and the names for the explanatory variables (dummy variables for categorical data), as defined in the main body of the text:

```
eta = b0 + b1*x1 + b2*x2 + .. + bp*xp
```

The inverse links are either defined as mean (for continuous data) and p (for proportions bounded between 0 and 1). In the examples to follow, the links commonly applied to the respective distributions are given, though other links are possible. The complementary log-log link (not symmetric) is useful for data bounded between zero and one where most of the values lie close to the boundaries, that is for events that either rare (occur infrequently) or common (occur most of the time).

## Probability Distributions for Model Statement

For each of the procedures, the mathematical notation for the formulas of the PDFs or the log likelihood equations used in SAS/STAT 14.1 can be found on these pages:

PROC GENMOD

http://support.sas.com/documentation/cdl/en/statug/68162/HTML/default/viewer.htm#statug_genmod_details01.htm

PROC GLIMMIX

http://support.sas.com/documentation/cdl/en/statug/68162/HTML/default/viewer.htm#statug_glimmix_details02.htm

PROC FMM

http://support.sas.com/documentation/cdl/en/statug/68162/HTML/default/viewer.htm#statug_fmm_details07.htm

### Log-Likelihood Equations to Produce NLMIXED and MCMC Code

For the discrete and continuous distributions, multiple choices for the link functions may exist. Regardless of which one is selected, the log-likelihood equation is not affected. The file with SAS code for various discrete and continuous distributions provides a starting point to code specific models of interest. The log likelihood equations and MODEL statement in NLMIXED assume the response variable is named “y”; rather than recode the statements, it is likely easier to rename the response variable when accessing the SAS data set, as shown on the PROC statement below. The process of constructing NLMIXED code includes entering a PARMS statement (for initial parameter values) and the linear predictor eta for a given data set, plus adding a variable name for the dispersion parameter, if it exists. From the linear predictor(s), compute the inverse link followed by the log-likelihood equations. The computations for many of these models can be checked with PROCs FMM, GENMOD, or GLIMMIX.

```
PROC NLMIXED DATA= <infile> (rename=(<response> = y));
PARMS <enter initial parameter estimates, including the dispersion param, if any > ;
 eta = < linear predictor equation > ;
mean = < linear function that relates eta to the mean, an inverse link function >
lglk = < Log Likelihood equation for the selected data distribution >;
MODEL y ~ GENERAL(lglk);
RUN;
```

## Discrete Distributions

**Bernoulli:** y is binary, coded numerically as 0 or 1

The probability p refers to the level of the response variable defined as success. The LgLk equation can have two forms: for the model y = 1 (descending) or for y =0 (ascending).

**Binomial:** y is number of successes in n independent trials

The probability p refers to the ratio of the two counts, y/n.

The following distributions evaluate overdispersed binomial data in logistic regression models.

**Zero Inflated Binomial:** y is the number of successes in n independent trials with excess 0's

**Random Clumped Binomial:** y is the number of successes in n trials

**Beta Binomial:** y successes in n trials with overdispersion parameter rho

**Multinomial:** $y_i = 0/1$ for i = 1 .. M, with M usually less than or equal to 5

Methods and examples for working with ordinal data with NLMIXED are described in High (2013).

**Poisson:** y is a count defined to be an integer greater than or equal to 0

**Truncated Poisson:** y is a count variable with no 0s represented by an integer greater or equal to 1

**Zero Inflated Poisson:** y is count represented by an integer greater than or equal to 0 with excess 0s present

In this model, a zero can be encountered either through the probability model or from the Poisson distribution.

**Poisson Hurdle:** y is an integer GE 0 with excess 0's

The Poisson hurdle model combines features of the zero-inflated and truncated Poisson models, where in the hurdle model a 0 can only occur with the probability model, not the Poisson counts.

**Negative Binomial:** y is a count, an integer greater than or equal to 0

**Truncated Negative Binomial:** y is a count, an integer greater than or equal to 1

**Zero-Inflated Negative Binomial:** y is an integer greater than or equal to 0 with excess 0's present

A zero can be encountered either through the probability model or from the negative binomial distribution.

**Negative Binomial Hurdle:** y is an integer GE 0 with excess 0's

The negative binomial hurdle model combines the features of the zero-inflated and truncated negative binomial models, where in the hurdle model a 0 can only occur with the probability model, and not the negative binomial counts.

## Continuous Distributions

**Normal:** `y is continuous and unbounded`

**Normal with censored values:** `y is continuous between lower bound a and upper bound b`

**Beta:** y bounded between 0 and 1, endpoints not included

The beta distribution is primarily used with percentiles or ratios where the non-zero numerator is less than the denominator so the ratio is bounded between 0 and 1, not including the endpoints. The normal distribution may work reasonably well for values that are not close the endpoints. However, as the minimum values approach 0 and the maximum values approach 1, this type of bounded data modeled with an unbounded normal distribution may not be a good choice.

**Zero Inflated (Altered) Beta:** y bounded between 0 and 1, y=0 is included

This example has the most extensive code with two categorical factors and their interaction. It shows how PROC TRANSREG produces dummy coded values, plus NLMIXED computes Ls-Means for the interaction. When running this type of model in NLMIXED, the same variable may appear in any of the three linear predictors: the probability model, the mean, and the dispersion parameter. To avoid warnings in the LOG window, copies of the same variable can be placed in the data set, and each variable entered in the respective linear equation, as demonstrated in the SAS file example.

**Exponential:** y is greater than 0

**Truncated Exponential:** (a < y < b) with a > 0, b > a

In the examples, a left-truncated exponential distribution is shown with a > 0 and b=infinity.

**Right Censored Exponential:** 0 < y with cens=0 and y is less than observed y with cens=1

**Gamma:** y is greater than 0

For GENMOD and FMM the dispersion parameter is entered as phi. FOR GLIMMIX, the dispersion parameter is expressed as the inverse 1/phi. This parameterization of the same model differs in order to achieve a variance function suitable for mixed model analysis.

**Left Truncated Gamma:** y is greater than a specified lower bound a > 0
**Right Censored Gamma:** y > 0 with cens=0 if y is final; cens=1 if y is less than the unobserved value

Code for a truncated gamma distribution is not presented; PROC FMM does not have a truncated gamma option for verification; however, comparison of results for a right-censored data from LIFEREG and NLMIXED is presented with a likelihood equation using the LOGSDF function.

**Log-Logistic:** y > 0
**Right Censored Log-Logistic:** y > 0 with cens=0 if y is final; cens=1 if y is less than the unobserved value

**Weibull:** y > 0
**Right Censored Weibull:** y > 0 with cens=0 if y is final and with cens=1 if y is less than the unobserved value