Paper AS14

GLMM and GEE for Categorical Repeated Measures – Linking SAS and R

Miriam Amor, Veramed, Madrid, Spain

ABSTRACT

Since R started being used in clinical trials, hybrid SAS and R programming has become a reality. Collaborative initiatives, such as like PHUSE's CAMIS working group, are leading the way in facilitating the alignment between different languages.

In statistics, repeated measures are frequently analysed, with MMRM models being well-known models to analyse continuous variables. However, when the outcome is categorical, different methodologies are needed. Statistical methods that rely on a link function such as GEE and GLMM (using Laplace approximation, PQL or GHQ) become key alternatives. While SAS provides procedures like GENMOD and GLIMMIX, R offers functions as glmer, glmmPQL and geeglm, and the combination of both is the perfect toolkit to analyse categorical variables with repeated measures.

This presentation aims to show how to use R and SAS to program different models when the outcome is a categorical variable (binomial and multinomial) with repeated measures.

INTRODUCTION

The use of open-source software in the industry is a present reality. R has gained significant traction and is often routinely used alongside SAS. As a result, modern programmers are expected to understand the nuances and potential discrepancies in statistical outputs across these tools, especially in statistical models.

To support this transition, the PHUSE CAMIS (Comparing Analysis Method Implementations in Software) initiative has developed a repository that documents and compares statistical methodologies implemented in SAS, R, and Python.

One area where these differences are particularly relevant is the analysis of longitudinal data. While Mixed Models for Repeated Measures (MMRM) are widely known for continuous outcomes, categorical outcomes require alternative approaches such as Generalized Linear Mixed Models (GLMM) and Generalized Estimating Equations (GEE).

This paper explores how GLMM and GEE models for dichotomous outcomes under the frequentist framework are programmed in both SAS and R, highlighting key differences in syntax and default settings. Where feasible, strategies are proposed to align results across platforms to ensure consistency and reproducibility Additionally, this paper examines extensions of these models for outcomes with more than two categories are examined, including both nominal and ordinal outcomes.

GEE AND GLMM: AN OVERVIEW

Both GLMM and GEE methods extend the Generalized Linear Model (GLM) framework using link functions that relate the predictors to transformed outcome variable. For dichotomous response variables, the link functions is commonly the probit (in case of rare events complementary log-log may be preferable). For outcomes with more than two categories of response the cumulative link function is used in case of ordinal variables and generalized logit for nominal variables.

Though both methods are based on the link function, the modelling approach differs. Table 1 summarizes differences between the two models.

Table 1. Comparison of GEE and GLMM

	GEE	GLMM
Model Type	Marginal (Population-averaged effects)	Conditional (Subject-average effects)
Use case	Population-level inference	Individual-level inference
Random Effects	Not explicitly included	Explicitly Included
Correlation	Working correlation structure	Modelled via random effects
Estimation method	Quasi-likelihhod via iterative solving of estimating equations	Maximum Likelihood, based on approximations. - Gauss Hermite Quadrature - Laplace Penalized Quasi-Likelihood:

SAS PROCEDURES AND R FUNCTIONS FOR GEE AND GLMM: AN OVERVIEW

In SAS, PROC GEE or PROC GENMOD can be used to compute GEE models, while PROC GLIMMIX is commonly used to fit GLMMs, as it allows random effects.

In R, GEE models can be fitted using `gee::gee` or `geepack::geeglm`. For GLMMs, several packages offer functions with varying capabilities and optimization strategies. The most widely used include: `lme4::glmer`, `glmmTMB::glmmTMB`, `GLMMadaptive::mixed_model`, and `MASS::glmmPQL`. Table 2 summarizes which types of GLMMs can be fitted with each function.

Table 2. GLMM Model Types supported by SAS Procedures and R Functions

GLMM		QHQ	LAPLACE	PQL
SAS	GLIMMIX	✓	✓	✓
R	glmer	✓	✓	×
	glmTMB	×	✓	×
	mixed_model	✓	✓	×
	glmPQL	×	×	✓

Table 3 summarizes some key differences between SAS procedures and R functions, differences that are not limited to that list, as additional distinctions such as convergence criteria and optimization methods may exist.

Table 3. GLMM Model Types supported by SAS Procedures and R Functions

Model	Software	Procedure/ Function	SE	Correlation Matrix	DF	Link Function			
						Probit	Logit	Clogit	Glogit
GLMM	**SAS**	GLIMMIX	Model-Based	NA	BW	✓	✓	✓	✓
	R	glmer	Model-Based	NA	Inf.	✓	✓	✓	✓
		glmTMB	Model-Based	NA	Inf.	✓	✓	×	×
		mixed_model	Model-Based	NA	Inf.	✓	✓	×	×
		glmPQL	Model-Based	NA	Residual	×	✓	×	×
GEE	**SAS**	GENMOD	Sandwich	Independent	NA	✓	✓	✓	×
		GEE	Sandwich	Independent	NA	✓	✓	✓	✓
	R	gee	Sandwich	Independent	NA	✓	✓	×	×
		geeglm	Sandwich	Independent	NA	✓	✓	×	×
		ordLORgee	Sandwich	Exchangeable	NA	×	×	✓	×
		nomLORgee	Sandwich	Exchangeable	NA	×	×	×	✓

DF: Degrees of Freedom. BW: Between-Within. LR: Likelihood-ratio test. Clogit: Cumulative Logit. Glogit: General Logit. NA: Not applicable.

Given the variety of available modelling approaches, it is not surprising that results may differ depending on the method selected.

This paper aims to provide a comparative overview of these functions and procedures for fitting GLMM models in R and SAS through a series of illustrative examples. Particular attention is given to identifying functions that yield matching results.

EXAMPLE DATA

A SAS dataset of clinical trial data comparing two treatments for a respiratory disorder available in "Gee Model for Binary Data" in the SAS/STAT Sample Program Library [1] is used to create these examples. These data are from Stokes, Davis, and Koch (2000) [2].

Patients in each of two centers are randomly assigned to receive active treatment or placebo. During treatment, respiratory status, represented by the variable *outcome* (coded here as 0=poor, 1=good), is determined for each of four visits. The variables *center*, *treatment*, *sex*, and *baseline* (baseline respiratory status) are classification variables with two levels. The variable *age* (age at time of entry into the study) is a continuous variable.

To uniquely identify subjects, a new variable USUBJID was created by concatenating SITE and ID.
Variables TREATMENT, BASELINE, and VISIT were renamed to TRTP, BASE, and AVISITN. Two additional variables were generated using random values to simulate ordinal and nominal multi-level outcomes to assess the models with these outcomes (See Appendix 3).

GLMM and GEE models were fitted using treatment, visit and treatment by visit interaction as fixed effects, with random intercepts and subject-level effects.

In R, the variables OUTCOME, TRTP and AVISITN were converted to factors. Since the modeling functions use the first (alphabetically) level as the reference category, TRTP levels are ordered as 'P' (placebo) and 'A' (active), to ensure that placebo is used as the reference category in the models:

```
resp$trtp<-factor(resp$trtp, levels=c('P', 'A'))
resp$avisitn<-factor(resp$avisitn)
resp$outcome<-factor(resp$outcome)
```

GENERALIZED ESTIMATED EQUATIONS (GEE)

In SAS, GEE and GENMOD procedures can be used with identical syntax, while in R we found functions such as, `gee::gee` and `geepack::geeglm` are available, note that `geepack::geeglm` requires the response variable to be numeric (0/1), with 1 indicating the event of interest.

In both SAS and R, fitting a GEE model involves specifying the fixed effects, the binomial distribution for binary outcomes, the link function (specified as `link=logit`). The unique subject and the correlation structure are defined in repeated `subject…/type=(or corr=)…` statement in SAS, and through the `id=` and `corstr=` arguments in R. Both use an independent correlation matrix by default.

The Robust (Sandwich) S.E. is computed by default, while `gee::gee` also provides model-based (naïve) S.E. In SAS, model-based S.E. can be easily obtained by including the `modelese` option in the repeated statement.

Predicted probabilities and odds ratios (ORs) can be obtained in SAS using the `LSMEANS` statement:

- The ILINK option provides back-transformed predicted probabilities.
- The DIFF option, combined with either EXP or ODDSRATIO, computes ORs.
- CL computes confidence intervals.

In R, these estimates are not directly available from the model functions but can be derived using the well-known package `emmeans`. Table 4 summarizes the code used, the probabilities and ORs are computed for the treatment by visit interaction.

Table 4. GEE: SAS® and R code

SAS

```
proc gee data=resp;
   class trtp(ref="P") avisitn(ref='1') usubjid;
   model outcome(event='1') = trtp avisitn trtp*avisitn/ dist=bin link=logit;
   lsmeans trtp*avisitn/cl exp ilink oddsratio diff;
   repeated subject=usubjid/type=ind /*modelse [1]*/;
run;
```

R

```r
#Create numeric outcome (values: 1/0).
resp$outcome_num<-as.numeric(resp$outcome)-1

library(geepack)
model <- geeglm(outcome_num ~ trtp + avisitn +trtp*avisitn,
                id=usubjid,
                data=resp,
                family=binomial(link='logit'),
                corstr="independence")
```

```r
library(gee)
model<- gee(outcome ~ trtp + avisitn + trtp*avisitn,
            id=usubjid,
            data=resp,
            family=binomial(link='logit'),
            corstr = "independence")
```

[1] Option to display the model-based (naïve) S.E. along with the Sandwich S.E.

This example shows syntax with PROC GEE, and the same syntax can be used in PROC GENMOD. While the syntax is equivalent, minor differences in results may be found in the later decimal places. See Appendix 2 for details on comparison results across SAS procedures).

The estimated parameters obtained from the code above are presented in Table 5 SAS outputs have been processed for better visual comparison. Parameter estimates and SE match up to four decimal places, except for GEE results, where S.E. is rounded to three decimal places.

Table 5. GEE model: Estimated parameters. Results obtained with SAS® and R.

SAS

Parm	Level1	Level2	Estimate	StdErr	Z	ProbZ
Intercept			-0.0351	0.2649	-0.13	0.8946
trtp	A		0.8128	0.3950	2.06	0.0396
avisitn	2		-0.4292	0.2636	-1.63	0.1034
avisitn	3		-0.1408	0.2983	-0.47	0.6370
avisitn	4		-0.2118	0.2534	-0.84	0.4034
trtp*avisitn	A	2	0.5165	0.4104	1.26	0.2082
trtp*avisitn	A	3	0.3186	0.4284	0.74	0.4570
trtp*avisitn	A	4	-0.1140	0.3948	-0.29	0.7729

R

geepack::geeglm

	Estimate	Std.err	Wald	Pr(>|W|)
(Intercept)	-0.0351	0.2649	0.02	0.89
trtpA	0.8128	0.3950	4.23	0.04
avisitn2	-0.4292	0.2636	2.65	0.10
avisitn3	-0.1408	0.2983	0.22	0.64
avisitn4	-0.2118	0.2534	0.70	0.40
trtpA:avisitn2	0.5165	0.4104	1.58	0.21
trtpA:avisitn3	0.3186	0.4284	0.55	0.46
trtpA:avisitn4	-0.1140	0.3948	0.08	0.77

gee::gee

	Estimate	Naive S.E.	Naive z	Robust S.E.	Robust z
(Intercept)	-0.0351	0.267	-0.131	0.265	-0.132
trtpA	0.8128	0.399	2.039	0.395	2.058
avisitn2	-0.4292	0.383	-1.120	0.264	-1.628
avisitn3	-0.1408	0.379	-0.372	0.298	-0.472
avisitn4	-0.2118	0.380	-0.558	0.253	-0.836
trtpA:avisitn2	0.5165	0.570	0.906	0.410	1.258
trtpA:avisitn3	0.3186	0.570	0.559	0.428	0.744
trtpA:avisitn4	-0.1140	0.558	-0.204	0.395	-0.289

In this example, the LSMEANS option in SAS was used to obtain probabilities and treatment comparisons by visit. In R, similar functionality is provided by the `emmeans` and contrast functions from the `emmeans` package. The code below shows how to compute these estimates in R using results from `geepack::geeglm`, offering a parallel approach to the SAS implementation:

```r
library(emmeans)

#Get predicted probabilities for each treatment.
#Get predicted probabilities for each treatment, using the covariance matrix
computed by the model
prob <- emmeans(model, ~ trtp*avisitn, data=resp, vcov.method=vcov(model),
type='response')
prob

#Get differences between treatments by visit (option "revpairwise" is used to
compare A vs P)
diffs_mean<-emmeans(model, ~ trtp*avisitn, data=resp, vcov.method=vcov(model))
diffs <- contrast(diffs_mean,"revpairwise", simple="trtp")
diffs <- as.data.frame(diffs)
```

```r
#Calculate CI (alpha=0.05)
alpha<-0.05
z_crit <- qnorm(1 - alpha / 2)
diffs$low<-diffs$estimate - (z_crit*diffs$SE)
diffs$upp<-diffs$estimate + (z_crit*diffs$SE)

#Get OR applying exponential transformation;
or<-exp(diffs$estimate)
or_low<-exp(diffs$low)
or_upp<-exp(diffs$up)

#Get two-sided p-value
z <- diffs$estimate/diffs$SE
pvalue <- 2 * (1 - pnorm(z))

#Create a dataset with all the results
OR<-as.data.frame(cbind(diffs$avisitn, or,or_low, or_upp, z, round(pvalue,
digits=4)))
colnames(OR)<-c('avisit', 'OR', 'lower.CL', 'upper.CL', 'Z', 'p-value')
OR
```

The results of estimated probabilities and ORs, obtained by using `LSMEANS` statement in SAS and the `emmeans` functions in R, are displayed below, showing identical results up to four decimal places.

Table 6. GEE model: Probabilities and ORs. Results obtained with SAS® and R.

PROBABILITIES

SAS

Effect	trtp	avisitn	Mu	StdErrMu	LowerMu	UpperMu
trtp*avisitn	P	1	0.491	0.0662	0.365	0.619
trtp*avisitn	A	1	0.685	0.0632	0.551	0.794
trtp*avisitn	P	2	0.386	0.0645	0.269	0.517
trtp*avisitn	A	2	0.704	0.0621	0.570	0.810
trtp*avisitn	P	3	0.456	0.0660	0.332	0.585
trtp*avisitn	A	3	0.722	0.0610	0.589	0.825
trtp*avisitn	P	4	0.439	0.0657	0.316	0.569
trtp*avisitn	A	4	0.611	0.0663	0.476	0.731

R

```
> prob
 trtp avisitn  prob     SE  df lower.CL upper.CL
 P         1 0.491 0.0662 436    0.365    0.619
 A         1 0.685 0.0632 436    0.550    0.795
 P         2 0.386 0.0645 436    0.269    0.518
 A         2 0.704 0.0621 436    0.569    0.810
 P         3 0.456 0.0660 436    0.332    0.586
 A         3 0.722 0.0610 436    0.589    0.825
 P         4 0.439 0.0657 436    0.316    0.569
 A         4 0.611 0.0663 436    0.476    0.731
```

OR

SAS

Effect	trtp	_trtp	avisitn	OR	Lower OR	Upper OR	zValue	Probz
trtp*avisitn	A	P	1	2.254	1.039	4.889	2.06	0.0396
trtp*avisitn	A	P	2	3.778	1.713	8.333	3.29	0.0010
trtp*avisitn	A	P	3	3.100	1.405	6.840	2.80	0.0051
trtp*avisitn	A	P	4	2.011	0.944	4.288	1.81	0.0704

R

```
> OR
  avisit   OR lower.CL upper.CL    Z p-value
1      1 2.25    1.039     4.89 2.06  0.0396
2      2 3.78    1.713     8.33 3.29  0.0010
3      3 3.10    1.405     6.84 2.80  0.0051
4      4 2.01    0.944     4.29 1.81  0.0704
```

GENERALIZED LINEAR MIXED MODEL (GLMM) - GHQ APROXIMATION

Likelihood approximation using GHQ is based on an integral split in a given number of points.

GLMM with GHQ approximation can be fitted using the GLIMMIX procedure in SAS, or the `lme4::glmer` and `GLMMadaptative::mixed_model` R functions. The syntax is similar to GEE models, with the difference that GLMM incorporates intercept and subject as random effects (i.e.: random baseline and slope for each subject).

In GLMMs, intra-subject correlation is estimated through the random effects. In SAS the random effects are specified with the `RANDOM` statement, while the `TYPE= option` statement can be used to specify the covariance structure of G (variance matrix of the random effects). Variance Components by default (`type=VC`) is the default option. In R, the equivalent specification is 1|usubjid, where 1 denotes the random intercept.

Unlike in GEE models discussed in previous section, which computed the robust Sandwich S.E. by default, the GLIMMIX procedure in SAS and the R functions used in this section display the model-based S.E. (also called naïve S.E. in R) by default.

In this example, a GHQ proximation with 5 quadrature points is used. The number of points is specified using `method=quad(qpoints=n)` in SAS, and `nAGQ=n` in R. In SAS, the option solution needs to be added to display the

estimated parameters. Since degrees of freedom (df) by default differ between software (SAS uses between-within, while R uses Infinite), the option `ddfm=none` is included in SAS to align with R. The corresponding syntax is shown in Table 7.

Table 7. GLMM - GHQ approximation (5 points): SAS® and R code.

SAS

```
proc glimmix data=resp method=quad(qpoints=5);
class trtp(ref="P") avisitn(ref='1') usubjid;
model outcome=trtp avisitn trtp*avisitn/dist=bin link=logit solution
ddfm=none/*1*/;
random intercept /subject=usubjid /*type=vc [2]*/;
run;
```

R

```
model <- glmer(outcome ~ trtp + avisitn + trtp*avisitn + (1 | usubjid),
               data = resp,
               family = binomial(link ="logit"),
               nAGQ=5)
```

```
model <- mixed_model(fixed = outcome ~ trtp + avisitn + trtp*avisitn,
             random = ~1|usubjid,
             data = resp,
             family = binomial(link = "logit"),
             nAGQ=5)
```

[1] Option to modify df (Between-within by default), set to none to get same results as with R.
[2] Covariance structure of G (residuals covariance matrix). Variance components (VC) by default.

The results obtained are shown in Table 8. The main general difference between SAS and R lies in how p-values are calculated: SAS uses the t- distribution, while R relies on the standard normal (Z) distribution. In R, the results from the `glmer` function closely align those from SAS, with differences found in the 5th decimal (eg: -0.87191 vs. -0.87199 in parameter estimation for AVISITN=2). However, the results produced by the mixed_model function in R show slight deviations when compared to both GLIMMIX and `glmer`, suggesting differences in the model implementation or estimation

Table 8. GLMM - GHQ approximation (5 points): Estimated parameters. Results obtained with SAS® and R. Model Based S.E. and df=Inf.

SAS

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.06170	.5274	I	-0.12	0.9069
trtp	A	_	1.57500	.7835	I	2.01	0.0444
avisitn		2	-0.87191	.5476	I	-1.59	0.1113
avisitn		3	-0.28727	.5369	I	-0.54	0.5926
avisitn		4	-0.43174	.5386	I	-0.80	0.4228
trtp*avisitn	A	2	1.04384	.8036	I	1.30	0.1939
trtp*avisitn	A	3	0.63608	.8000	I	0.80	0.4265
trtp*avisitn	A	4	-0.22001	.7866	I	-0.28	0.7797

R

glmer

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-0.0617	0.5274	-0.12	0.907
trtpA	1.5750	0.7835	2.01	0.044
avisitn2	-0.8720	0.5476	-1.59	0.111
avisitn3	-0.2873	0.5369	-0.54	0.593
avisitn4	-0.4318	0.5386	-0.80	0.423
trtpA:avisitn2	1.0440	0.8036	1.30	0.194
trtpA:avisitn3	0.6362	0.8000	0.80	0.426
trtpA:avisitn4	-0.2200	0.7866	-0.28	0.780

mixed_model

	Estimate	Std.Err	z-value	p-value
(Intercept)	-0.076	0.581	-0.131	0.90
trtpA	1.639	0.873	1.878	0.06
avisitn2	-0.875	0.547	-1.597	0.11
avisitn3	-0.289	0.537	-0.538	0.59
avisitn4	-0.433	0.539	-0.804	0.42
trtpA:avisitn2	1.046	0.803	1.303	0.19
trtpA:avisitn3	0.637	0.800	0.797	0.43
trtpA:avisitn4	-0.219	0.787	-0.278	0.78

SANDWICH S.E. AND D.F.

However, Li, P., & Redden, D. T. (2015) [4], recommended using the Between-Within denominator degrees of freedom approximation method when using GLMMs in cluster randomized trials with binary outcomes and a small number of heteregoneous clusters.

Additionally, the U.S. Food and Drug Administration (FDA) advises *sponsors to consider using of robust standard error method such as the Huber-White "sandwich" standard error, particularly when the model does not include*

treatment by covariate interactions [5].

Therefore, estimatations may need to incorporte the between-within df and/or the Sandwich robust S.E.. In SAS, df method can be specified using the dffm option in the model statment (with Between-Within being the default, or `ddfm=BW`). The Sandwich S.E. is easily obtained by adding the `empirical` option in the proc statment.

In R, `lme4::glmer` function does not have these options, and suplementary functions such as `merDeriv::sandwich` can be used to obtain Sandwich S.E. For the df, the function `parameters::dof_betwithin` can be used, though it implements a heuristic based on the between-within approach and does not return exactly the same results as shown in Li and Redden 2015 [4], but similar results are obtained [8].

Table 9. displays the R code to obtain the Sandwich S.E. A macro called `new_model` is created (See code in Appendix 1), to re-calculate p-values using the Sandwich S.E. and the specified df. These p-values are based in the t-distribution to align with SAS calculations.

Table 9. GLMM - GHQ approximation (5 points): SAS® and R code

SAS

```
proc glimmix data=resp method=quad(qpoints=5) empirical;
 class trtp(ref="P") avisitn(ref='1') usubjid;
 model outcome=trtp avisitn trtp*avisitn/dist=bin link=logit solution
ddfm=betwithin /*[1]*/;
 random intercept /subject=usubjid /*type=vc*/; /*[2]*/;
run;
```

R

```
 model <- glmer(outcome ~ trtp + avisitn + trtp*avisitn + (1 | usubjid),
                data = resp,
                family = binomial(link ="logit"),
                nAGQ=5)

#Get parameter estimation
est<-fixef(model)

#Get Sandwich covariance matrix
library(merDeriv)
vcov<-sandwich(model)

#Get S.E. (the diagonal from the covariance matrix), remove last value as it
corresponds to random effects
se_sw0<- sqrt(diag(sandwich(model)))
se_sw <-head(se_sw0, -1)

#Re-calculate p-values using different df
new_model(model, est_fix=est, se=se_sw , df=Inf )
new_model(model, est_fix=est, se=se_sw , df=dof_betwithin(model))
```

[1] Option to set up df (Between-within by default), set to none to get same results as with R.
[2] Covariance structure of G (residuals covariance matrix). Variance components (VC) by default.

The results (displayed in Table 10) are largely consistent when using infinite df, with only a minor difference in the estimated S.E. for the intercept (-0.8719 SAS vs. -0.8720 in R), attributable to differences in rounding (R rounds to the even digit (i.e. both 1.5 and 2.5 round to 2), while SAS uses “conventional” rounding rules (i.e 1.5 is rounded to 2 and 2.5 to 3 [6]) However, the estimated df differ between the two software (estimation in R is an aproximate version of the between-within method). This discrepancy in the df leads to small variations in the p-value computations across software.

Table 10. GLMM - GHQ approximation (5 points): Estimated parameters. Results obtained with SAS® and R. Sandwich S.E

DF: Inf

SAS

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.0617	.5345	I	-0.1154	0.9081
trtp	A	_	1.5750	.7868	I	2.0019	0.0453
avisitn		2	-0.8719	.5391	I	-1.6173	0.1058
avisitn		3	-0.2873	.6102	I	-0.4708	0.6378
avisitn		4	-0.4317	.5164	I	-0.8361	0.4031
trtp*avisitn	A	2	1.0438	.8219	I	1.2700	0.2041
trtp*avisitn	A	3	0.6361	.8588	I	0.7406	0.4589
trtp*avisitn	A	4	-0.2200	.7939	I	-0.2771	0.7817

R

```
             Estimate Std. Error  df tvalue P-value
(Intercept)   -0.0617      0.538 Inf -0.115  0.9087
trtpA          1.5750      0.791 Inf  1.990  0.0466
avisitn2      -0.8720      0.539 Inf -1.617  0.1058
avisitn3      -0.2873      0.610 Inf -0.471  0.6378
avisitn4      -0.4318      0.516 Inf -0.836  0.4031
trtpA:avisitn2  1.0440      0.822 Inf  1.270  0.2041
trtpA:avisitn3  0.6362      0.859 Inf  0.741  0.4589
trtpA:avisitn4 -0.2200      0.794 Inf -0.277  0.7817
```

DF: Between-Within

SAS

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.0617	.5345	109	-0.1154	0.9083
trtp	A	_	1.5750	.7868	109	2.0019	0.0478
avisitn		2	-0.8719	.5391	327	-1.6173	0.1068
avisitn		3	-0.2873	.6102	327	-0.4708	0.6381
avisitn		4	-0.4317	.5164	327	-0.8361	0.4037
trtp*avisitn	A	2	1.0438	.8219	327	1.2700	0.2050
trtp*avisitn	A	3	0.6361	.8588	327	0.7406	0.4594
trtp*avisitn	A	4	-0.2200	.7939	327	-0.2771	0.7819

R

```
             Estimate Std. Error  df tvalue P-value
(Intercept)   -0.0617      0.538 324 -0.115  0.9088
trtpA          1.5750      0.791 324  1.990  0.0474
avisitn2      -0.8720      0.539 324 -1.617  0.1068
avisitn3      -0.2873      0.610 324 -0.471  0.6381
avisitn4      -0.4318      0.516 324 -0.836  0.4037
trtpA:avisitn2  1.0440      0.822 324  1.270  0.2050
trtpA:avisitn3  0.6362      0.859 324  0.741  0.4594
trtpA:avisitn4 -0.2200      0.794 324 -0.277  0.7819
```

To obtain probabilities and OR, the same code as described in GEE models can be used. To compute results using the Sandwich S.E., the sandwich function can be used in the corresponding argument in emmeans:

```
emmeans(model, ~ trtp*avisitn, data=resp, vcov.method=sandwich(model),…)
```

GENERALIZED LINEAR MIXED MODEL (GLMM) - LAPLACE APROXIMATION

Laplace is a particular GHQ where only one point is used. In SAS, it can be obtained in the method statement using either `method=quad(qpoints=1)` or `method=laplace`. Both approaches return similar results with slight differences in later decimal places (See Appendix 2 for comparison between approaches).

Table 11 presents the code used to fit the model using Laplace approximation. In R, two functions are highlighted: `lme4::glmer` and `glmmTMB::glmmTMB`. An initial attempt using `glmer` resulted in a convergence warning:

```
In checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv,  :
Model failed to converge with max|grad| = 0.00442931 (tol = 0.002, component 1).
```

This warning can be addressed by switching the optimizer to "bobyqa" and extending the maximum number of iterations. These adjustments not only suppress the warning but provides closer results to SAS [6].

The `glmmTMB::glmmTMB` function is also a viable alternative. Recent versions of the package R (>= 3.6.0) support the computation of Sandwich S.E. using the using `clubSandwich::vcovHC` function [7].

Table 11. GLMM - Laplace: SAS® and R code. Sandwich S.E.

SAS

```
proc glimmix data=resp method=laplace empirical;
  class trtp(ref="A") avisitn(ref='1') usubjid;
  model outcome=trtp avisitn trtp*avisitn/dist=bin link=logit solution ddfm=none;
  random intercept /subject=usubjid /*type=vc*/;
run;
```

R

```r
library(lme4)
model <- glmer(formula =outcome ~ trtp + avisitn + trtp*avisitn + (1 | usubjid),
               data = resp,
                family = binomial(link ="logit"),
                nAGQ=1,
                control=glmerControl(optimizer="bobyqa",
                        optCtrl=list(maxfun=100000)))

#Get estimator
est<-fixef(model)
library(merDeriv)
#Get Sandwich covariance matrix
vcov<-sandwich(model)

#Get S.E. (the diagonal from the covariance matrix), remove last value as it
corresponds to random effects
se_sw0<- sqrt(diag(sandwich(model)))
se_sw <-head(se_sw0, -1)

#Re-calculate p-values using different df
new_model(model, est_fix=est1, se=se_sw , df=Inf )
new_model(model, est_fix=est1, se=se_sw , df=dof_betwithin(model))
```

```r
model<- glmmTMB(outcome ~ trtp + avisitn + trtp*avisitn + (1 | usubjid),
                data = resp,
                family = binomial(link = "logit"))

#Get estimator
est<-fixef(model)$cond

#Get Sandwich S.E.
library(clubSandwich)
se_sw<- sqrt(diag(vcovHC(model)))

#Re-calculate p-values using different df
new_model(model, est_fix=est, se=se_sw , df=Inf )
new_model(model, est_fix=est, se=se_sw , df=dof_betwithin(model))
```

Results are shown in Table 12. While the `glmer` function provides results reasonably close to SAS, the `glmmTMB` yields results that align more closely with SAS.

Table 12. GLMM - Laplace: Estimated parameters. Results obtained with SAS® and R. Sandwich S.E.

SAS

Df: Inf.

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.0847	.5452	I	-0.1554	0.8765
trtp	A	_	1.6517	.8182	I	2.0188	0.0435
avisitn		2	-0.8576	.5302	I	-1.6175	0.1058
avisitn		3	-0.2819	.5986	I	-0.4709	0.6377
avisitn		4	-0.4236	.5069	I	-0.8357	0.4033
trtp*avisitn	A	2	1.0248	.8026	I	1.2768	0.2017
trtp*avisitn	A	3	0.6222	.8397	I	0.7410	0.4587
trtp*avisitn	A	4	-0.2050	.7727	I	-0.2653	0.7908

Df: Between-Within

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.0847	.5452	109	-0.1554	0.8768
trtp	A	_	1.6517	.8182	327	2.0188	0.0443
avisitn		2	-0.8576	.5302	327	-1.6175	0.1067
avisitn		3	-0.2819	.5986	327	-0.4709	0.6380
avisitn		4	-0.4236	.5069	327	-0.8357	0.4039
trtp*avisitn	A	2	1.0248	.8026	327	1.2768	0.2026
trtp*avisitn	A	3	0.6222	.8397	327	0.7410	0.4592
trtp*avisitn	A	4	-0.2050	.7727	327	-0.2653	0.7910

R: glmer

	Estimate	Std. Error	df	tvalue	P-value
(Intercept)	-0.0848	0.479	Inf	-0.177	0.8596
trtpA	1.6518	0.697	Inf	2.369	0.0179
avisitn2	-0.8576	0.531	Inf	-1.615	0.1062
avisitn3	-0.2819	0.599	Inf	-0.471	0.6377
avisitn4	-0.4236	0.507	Inf	-0.836	0.4030
trtpA:avisitn2	1.0247	0.804	Inf	1.275	0.2022
trtpA:avisitn3	0.6222	0.840	Inf	0.741	0.4589
trtpA:avisitn4	-0.2050	0.771	Inf	-0.266	0.7902

	Estimate	Std. Error	df	tvalue	P-value
(Intercept)	-0.0848	0.479	324	-0.177	0.8597
trtpA	1.6518	0.697	324	2.369	0.0184
avisitn2	-0.8576	0.531	324	-1.615	0.1072
avisitn3	-0.2819	0.599	324	-0.471	0.6380
avisitn4	-0.4236	0.507	324	-0.836	0.4037
trtpA:avisitn2	1.0247	0.804	324	1.275	0.2031
trtpA:avisitn3	0.6222	0.840	324	0.741	0.4594
trtpA:avisitn4	-0.2050	0.771	324	-0.266	0.7903

glmmTMB

	Estimate	Std. Error	df	tvalue	P-value
(Intercept)	-0.0847	0.545	Inf	-0.155	0.8765
trtpA	1.6514	0.818	Inf	2.018	0.0436
avisitn2	-0.8575	0.530	Inf	-1.617	0.1058
avisitn3	-0.2819	0.599	Inf	-0.471	0.6378
avisitn4	-0.4236	0.507	Inf	-0.836	0.4033
trtpA:avisitn2	1.0247	0.803	Inf	1.277	0.2017
trtpA:avisitn3	0.6221	0.840	Inf	0.741	0.4588
trtpA:avisitn4	-0.2050	0.773	Inf	-0.265	0.7908

	Estimate	Std. Error	df	tvalue	P-value
(Intercept)	-0.0847	0.545	324	-0.155	0.8766
trtpA	1.6514	0.818	324	2.018	0.0444
avisitn2	-0.8575	0.530	324	-1.617	0.1068
avisitn3	-0.2819	0.599	324	-0.471	0.6381
avisitn4	-0.4236	0.507	324	-0.836	0.4040
trtpA:avisitn2	1.0247	0.803	324	1.277	0.2026
trtpA:avisitn3	0.6221	0.840	324	0.741	0.4593
trtpA:avisitn4	-0.2050	0.773	324	-0.265	0.7910

GENERALIZED LINEAR MIXED MODEL (GLMM) - PQL APROXIMATION

The PQL approach uses linear approximations instead of likelihood, making it less accurate for binary outcomes compared to the GHQ or Laplace methods described above. In SAS, this is implemented by default using the Residual Pseudo-Likelihood method (`method=RSPL`), which is a refinement of PQL which incorporates residual adjustments to better approximate the marginal likelihood, in the GLIMMIX procedure. In R the PQL computation can be obtained using the `glmmPQL` function form the `MASS` package.

Table 13 shows the syntax for both software, where df in PROC GLIMMIX are set to residual, for consistency and Table 14 the corresponding results.

Table 13. GLMM - PQL: SAS® and R code

SAS

```
proc glimmix data=resp method=rspl;
 class trtp(ref="A") avisitn(ref='1') usubjid;
 model outcome=trtp avisitn trtp*avisitn/ dist=bin link=logit solution
ddfm=residual;
 random intercept /subject=usubjid;
run;
```

R

```
library(MASS)
model <- glmmPQL(outcome ~ trtp + avisitn + trtp*avisitn,
                 random=list(~1|usubjid),
                 data = resp,
                 family = binomial(link = "logit"))
```

Table 14. GLMM - PQL: Estimated parameters. Results obtained with SAS® and R.

SAS

Effect	trtp	avisitn	Estimate	StdErr	DF	tValue	Probt
Intercept		_	-0.01934	0.3915	436	-0.05	0.9606
trtp	A	_	1.0241	0.5756	436	1.78	0.0759
avisitn		2	-0.6228	0.4583	436	-1.36	0.1749
avisitn		3	-0.2047	0.4527	436	-0.45	0.6514
avisitn		4	-0.3079	0.4537	436	-0.68	0.4977
trtp*avisitn	A	2	0.7466	0.6767	436	1.10	0.2705
trtp*avisitn	A	3	0.4563	0.6762	436	0.67	0.5001
trtp*avisitn	A	4	-0.1560	0.6630	436	-0.24	0.8141

R

	Value	Std.Error	DF	t-value	p-value
(Intercept)	0.000	0.413	327	0.000	1.0000
trtpA	1.220	0.603	109	2.024	0.0454
avisitn2	-0.815	0.394	327	-2.067	0.0395
avisitn3	-0.268	0.388	327	-0.691	0.4898
avisitn4	-0.403	0.389	327	-1.037	0.3006
trtpA:avisitn2	0.976	0.580	327	1.684	0.0931
trtpA:avisitn3	0.595	0.578	327	1.029	0.3041
trtpA:avisitn4	-0.200	0.568	327	-0.352	0.7251

Results differ because of the different computation methods (PQL vs. RSPL) across software. Since `glmmPQL` is widely recognized as less reliable for binary outcomes, more robust approaches such as Laplace or GHQ—discussed in previous sections—are generally preferred. Consequently, further investigation using `glmmPQL` is not pursued.

CATEGORICAL OUTCOME WITH MORE THAN TWO CATEGORIES

Although less common than binary outcomes, endpoints with more than two categories may be the outcome of interest, which can be either ordinal or nominal.

In SAS, similar syntax used for GEE and GLMM models can be applied by specifying a multinomial distribution and selecting the appropriate link function. Models with cumulative link functions apply to ordinal data and generalized logit models are fit to nominal data [3]).

In R, the functions described earlier (e.g.: `glmer, glmmTMB, geeglm`, etc.) do not support multinomial outcomes, as these functions relay on the family function, which does not include multinomial option. Nevertheless, the `multgee package` [9] provides two functions for estimating GEE models with categorical outcomes.

Table 15 displays the syntax used to fit GEE models. In R, two different functions are available in the `multgee` package [9], depending on the type of outcome: `ordLORgee` for ordinal variables and `nomLORgee` for nominal variables.

By default, the correlation matrix structure in SAS is independence. In contrast, the default in R is exchangeable, specified as "`category.exch`" for `ordLORgee` and "`time.exch`" for `nomLORgee`. To ensure consistency with the SAS default, the R code shown sets the correlation structure to be independent.

Table 15. GEE – Categorical outcome with more than two categories: SAS® and R code

SAS – Ordinal

```
proc gee data=resp ;
class trtp(ref="A") avisitn(ref='1') usubjid;
model respord=trtp avisitn trtp*avisitn/ dist=multinomial link=cumlogit;
repeated subject=usubjid  /corr=ind;
run;
```

SAS – Nominal

```
proc gee data=resp ;
class trtp(ref="A") avisitn(ref='1') usubjid;
model respnom(event='Liver')=trtp avisitn trtp*avisitn/ dist=multinomial
link=glogit;
repeated subject=usubjid  /corr=ind;
run;
```

R – Ordinal

```
model <- ordLORgee(formula = respord ~ trtp + avisitn  + trtp*avisitn,
                   data = resp,
                   id = usubjid,
                   repeated = avisitn,
                   LORstr = "independence")
```

R – Nominal

```
model <- nomLORgee(formula = respnom ~ trtp + avisitn + trtp*avisitn,
                   data = resp,
                   id = usubjid,
                   repeated = avisitn,
                   LORstr = "independence")
```

For GLMM, in SAS, similar modifications apply to the GLIMMIX procedure. One notable limitation is that the LSMEANS statement does not work as expected in GLIMMIX with the multinomial distribution. However, ORs can still be obtained by using the `oddsratio` option in the model statement. No R functions equivalent to SAS's GLIMMIX procedure have been identified for handling multinomial distributions in a frequentist framework.

CONCLUSION

SPECIFIC CONCLUSIONS

GEE and GLMM are effective for modelling categorical outcomes using appropriate link functions: *logit* for binary, *cumlogit* for ordinal, and *glogit* for nominal outcomes. A key distinction is that GLMMs account for intra-subject correlation through random effects, while GEEs use a working correlation matrix.

For GEE models, `PROC GEE/GENMOD` in SAS and `gee/geepack` in R yield comparable results, based on robust (sandwich) standard errors.

For GLMMs, Laplace and Gauss-Hermite Quadrature (GHQ) are preferred over Penalized Quasi-Likelihood (PQL). In SAS, PROC GLIMMIX allows specification of the approximation method with x points (e.g: `method=quad(qpoints=x)` or `method=laplace`). In R, `glmer` supports GHQ via `nAGQ=x`, while `glmmTMB` is recommended for Laplace approximation.

- Df handling differs: SAS defaults to between-within (`ddfm=bw`), while R assumes infinite df. To align results, `ddfm=none` can be specified in SAS or the function `dof_between` can be used in R, though it is an approximation and does not exactly match with SAS computation.
- Model-based standard errors are default in GLMMs, but robust SEs can be obtained using the empirical option in SAS, or packages like `merDeriv` and `clubSandwich` in R.
- Predicted probabilities and odds ratios (ORs) can be extracted via `lsmeans` and/or `oddsratio` in SAS, and using the `emmeans` package in R.
- Convergence criteria and optimization methods may be considered as well when discrepancies across methods are found.

GENERAL CONCLUSIONS

Differences between SAS and R, (and even between different functions/procedures within same software) can lead to different results. These differences do not imply that one tool is more reliable than another but rather reflect variations in default settings and computational methods. Carefully reviewing and aligning these defaults is essential for consistency and reproducibility.

To support this effort the CAMIS repository [1] documents known differences between statistical implementations in different software (such SAS, R and/or Phyton). It is a valuable resource for understanding and resolving discrepancies, promoting transparency and reproducibility across platforms.

REFERENCES

[1] Comparing Analysis Method Implementations in Software (CAMIS)
[2] SAS Institute Inc. SAS Help Center. The GEE procedure.
[3] Li, P., & Redden, D. T. (2015). Comparing denominator degrees of freedom approximations for the generalized linear mixed model in analyzing binary outcome in small sample cluster-randomized trials. BMC Medical Research Methodology, 15, 38.
[4] U.S. Food and Drug Administration. (2023). Adjusting for Covariates in Randomized Clinical Trials for Drugs and Biological Products: Guidance for Industry. Center for Drug Evaluation and Research (CDER), Center for Biologics Evaluation and Research (CBER).
[3] SAS/STAT® 13.1 User's Guide The GENMOD Procedure.
[5] SAS/STAT® 13.1 User's Guide The GLIMMIX Procedure
[6] Stack Overflow [Internet]. 2008 [Last visited: 2025 Sep 30]
[7] Brooks, M. E., et al. (2025). glmmTMB: Generalized Linear Mixed Models using Template Model Builder (Version 1.1.12) [R package manual]. The Comprehensive R Archive Network (CRAN). https://cran.r-project.org/web/packages/glmmTMB/glmmTMB.pdf
[8] Documentation of package parameters. dof_betwithin
[9]Touloumis A. (2015). "R Package multgee: A Generalized Estimating Equations Solver for Multinomial Responses." Journal of Statistical Software.

ACKNOWLEDGMENTS

I would like to thank Isabelle Smith for her careful and thorough review of this paper, and her valuable corrections and suggestions.

CONTACT INFORMATION

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

Author Name: Miriam Amor
Company: Veramed GmBH
Address: Registered Office: 5th Floor Regal House, 70 London Road, Twickenham TW1 3QS,
Work Phone: NA
Email: Miriam.amor@veramed.co.uk
Website: https://www.veramed.co.uk

APPENDIX 1: MACRO CODE TO CALCULATE P-VALUES

Macro parameters:
MODEL: object resulting from a function (e.g.: geeglm, glmer, glmmTMB, etc.)
EST_FIX: vector with estimated parameters for the fixed effects
SE: vector with the estimated S.E. for the fixed effects.
DF: Degrees of freedom (e.g.: Inf. or dof_between(model), to use infinite or between-within, respectively.

Macro for objects from lme4::glmer:

```
new_model <- function(model, est_fix , se , df){

  #Re-calculate (2-sided) p-values using estimated parameters and its SE
  t <- est_fix /se
  pvalue <-2*pt(q=abs(t), df=df, lower.tail=FALSE)

  #Combine results in a data frame and add row names
  new_model <- round(cbind(est_fix, se, df, t, pvalue), digits=4)
  colnames(new_model) <- c("Estimate", "Std. Error", "df", "tvalue", "P-value")
  rownames(new_model) <- rownames(summary(model)$coefficients)
  new_model
}
```

Macro for objects from glmmTMB::glmmTMB:

```
new_model <- function(model, est_fix , se , df){

  #Re-calculate (2-sided) p-values using estimated parameters and its SE
  t <- est_fix /se
  pvalue <-2*pt(q=abs(t), df=df, lower.tail=FALSE)

  #Combine results in a data frame and add row names
  new_model <- round(cbind(est_fix, se, df, t, pvalue), digits=4)
  colnames(new_model) <- c("Estimate", "Std. Error", "df", "tvalue", "P-value")
  rownames(new_model) <- rownames(summary(model)$coefficients$cond)
  new_model
}
```

APPENDIX 2: COMPARISON BETWEEN SAS PROCEDURES RESULTS

GEE: PROC GEE VS PROC GENMOD

PROC GEE and PROC GENMOD share identical syntax. Results are compared, and, while exact match is not possible due to differences in least significant decimal places, a clean comparison accurate up to the 9th decimal place can be obtained using criterion in the PROC COMPARE.

```
proc gee data=resp ;
  class trtp(ref="P") avisitn(ref='1') usubjid ;
  model outcome(event='1')= trtp avisitn trtp*avisitn/ dist=bin link=logit;
  repeated subject=usubjid  /corr=ind
  ods output   GEEEmpPEst=GEEEmpPEst_gee;
run;

proc genmod data=resp ;
  class trtp(ref="P") avisitn(ref='1') usubjid ;
  model outcome(event='1')= trtp avisitn trtp*avisitn/ dist=bin link=logit;
  repeated subject=usubjid  /corr=ind;
run;

proc compare base=GEEEmpPEst_gee compare=GEEEmpPEst_genmod criterion=0.00000000000001;
run
```

```
                    The COMPARE Procedure
    Comparison of WORK.GEEEMPPEST_GEE with WORK.GEEEMPPEST_GENMOD
               (Method=RELATIVE(1), Criterion=1.0E-14)

                         Data Set Summary

Dataset                          Created          Modified  NVar    NObs

WORK.GEEEMPPEST_GEE     01SEP25:08:31:07  01SEP25:08:31:07     9      15
WORK.GEEEMPPEST_GENMOD  01SEP25:08:31:07  01SEP25:08:31:07     9      15


                         Variables Summary

                Number of Variables in Common: 9.


                       Observation Summary

                   Observation      Base  Compare

                   First Obs           1        1
                   Last  Obs          15       15

  Number of Observations in Common: 15.
  Total Number of Observations Read from WORK.GEEEMPPEST_GEE: 15.
  Total Number of Observations Read from WORK.GEEEMPPEST_GENMOD: 15.

  Number of Observations with Some Compared Variables Unequal: 0.
  Number of Observations with All Compared Variables Equal: 15.


                     Values Comparison Summary

   Number of Variables Compared with All Observations Equal: 9.
   Number of Variables Compared with Some Observations Unequal: 0.
   Total Number of Values which Compare Unequal: 0.
   Total Number of Values not EXACTLY Equal: 47.
   Maximum Difference Criterion Value: 2.1539E-15.
```

GLMM – LAPLACE: GLIMMIX with method=Laplace vs.method=quad(qpoints=1)

As Laplace is a special case of GHQ approximation using only 1 point, it can be specified in SAS using either method=laplace or method=quad(qpoints=1). Although both options return similar results, discrepancies may occur beyond the 8th decimal place may be found. Results would match using the criterion option in the proc compare.

```
                                The COMPARE Procedure
                  Comparison of WORK.PARAMETERLAPLACE with WORK.PARAMETERGHQ_1P
                        (Method=RELATIVE(0.00000222), Criterion=1.0E-08)

                                     Data Set Summary

Dataset                         Created          Modified  NVar    NObs  Label

WORK.PARAMETERLAPLACE  01SEP25:08:19:28  01SEP25:08:19:28     8      15  Solutions for Fixed Effects
WORK.PARAMETERGHQ_1P   01SEP25:08:19:28  01SEP25:08:19:28    11      15  Solutions for Fixed Effects


                                     Variables Summary

                               Number of Variables in Common: 8.
                               Number of Variables in WORK.PARAMETERGHQ_1P but not in WORK.PARAMETERLAPLACE: 3.


                                     Observation Summary

                                   Observation      Base  Compare

                                   First Obs           1        1
                                   Last  Obs          15       15

                Number of Observations in Common: 15.
                Total Number of Observations Read from WORK.PARAMETERLAPLACE: 15.
                Total Number of Observations Read from WORK.PARAMETERGHQ_1P: 15.

                Number of Observations with Some Compared Variables Unequal: 0.
                Number of Observations with All Compared Variables Equal: 15.


                                  Values Comparison Summary

                 Number of Variables Compared with All Observations Equal: 8.
                 Number of Variables Compared with Some Observations Unequal: 0.
                 Total Number of Values which Compare Unequal: 0.
                 Total Number of Values not EXACTLY Equal: 32.
                 Maximum Difference Criterion Value: 2.2958E-09.
```

```
proc glimmix data=resp method=laplace empirical;
class trtp(ref="P") avisitn(ref='1') usubjid;
model outcome=trtp avisitn trtp*avisitn/ dist=bin link=logit solution ddfm=none;
random intercept /subject=usubjid;
ods output ParameterEstimates=ParameterLaplace;
run;

proc glimmix data=resp method=quad(qpoints=1) empirical;
class trtp(ref="P") avisitn(ref='1') usubjid;
model outcome=trtp avisitn trtp*avisitn  / dist=bin link=logit solution cl ddfm=none;
random intercept /subject=usubjid;
ods output ParameterEstimates=ParameterGHQ_1p;
run;

proc compare base=ParameterLaplace compare=ParameterGHQ_1p criterion=0.00000001;
run;
```

APPENDIX 3: DERIVATION OF MULTI-LEVEL VARIABLES (ORDINAL AND NOMINAL)

```
proc format;
  value respmulti
  1='Liver'
  2='Lung'
  3='Bone';
run;

data resp;
  set resp1;
   call streaminit(1234);
  respord = rand("integer", 1, 3);     *Ordinal;
   respnom = put(respord, respmulti.);  *Nominal;
run;
```