Paper PH-05–2016

Fitting Complex Statistical Models with PROCs NLMIXED and MCMC

Robin High, 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 which includes proportions and counts. Procedures such as PROCs LOGISTIC, GENMOD, COUNTREG, GLIMMIX, and FMM, among others, offer a flexible range of analysis options to work with data from a variety of distributions and also with correlated or clustered data. Recent versions of SAS have procedures that model zero-inflated and truncated data. This paper demonstrates how statements from PROC NLMIXED can be written to match the output results from these procedures, including the LsMeans. Situations may arise where the flexibility of programming statements of PROC NLMIXED are needed such as for zero inflated, truncated counts, or proportions with random effects. A useful application of these coding techniques is that several programming statements from NLMIXED can be directly transferred into PROC MCMC to perform analyses from a Bayesian perspective with these various types of complex models.

Introduction

Statistical modeling can be a complex process where 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 a dist=<option>. One procedure which can often be programmed to work with these somewhat infrequent, yet important, modeling situations is PROC NLMIXED; however, programming techniques are needed to write the SAS code to do so.

One way to learn NLMIXED is to produce results with basic generalized linear models that match output with PROC GLIMMIX. One objective of this paper is to first demonstrate how to write code in NLMIXED that matches what PROCs FMM, GENMOD, and GLIMMIX produce, and then observe how NLMIXED statements can be enhanced to provide enhanced features not available with them. Also, a secondary objective is to observe that the NLMIXED statements described here can be copied directly into PROC MCMC to produce results from a Bayesian perspective.

Motivating Example

Assume the response variable (notated as y for all types of distributions) is a ratio of two numbers where the denominator is expected to be larger than the non-zero numerator. Thus, all data will be bounded between 0 and 1. The data set contains expense ratios as the response variable collected from hospitals spread across a few states in various regions of the US. Of interest is the relationship of the ratio to three categorical variables and one continuous variable, the fixed effects of the mode:

Label	Variable Name	Model Coef	Levels	Values
Hospital Size	HspSze_	b1	3	L=Large / M=Medium / S=Small
Location	RUrb_	b2	2	R=Rural / U=Urban
Tax Year	txyr_	b3	3	1/2/3
Unemployment Rate	unempl	b4		Continuous
Financial Ratio	y			Continuous, bounded 0 LE y LT 1

Table 1. Factors and characteristics of values of the model

The variable names for the three classification factors in the second column of the table all end with the underscore for reasons that will become evident in the coding examples. In general, whenever naming variables to be written into programming statements in GLIMMIX or NLMIXED, avoid variable names that begin with an underscore _ since the name may conflict with internal variables produced by the procedure. The column labeled “Model Coef” is the first two letters of the name of the variable in the linear predictor equation for the model with PROC NLMIXED.

The ratios collected from each hospital for three successive years are defined on each record with the subject variable “facility” which is to be treated as a random effect, that is, the facilities selected are interchangeable with a much larger number of facilities not selected. The ratios of interest are generally expected to be small, with many close to 0 and in some situations will equal 0 if the particular expense was not incurred in a given year. Since the ratios are expected to be near to the lower bound of 0, perhaps with multiple 0s, fitting a model based on a normal distribution for the response may not be the best choice. For data bounded between 0 and 1, a beta distribution is an alternative; if 0s are present, a zero-inflated beta model may be utilized (see Smithson, 2012, Chapter 6). (The presence of both 0s and 1s is also a possibility, though will not be covered here.)

PROC GLIMMIX Code

With one observation per organization PROCs GLIMMIX or FMM would be choices to model these data collected as bounded ratios between 0 and 1 modeled with a beta distribution. If the objective is to compare these non-zero ratios over a period of several years PROCs GLIMMIX would also work:

```
PROC GLIMMIX DATA=dt1 method=quad;
CLASS HspSze_ RUrb_ facility txyr_ ;
MODEL y = HspSze_ | RUrb_ txyr_ unempl / dist=beta solution;
RANDOM intercept / subject=facility ;
LSMEANS hspsze_ * RUrb_ txyr_ / ilink cl at unempl=.05 e;
TITLE1 "ToPtToEx";
RUN;
```

If the data set has several observations with responses of 0, PROC FMM offers the choice of a zero-inflated model whereas GLIMMIX does not. Both procedures only compute one precision parameter. With responses of 0, multiple observations from the same organization, and the capability to model the precision, PROC NLMIXED can be utilized.

PROC NLMIXED

The next few sections illustrate the programming statements that form the basis to generate results for data analysis from various response distributions with PROC NLMIXED. The framework for coding linear statistical models consists of the following essential statements:

```
PROC NLMIXED DATA=indata < options > ;
PARMS < initial parameter values > ;
* enter programming statements;
eta = < linear predictors for the probability model, the mean, and precision > ;
mean = G(eta) ; * G is the inverse link function;
LgLk=log(f(x)); * Log-Likelihood equation of the probability density function f(x);
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, as an example will later demonstrate. Most relevant here, are the equations defining the linear predictor, inverse link, and log likelihood statements all have the same programming syntax. A brief description of these programming statements follow.

The PARMS Statement

Initial values of the parameter estimates are set to 1 by default (variable names in equations which are not included in the input data set). 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 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, though it is often not needed. 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;
```

Applying initial parameter values with a data set will be illustrated in a subsequent example.

The MODEL Statement

In NLMIXED, seven response distributions can be specified with the MODEL statement. A specific name of the desired distribution with its parameters will model the response such as:

MODEL y ~ < distribution > ;

Distribution	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

However, since the objective of this paper is to describe how NLMIXED can be utilized for more complex modeling situations, such as mixtures of distributions, truncated distributions, or for distributions not available from this list, the statement:

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

will appear in all examples. 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 SAS code for the log likelihood equations of several commonly applied probability models are given in the Appendix and can be copied directly into the PROC NLMIXED code along with the other programming statements.

Constructing Linear Predictors

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

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

The x_i variables refer to a dummy coded main effects, their interactions, along with names for continuous data. For consistency and ease of programming, parameter estimates 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 levels of the explanatory data. Although the first two letters of the proc name NLMIXED imply it is designed to work with non-linear equations (see Kurada and the NLMIXED documentation for examples), the statistical models demonstrated here will have a linear equation which serves as the basis for model estimation. One important characteristic of the linear predictor is that the computed values should freely cover the entire practical range of real numbers, both negative and positive.

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

The linear predictor (eta) may be constructed with the parameter estimates multiplied by indicator variables for the explanatory data without any modifications to the input data set:

```
eta = b0 +
      b1a*(HspSze_ = 'L') + b1b*(HspSze_ = 'M') +
      b2*(RUrb_ = 'R') +
      b3a*(Txyr_ = '1') + b3b*(txyr_ = '2') +
      b4*unempl;
```

The rule of “last” sorted value as the reference category is applied here (resulting in a coefficient of 0 for each reference category); “S” is the reference for Hospital Size, “U” for Location, and “3” for tax year. For smaller models this manner of writing the linear equation clearly identifies factor levels of categorical data. It assumes none of the predictor variables will have missing data (if so, a 0 will replace a missing value). A WHERE statement placed after the PROC statement can be entered to exclude all records with missing categorical data:

```
WHERE MISSING(Hspsze_)=0 AND MISSING(RUrb_)=0 and MISSING(txyr_)=0 ;
```

A linear predictor of this type generally is simple to write for models having only main effects. In the example analysis problem, an extension is to include the interaction between hospital size (3 levels) and location (2 levels) which could be added to the linear predictor with two additional terms coded as:

```
+ b1a_b2*(hspsze_='L' AND RUrb_='R') + b1b_b2*(hspsze_='M' AND RUrb_='R')
```

Adding interaction terms linking conditions with AND (as in a DATA step) may cause estimation problems as shown on the output by very large gradient values and bogus or missing estimates, which essentially indicates the optimization could not be completed (also suggested by warnings written in the log window). A way to avoid this problem is to enter the test of each factor level within () as a separate condition:

```
+ b1a_b2*(hspsze_='L')*(RUrb_='R') + b1b_b2*(hspsze_='M')*(RUrb_='R')
```

This problem can also be resolved by entering dummy coded data for the classification factors into the data set externally. These conversions could be coded in a DATA step, though for models with several categorical predictor variables, including interactions, dummy coded categorical data can be produced more efficiently. For example, 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 them to parameters. For models computed with PROC NLMIXED, the conversion of categorical data into dummy codes may work best with PROC TRANSREG which produces new variable names which combine the variable name(s) and their coded levels (the reason for the trailing _ on the name and specifying one-letter values for the discrete levels):

```
PROC TRANSREG DATA=dt1 design;
MODEL CLASS(HspSze_ RUrb_ hspsze_*Rurb_ txyr_ / ZERO=last);
ID facility unempl y;
OUTPUT out=dt1I(drop=_type_ intercept );
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 where the option ZERO=last indicates the highest sorted value is the reference category. Other data including identification codes and numerical values (including the response variable and 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 existence of the macro variable &_trgind which defines the name and specific level of each dummy coded categorical variable. The names can be extracted by printing the list of variable names into the LOG window:

```
%PUT &_trgind;

HspSze_L HspSze_M RUrb_R HspSze_LRUrb_R HspSze_MRUrb_R txyr_1 txyr_2
```

The first part of the new variable name is the original name (e.g., HspSze_) immediately followed by the dummy coded value (L or M; level S is the reference category and is not given a column name). The macro variable can be used as it is given or elements extracted as needed.

```
PROC PRINT DATA=dt1I(obs=6) NOObs;
VAR facility &_trgind unempl y hspsze_ rurb_ txyr_;
Run;
        Hsp    Hsp            HspSze_  HspSze_                    Un          Hsp
facility Sze_L  Sze_M  RUrb_R  LRUrb_R  MRUrb_R  txyr_1 txyr_2  Empl      y  Sze_   RUrb_ txyr_

 001       1      0       0       0        0       1      0      3.7  0.040  L      U      1
 001       1      0       0       0        0       0      1      5.1  0.046  L      U      2
 001       1      0       0       0        0       0      0      5.3  0.031  L      U      3
 002       0      0       1       0        0       1      0      3.7  0.036  S      R      1
 002       0      0       1       0        0       0      1      5.1  0.046  S      R      2
 002       0      0       1       0        0       0      0      5.3  0.026  S      R      3

Coef:    b1a    b1b      b2    b1a_b2   b1b_b2    b3a    b3b     b4       y
```

An efficient way to enter the linear predictor to the PROC NLMIXED code is two ARRAY statements, one for the predictor variables (dt) and the other for the variable names of the parameter estimates (bt) which are then multiplied and summed. The initial value is equal to the intercept:

```
ARRAY dt[8] &_trgind. UnEmpl ;
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;
```

The value of eta is initially set to the intercept (b0) followed by the accumulation of the sum with the product of the dummy variables and covariates multiplied by the values of the parameter estimates. For random effect models, a component v is also added to eta (which is defined as having mean 0 and variance sv2 by the 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, which are inevitable, are acceptable.

Initial Parameter Values

A common practice is to construct a full model with all variables entered into a complex model and then read a note in the log window that the model did not converge properly. In complex statistical models, entering initial coefficient estimates that approximate the optimal solution in sign and magnitude can influence whether or not the model converges to a global maximum, as well as the amount of time required to do so. PROC NLMIXED itself can efficiently compute initial values, since the ODS file for parameter estimates has the necessary variable names and contents. A model may be written with the respective linear predictor, inverse link, and scale parameter such as the negative binomial, but instead estimate parameters with a normal distribution:

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

The parameter estimates are saved in an output file which is entered as an option following a / in the PARMS statement as the starting values for a negative binomial count model:

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

Initial parameter estimates can also be derived externally from another procedure (e.g., a zero-inflated model); however, the file with the estimates will need to be modified in order to give the defined parameter names in NLMIXED, as demonstrated here with PROC FMM:

```
ODS OUTPUT ParameterEstimates=prms1(keep=parameter estimate
                                  rename=(parameter=prm) where=(estimate NE 0));
ODS LISTING close;
PROC FMM data=dt1;
CLASS hspsze_ RUrb_ txyr_ ;
MODEL y = hspsze_ RUrb_ hspsze_*RUrb_ txyr_ unempl / dist=beta link=logit;
RUN;
ods listing;

DATA init_prms; SET prms1;
LENGTH parameter $6;
KEEP parameter estimate ;
ARRAY pnms{10} $6 ('b0' 'b1a' 'b1b' 'b2' 'b1a_b2' 'b1b_b2' 'b3a' 'b3b' 'b4' 'd0');
if _n_ = 10 then estimate = log(1/estimate);
parameter = pnms{_n_};
RUN;
```

A parameter estimate from another procedure may need a functional transformation, such as the precision parameter, d0. The PARMS statement then includes initial estimates for any new variables (i.e., extra precision parameters or a random effect):

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

In this example, the initial data set values is in the narrow format with one record for each parameter defined by the variable names “parameter” and “estimate”. A wide-form of the data set with one record consists of the estimates defined with the ten parameters as the variable names.

ANOVA Table

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

```
CONTRAST "b1: Hosp Size"        b1a - 0, b1b – 0       df=219;
CONTRAST "b2: Location"         b2 – 0                 df=219;
CONTRAST "b1_b2: HspSze x RUrb" b1a_b2 - 0, b1b_b2 – 0 df=219;
CONTRAST "b3: Taxyear"          b3a - 0, b3b - 0;
CONTRAST "b4: Unempl"           b4 - 0;
```

The descriptive content of each effect can be formed by extracting the variable names from the &_trgind macro variable 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 will equal the one given on the parameter estimate table). Note that hospital size and taxyear both have three levels, so they are coded with two dummy variables and have two degrees of freedom. Also the two-factor interaction consists of 2 additional coefficients to be estimated. Entering df=### is optional and is only necessary to change the default value. Note that with more complex situations, such as models with random effects, these tests will differ from GLIMMIX since the more complicated means of computing F values and degrees of freedom is not available in NLMIXED.

LSMEANS

LSMEANS for the levels of the categorical fixed effects computed with PROC GLIMMIX only needs to have the name of the effect of interest on the LSMEANS statement. The values computed 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 hspsze_ * RUrb_ txyr_ / ilink cl at unempl=.05 e;
```

The coefficient matrix (requested with the option, the letter e) is saved with ODS:

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

The contents of the file lsmcoef include a variable called Lmatrix, an increasing integer for each factor for which LsMeans are requested. In this case there are one set of coefficients for the HspSze_RUrb interaction and a second second set of LsMeans for taxyear. In order to utilize these coefficients in PROC NLMIXED, the file can be transposed and fractions converted to integers. In this case the least common denominator for the 3-level variable hspsze_ and the 2-level variable RUrb_ is **6** (the place for this multiplication is emphasized in bold print):

```
DATA lsmcoef; SET lsmcoef; LENGTH effct $20; DROP i;
effct=COMPRESS(catt(effect,hspsze_,Rurb_,txyr_),'*'); *Combine the effect and levels;
array rw{6} row1 - row6;
DO i = 1 to 6; if rw{i} = . then rw{i} = 0; rw{i} = rw{i}*6; end;

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

PROC PRINT DATA=lsmcoef NOObs; run;
```

Due to the long variable names, they 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 HspSze_* RUrb_ interaction is:

```
    Intcp  HspSze_     RUrb_    HspSze_*RUrb_       Txyr_   unempl
Row1  6     6  0  0     6  0    6  0  0  0  0  0     2  2  2   0.3
Row2  6     6  0  0     0  6    0  6  0  0  0  0     2  2  2   0.3
Row3  6     0  6  0     6  0    0  0  6  0  0  0     2  2  2   0.3
Row4  6     0  6  0     0  6    0  0  0  6  0  0     2  2  2   0.3
Row5  6     0  0  6     6  0    0  0  0  0  6  0     2  2  2   0.3
Row6  6     0  0  6     0  6    0  0  0  0  0  6     2  2  2   0.3
```

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

```
ESTIMATE 'HR L/R' intercept 6 hspsze_ 6 0 0 RUrb_ 6 0 hspsze_*RUrb_ 6 0  0 0  0 0  txyr_ 2 2 2 unempl 0.3 / divisor=6;
ESTIMATE 'HR L/U' intercept 6 hspsze_ 6 0 0 RUrb_ 0 6 hspsze_*RUrb_ 0 6  0 0  0 0  txyr_ 2 2 2 unempl 0.3 / divisor=6;
ESTIMATE 'HR M/R' intercept 6 hspsze_ 0 6 0 RUrb_ 6 0 hspsze_*RUrb_ 0 0  6 0  0 0  txyr_ 2 2 2 unempl 0.3 / divisor=6;
ESTIMATE 'HR M/U' intercept 6 hspsze_ 0 6 0 RUrb_ 0 6 hspsze_*RUrb_ 0 0  0 6  0 0  txyr_ 2 2 2 unempl 0.3 / divisor=6;
ESTIMATE 'HR S/R' intercept 6 hspsze_ 0 0 6 RUrb_ 6 0 hspsze_*RUrb_ 0 0  0 0  6 0  txyr_ 2 2 2 unempl 0.3 / divisor=6;
ESTIMATE 'HR S/U' intercept 6 hspsze_ 0 0 6 RUrb_ 0 6 hspsze_*RUrb_ 0 0  0 0  0 6  txyr_ 2 2 2 unempl 0.3 / divisor=6;

ESTIMATE 'Txyr 1' intercept 6 hspsze_ 2 2 0 RUrb_ 3 3 hspsze_*RUrb_ 1 1  1 1  1 1  txyr_ 6 0 0 unempl 0.3 / divisor=6;
ESTIMATE 'Txyr 2' intercept 6 hspsze_ 2 2 0 RUrb_ 3 3 hspsze_*RUrb_ 1 1  1 1  1 1  txyr_ 0 6 0 unempl 0.3 / divisor=6;
ESTIMATE 'Txyr 3' intercept 6 hspsze_ 2 2 0 RUrb_ 3 3 hspsze_*RUrb_ 1 1  1 1  1 1  txyr_ 0 0 6 unempl 0.3 / 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 'HspSze: L / RU=R' (6*b0 + 6*b1a         + 6*b2 + 6*b1a_b2            + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'HspSze: L / RU=U' (6*b0 + 6*b1a                                      + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'HspSze: M / RU=R' (6*b0         + 6*b1b + 6*b2            + 6*b1b_b2 + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'HspSze: M / RU=U' (6*b0         + 6*b1b                              + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'HspSze: S / RU=R' (6*b0                 + 6*b2                       + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'HspSze: S / RU=U' (6*b0                                              + 2*b3a + 2*b3b + b4*0.3)/6;
ESTIMATE 'Taxyear 1'        (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2 + 6*b3a         + b4*0.3)/6;
ESTIMATE 'Taxyear 2'        (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2         + 6*b3b + b4*0.3)/6;
ESTIMATE 'Taxyear 3'        (6*b0 + 2*b1a + 2*b1b + 3*b2 + 1*b1a_b2 + 1*b1b_b2                 + b4*0.3)/6;
```

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

```
ESTIMATE 'Mean: HspSze: L / RU=R' 1 / (1 + EXP( -(lp)));
```

Looking ahead to PROC MCMC, the procedure does not have an ESTIMATE statement, yet computations can be made in this manner either by entering the equations within the procedure 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,sigmaF2) subject=facility;
```

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=facility(state) (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 much remain positive.

```
RANDOM v ~ normal(0,exp(2*logsig)) subject=facility;
```

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

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

Modeling Ratios with Zeros

If the data are ratios from independent observations which include 0s, PROC FMM can compute a zero-inflated beta model where the zero-altered component is specified with a PROBMODEL statement:

```
PROC FMM data=dt2;
CLASS hspsze_ RUrb_ txyr_ ;
MODEL y = hspsze RUrb_ hspsze*RUrb txyr unempl / dist=beta link=logit;
MODEL y = / DIST=constant(0);
PROBMODEL hspsze_ RUrb_ txyr_ unempl;
           * Probmodel is for the positive data so use link p= 1/(1 + EXP(lp));
TITLE "PROC FMM: zero inflated beta";
RUN;
```

PROC NLMIXED computes the zero-altered beta model with classification factors for both the zero-altered portion and the linear predictors for the mean and precision along with the log-likelihood of the zero inflated model with the following essential elements.

```
PROC NLMIXED DATA= < options> ;
PARMS < parameter initial values > ;
* Probability of 0;
eta_zr = < linear predictor of the probability model > ;
p_zr = G(eta_zr);  inverse link for the probability the response is 0;
* Mean;
eta = < linear predictor for the mean> ;
mean = G(eta) ; * inverse link for the mean;
* Precision ;
wd = d0 + < linear predictor > ;  * for precision;
phi = exp(-1*wd);
p = mu*phi; q = phi*(1 - mu);
Lglk = log(f(x)); * Log-Likelihood equation combines
                    the discrete and continuous pdfs (see below);
MODEL y ~ GENERAL(lglk);
RANDOM v ~ NORMAL(0,sg2) subject=<effect> ;
< ESTIMATE, CONTRAST statements >
RUN;
```

Despite the disadvantage of the details involved when coding the predictors, PROC NLMIXED offers features not available in PROC FMM, namely to work with correlated data through random effects with a RANDOM statement, modeling the precision with one or more explanatory data (see example statement above for wd), and placing boundary values on the parameter estimates for the probability model (not illustrated).

In the beta regression model for illustration there are now three linear predictors: one each for the probability of a 0, the mean, and precision. The first step is to make copies of desired fixed effects for the probability model (defined here with an initial letter P) in the input data set:

```
DATA dt2; SET dt2;
Ptxyr_   = txyr_ ;
Phspsze_ = hspsze_ ;
PRurb_   = RUrb_ ;
PUnEmpl  = UnEmpl ;
RUN;
```

Next, both sets of variables for the effects to be estimated are entered into the CLASS() option, with the variables for the probability model listed first, followed by variables for the mean. All ID and numerical variables are listed on the ID statement. The copies of the categorical data are dropped from the output data set.

```
PROC TRANSREG DATA=dt2 design;
MODEL CLASS(Phspsze_ PRurb_               Ptxyr_
             HspSze_  RUrb_  HspSze_*RUrb_ txyr_ / ZERO=last);
ID facility state y PUnempl UnEmpl;
OUTPUT OUT=dt2I(DROP = _type_ _name_ intercept Ptxyr_ Phspsze_ PRurb_ );
RUN;
```

The resulting macro variable now contains all the variable names and factor levels as before. The presence of the macro name in an ARRAY statement within NLMIXED poses two difficulties: the same warning about convergence and efficiency appears even if different variable names are accessed. Plus identifying the index of each variable with its coefficient requires extra vigilance. An approach is to access the variable names from the macro variable (e.g., with a **%PUT** statement to the log window) and extract only the variable names needed for each linear predictor. For the probability model:

```
ARRAY dtP[6] Phspsze_L Phspsze_M PRurb_R Ptxyr_1 Ptxyr_2 Punempl ;
ARRAY btP[6] bp1a      bp1b      bp2     bp3a    bp3b    bp4 ;
eta_zr=bp0; DO i = 1 to 6; eta_zr = eta_zr + btP[i]*dtP[i]; end;
p_0 = 1 /(1 + exp(-eta_zr));  * estimate probability the response is 0;
```

and for the linear predictor of the mean:

```
ARRAY dtM[8] HspSze_L HspSze_M RUrb_R HspSze_LRUrb_R HspSze_MRUrb_R txyr_1 txyr_2 unempl ;
ARRAY btM[8] b1a     b1b     b2     b1a_b2         b1b_b2       b3a    b3b   b4 ;
eta=b0; DO i = 1 to 8; eta = eta + btM[i]*dtM[i]; END;
```

When the same data set variable names are entered into the separate equations, NLMIXED produces a warning concerning convergence and efficiency:

```
WARNING: Multiple element aliasing in arrays dtM and dtP. Calculated derivatives may
be incorrect and run time will be less efficient.
```

The warning disappears when separate variables for each linear predictor (also in separate arrays) are made available, as entered in this example.

Precision (phi) is modeled with a third linear predictor where in this example, the variability of the outcome is assumed to be different between rural and urban locations:

```
wd = d0 + d2*(RUrb_ ='R');  * linear predictor for precision;
phi = exp(-1*wd);
```

For a zero-inflated model with a beta distribution, the log-likelihood equation comes from a mixture of distributions: for the zero model with probability p_0 and for the mean with probability (1-p_0):

```
mean = 1 /(1 + exp(-eta));
p = mean*phi; q = phi*(1 - mean);
IF y=0 then lglk = log(  p_0);
       else lglk = log(1-p_0) + lgamma(p+q) - lgamma(p) - lgamma(q)
                              + (p-1)*log(y) + (q-1)*log(1 - y);
```

The equations of the lsmeans for both the predictor of the probabilities and for the non-zero data are much the same as given above.

Assembling the Components

As stated in the introduction, the NLMIXED code presented here is nearly identical to statements that could be entered into PROC MCMC. The notable differences are the options placed on the PROC MCMC statement, the addition of non-informative PRIOR statements for each model parameter (having extremely large variances). Also, initial values must be specified in a PARMS statement (which has several alternative methods than NLMIXED provides, though it does not have the input data=<file> option for initial estimates). The complete NLMIXED code for a zero inflated beta model is listed below; however, here it is to be run with PROC MCMC with the addition of the two PRIOR statements:

```
PROC MCMC DATA=dt2I seed=21228 nmc=20000 nbi=5000 opti = quanew;
PARMS bp0 .1 bp1a .1 bp1b .1  bp2 .1                    bp4 .1
       b0 .1  b1a .1  b1b .1   b2 .1  b1ab2 .1 b1bb2 .1  b4 .1
       d0 -1 d2 .1;
PRIOR b: ~ normal(mean=0,var=1E6);  * PRIORs only needed in MCMC;
PRIOR d: ~ normal(mean=0,var=1E6);

ARRAY dtP[4] Phspsze_L Phspsze_M PRurb_R Punempl ;
ARRAY btP[4] bp1a bp1b bp2 bp4 ;
eta_zr=bp0; DO i = 1 to 4; eta_zr = eta_zr + btP[i]*dtP[i]; end;
p_0 = 1 / (1 + exp(eta_zr)); * FMM estimates prob of Y > 0, so do not enter –eta_zr;

ARRAY dtM[6] HspSze_L HspSze_M RUrb_R HspSze_LRUrb_R HspSze_MRUrb_R unempl ;
ARRAY btM[6] b1a b1b b2 b1ab2 b1bb2 b4 ;
eta=b0; DO i = 1 to 6; eta = eta + btM[i]*dtM[i]; END;
mean = 1 /(1 + exp(-eta));

wd = d0 + d2*(RUrb_='R');
phi = exp(-1*wd);
p = mean*phi; q = phi*(1 - mean);
IF y=0 then lglk = log(  p_0);
       ELSE lglk = log(1-p_0) + lgamma(p+q) - lgamma(p) - lgamma(q)
                 + (p-1)*log(y) + (q-1)*log(1 - y);
MODEL y ~ general(lglk);
RUN;
```

When these statements are entered within NLMIXED the parameter estimates match results from PROC FMM. With PROC MCMC the parameter estimates are close because everything in MCMC is random. Adding a RANDOM statement in both procedures to work with multiple measurements greatly increases the computational resources needed, especially for PROC MCMC.

With complex models, computational issues abound with both NLMIXED and MCMC, so check the respective documentation for the various methods that assist with convergence which may greatly increase the run 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.

Smithson, M, and Merkle, Edgar, (2014) Generalized Linear Models for Categorical and Continuous Limited Dependent Variables, CRC Press, Boca Raton.

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

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

APPENDIX

Link Functions

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

```
   Function         Link: eta=G(E(y))       Inverse Link (as coded in NLMIXED)
Continuous Responses
   Identity         eta                     mu = eta
   Reciprocal       eta =1 / mu             mu = 1 / eta
   Log              eta = LOG(mu)           mu = EXP(eta)
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)
   LogLog           eta = -(LOG(-LOG(p)))   p = EXP(-EXP(-eta))
                    eta =  LOG(-LOG(p))     p = EXP(-EXP(eta))
   Complimentary
   LogLog           eta=LOG(-LOG(1-p))      p = 1 - EXP(-EXP(eta))
```

The linear predictor for each link is eta (or eta_zr in the case of a model with excess 0s), a function of the model coefficients and the explanatory data, as defined in the main body of the text:

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

The inverse links are called mu (for continuous data) and p (for data bounded between 0 and 1). In the examples to follow, the links commonly applied to the various distributions are given, though others are possible.

Probability Distributions

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

Construct PROC NLMIXED Code

For several discrete and continuous distributions the link functions (multiple options may exist) and the log-likelihood equations are provided below. The process of constructing NLMIXED code includes entering a PARMS statement and linear predictor (as described in the text) for a given data set, from which are computed the inverse link and log-likelihood equations. Most of the computations for these models can be checked with PROCs FMM or GLIMMIX.

```
PROC NLMIXED DATA= <infile>;
PARMS <enter initial parameter estimates> ;
Eta = < linear predictor > ;
< enter inverse link and lglk equations from a distribution below
  that matches data characteristics >;
MODEL y ~ GENERAL(lglk);
RUN;
```

Discrete Distributions

Bernoulli (y = 0 or 1)

```
p = 1 /(1 + exp(-eta)); * logit link;
p = 1 - EXP(-EXP(eta)); * complementary log log link;

lglk = (y=1)*log(p) + (y=0)*log(1-p) ; * y descending;
lglk = (y=0)*log(p) + (y=1)*log(1-p) ; * y ascending;
```

where p is the probability of a success. For the first lglk equation, p models the probability y = 1, for the second lglk equation, p models the probability y=0.

Binomial (y is number of successes in n independent trials)

```
p = 1 /(1 + exp(-eta)); * logit link;
p = 1 - EXP(-EXP(eta)); * complementary log log link;

lglk = y*log(p) + (n-y)*log(1-p) + lgamma(n+1) - lgamma(y+1) - lgamma(n-y+1) ;
        * -2*LL matches GLIMMIX;
```

In this formula the lgamma function is invoked when the gamma function appears in the probability density. Note that for PROC GENMOD the portions with n and/or y only are not present:

```
lglk = y*log(p) + (n-y)*log(1-p);
```

Beta Binomial (y successes in n trials with overdispersion parameter rho)

```
linr = a0;
rho = 1/(1+exp(-linr));
phi = (1-(rho**2)) / (rho**2);

   p = 1/(1+exp(-eta));
lglk =   lgamma(n+1) - lgamma(y+1)
       - lgamma(n - y + 1)
       + lgamma(phi) - lgamma(n+phi) + lgamma(y+(phi*p))
       + lgamma((n-y) + phi*(1-p)) - lgamma(phi*p)
       - lgamma(phi*(1-p) );
```

Multinomial (yi = 0/ 1 for i = 1 .. M, usually M LE 5)

With M=4, four response variables, y1, y2, y3, y4, each coded as 0/1:

```
lglk = LOG( (p1**(y1))*(p2**(y2))*(p3**(y3))*(p4**(y4)) ) ;

or an equivalent formula is:
]
lglk = y1*LOG(p1) + y2*LOG(p2) + y3*LOG(p3) + y4*LOG(p4);
```

With M=4, p1 .. p4 are previously defined so that each pi > 0 and SUM pi = 1.

Alternatively, if one response variable y is coded with four numerical values in ascending order, y=1, 2, 3, 4, then enter a version based on one response variable:

```
lglk = LOG( (p1**(y=1))*(p2**(y=2))*(p3**(y=3))*(p4**(y=4)) ) ;
lglk = (y=1)*log(p1) + (y=2)*Log(p2) + (y=3)*Log(p3) + (y=4)*LOG(p4); * y ascending;
```

for y in descending order:

```
lglk = (y=4)*log(p1) + (y=3)*Log(p2) + (y=2)*Log(p3) + (y=1)*LOG(p4); * y descending;
```

The link functions for ordinal data models are complex, as defined in the appendix to High (2013). In summary, multiple linear predictors (eta_i), either for the proportional odds or non-proportion odds models, with one of the respective link functions are needed:

```
* Cumulative logits;
cp1 = 1 / (1 + exp(-eta1));
cp2 = 1 / (1 + exp(-eta2));
cp3 = 1 / (1 + exp(-eta3));
p1 = cp1; p2 = cp2 - cp1; p3 = cp3 - cp2; p4 = 1 - cp3;

* Adjacent logits;
* estimate probabilities in the ascending order;
tot = 1 + exp(eta1) + exp(eta1+eta2) + exp(eta1+eta2+eta3);
p1 = 1 / tot;
p2 = exp(eta1)*p1;
p3 = exp(eta2)*p2;
p4 = exp(eta3)*p3;

* Continuation ratio (ascending order assumed, since categories ordered 1,2,3,4);
p1 =  1 / (1 + exp(-eta1));
p2 = (1 / (1 + exp(-eta2))) * (1 - p1);
p3 = (1 / (1 + exp(-eta3))) * (1 - p1 - p2);
p4 = 1 - (p1 + p2 + p3);
```

Poisson (y is an integer GE 0)

```
mu = exp(eta);
lglk= y*LOG(mu) - mu  - LGAMMA(y+1);
```

Truncated Poisson (y is an integer GE 1)

```
mu = exp(eta);
lglk= y*LOG(mu) - mu  - lgamma(y+1) /* poisson loglikelihood*/
      - LOG(1-EXP(-mu));            /* divide by (1-Prob(y=0)) */
```

An equivalent equation for the log likelihood is:

```
lglk = y*LOG(mu) - LOG(EXP(mu)-1) - LGAMMA(y+1);
```

Zero Inflated Poisson (y is an integer GE 0 with excess 0s)

A zero can be encountered either through the probability model or from the Poisson distribution

```
p_zr = 1 / (1 + EXP(-eta_zr));
mu = EXP(eta);
IF y = 0 THEN lglk = LOG(p_zr + (1-p_zr)*EXP(-mu));
         ELSE lglk = LOG(1-p_zr) + y*LOG(mu) - mu - LGAMMA(y+1);
```

Poisson Hurdle (y is an integer GE 0 with excess 0s)

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.

```
p_zr = 1 / (1 + EXP(-eta_zr));
mu = EXP(eta);
IF y = 0 THEN lglk = LOG(p_zr);
         ELSE lglk = LOG(1-p_zr) + y*LOG(mu) - mu - LGAMMA(y+1)
                                 - LOG(1-EXP(-mu)); /* divide by (1-Prob(y=0)) */
```

Negative Binomial (y is an integer GE 0)

```
mean = exp(eta);
lglk = y*log(phi*mean) - (y+(1/phi))*log(1+(phi*mean))
      + lgamma(y+(1/phi)) - lgamma(1/phi) - lgamma(y+1) ;
```

where phi is the dispersion parameter greater than 0.

Truncated Negative Binomial (y is an integer GE 1)

```
mean = exp(eta);
lglk = y*LOG(phi*mean) - (y+(1/phi))*LOG(1+(phi*mean))
      + lgamma(y+(1/phi)) - (lgamma(1/phi) + lgamma(y+1)) /*neg bin loglikelihood*/
      - LOG(1-(((phi*mean)+1)**(-1/phi)) );               /* divide by (1 - P(y=0))*/
```

where phi is the dispersion parameter.

Zero-Inflated Negative Binomial (y is an integer GE 0)

A zero may occur either through the probability model or from the negative binomial distribution.

```
p_zr = 1 / (1 + EXP(-eta_zr));
mean = EXP(eta);

IF y = 0 THEN lglk = LOG(p_zr + (1-p_zr)*(1+(phi*mean))**(-1/phi));
         ELSE lglk = LOG(1-p_zr) + y*LOG(phi*mean) - (y+(1/phi))*LOG(1+(phi*mean))
                                 + lgamma(y+(1/phi)) - (lgamma(y+1) + lgamma(1/phi));
```

Negative Binomial Hurdle (y is an integer GE 0 with excess 0s)

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

```
p_zr = 1 / (1 + EXP(-eta_zr));
mean = EXP(eta);
IF y = 0 THEN lglk = LOG(  p_zr);
         ELSE lglk = LOG(1-p_zr) + y*LOG(phi*mean) - (y+(1/phi))*LOG(1+(phi*mean))
                                 + lgamma(y+(1/phi)) - (lgamma(y+1) + lgamma(1/phi))
                       - LOG(1-((1+(phi*mean))**(-1/phi))); /* divide by (1-P(y=0))*/
```

where phi is the dispersion parameter.

Continuous Distributions

Normal (y is unbounded)

```
mean = eta;
lglk = -0.5*( ( ((y-mean)**2)/sigma2) + LOG(sigma2) + log(2*3.1415926536) );
```

where the mean and variance (sigma2) are specified with the constant 3.14159…. is pi.

Beta (y is bounded between 0 and 1, endpoints not included)

```
mean = 1 / (1+exp(-eta)); * inverse for logit link;
wd   = d0 + d1*x1;        * linear predictor for the precision parameter;
phi  = exp(-1*wd);
lglk = LGAMMA(phi) - LGAMMA(mean*phi) - LGAMMA((1-mean)*phi)
       + ((mean*phi)-1)*log(y) + (((1-mean)*phi)-1) * LOG(1-y);
```

Exponential (y greater than 0)

```
mean = exp(eta); * log link;
lglk = -LOG(mean) - y/mean;
```

Gamma (y greater than 0)

```
mean = eta;      * identity link;
mean = exp(eta); * log link;
mean = 1/eta;    * inverse power;

lglk = phi*LOG((y*phi)/mean) - (y*phi)/mean - LOG(y) - LGAMMA(phi);* matches GLIMMIX;
```

where phi is the dispersion parameter reported as 1/phi. This parameterization from GLIMMIX differs from GENMOD and FMM procedures in order to achieve a variance function suitable for mixed model analysis.

An alternative form that matches PROCs GENMOD and FMM:

```
lglk = (1/phi)*LOG(y/(mean*phi)) - y/(mean*phi) - LOG(y) - LGAMMA(1/phi);
```

which is the log-likelihood equation from PROCs GENMOD and FMM (not printed this way in FMM documentation)

Weibull (y greater than 0 for the two-parameter distribution)

```
  mean = 1/exp(-eta);
lglk = -((phi-1)/phi) * LOG(y/mean) - log(mean*phi) - EXP( LOG(y/mean)/phi);
```

with the mean and scale parameter phi.