

Applied Aspects of Using Multiple Imputation in Clinical Data

Christina M. Gullion, Center for Health Research, Portland, OR
Chuhe Chen, Center for Health Research, Portland, OR
Gayle T. Meltesen, Center for Health Research, Portland, OR

ABSTRACT

Use of multiple imputation to replace missing outcomes in clinical research is a relatively new approach. Thus, there is little practical information published on how to use SAS to obtain and combine the diverse kinds of estimates that are needed in a typical paper reporting results of a clinical trial. This presentation summarizes procedures developed by us, or adapted from other sources, necessary to complete all the stages of preparing and analyzing a dataset when multiple imputation is used to replace missing values. The paper covers guidelines for analyzing multiple copies of a dataset, and setting up output of SAS PROCs (e.g., GLM, REG, MIXED) for processing by PROC MIANALYZE, as well as outlining other routines we developed for obtaining combined results not available from PROC MIANALYZE. This paper does not report on the efficiency or unbiasedness of the results. Rather, we focus on providing guidance on the use of SAS® to obtain meaningful results from analysis of a multiply-imputed dataset for consumption by non-statistical researchers and for publication.

INTRODUCTION

Use of multiple imputation (MI) to replace missing outcomes is a relatively new approach in clinical research, although it was developed for use in the US Census almost 30 years ago (Rubin, 1980). To date, there has been little practical information published on the coding challenges of analyzing multiply imputed data to produce the diverse kinds of estimates that are needed in a typical paper reporting results of a clinical trial.

Missing data are almost inevitable in multivariate clinical trials—especially in longitudinal studies. Subjects may decline to respond to certain questions at each time point, or drop out at an early stage of the study. In the past, it was common to analyze only participants with complete data. This often produced biased estimates and incorrect statistical inferences, especially in multivariate modeling. Multiple imputation is also important because it is necessary in order to use intention-to-treat analyses properly, because all randomized participants must be included in these analyses.

Previously, a single imputation of missing data was the most widely used alternative for dealing with missing data. Examples of the procedures commonly used are a) filling in the missing values with the sample mean, b) a predicted value from a regression model, or c) a sampled value from a “Hot Deck” procedure [Todd, 2001]. If the data were *missing completely at random* [users guide], this would yield an unbiased point estimate; however, this was rarely the case. Even if it were the case, the standard errors and confidence intervals would be too small because a single imputation does not account for the increased uncertainty about parameters associated with having missing data. Consequently, the Type I error rate may be inflated. In contrast, under fairly liberal assumptions about the process resulting in missing data, MI can be used to provide a basis for valid inferences from a complex and incomplete clinical study dataset, with standard errors adjusted for the uncertainty deriving from missing data.

This paper summarizes the procedures we use to analyze a dataset when MI is used to replace missing data. We have developed (or adapted from other sources) an entire sequence of procedures, from preparation of the data for imputation through combining results into a form that can be reported to non-statistical researchers and used in a paper or presentation. This paper does not report on the efficiency or unbiasedness of imputation results and does not provide technical information about the mathematics of multiple imputation. We use SAS®, Release 9.1.3, with PROC MI and PROC MIANALYZE as our tools for imputing data and combining results after analysis.

SUMMARY OF STEPS

The paper is organized according to the chronological sequence of activities starting with preparation of a dataset before imputation, through analysis and combination of results. We feel that this summary is important because SAS documentation provides limited examples of how to use PROC MIANALYZE, and there no integrated presentation of the entire sequence of steps exists. Thus, we have tried to accomplish this task in this paper. The major steps are:

- Specify imputation model
- Prepare data for imputation
- Carry out MI
- Post-process variables after imputation
- Analyze multiple versions of imputed data

- Combine analysis results across multiple versions

We had some surprises when we started using MI as a method of handling missing data. We found that variables had to be recoded both before and after MI, which can be quite time consuming. We also found that PROC MI can be tricky to use. It may present problems (such as failure to converge) that require an understanding of the iterative process, as well as an understanding of how characteristics of variables might interfere with successful completion of the MI estimation process. Yet another surprise was that several of the SAS/STAT PROCs (REG, GLM, LOGISTIC) do not provide output, in general, that MIANALYZE will accept without re-coding. Finally, PROC MIANALYZE provides only a test of whether the estimate is different from zero. If other tests are needed, these need to be constructed explicitly. Statistical expertise is needed throughout the process.

Specify imputation model

We start by assembling a list of candidate variables for the imputation model. We call this the *imputation model*. In a large clinical trial with a Data Safety and Monitoring Board and staff from funding agencies involved, this list may be requested and scrutinized by both groups as early as several years before the analysis will be started. This process may require educating members of the DSMB and agency staff about best practices for handling missing data.

A vital requirement for a valid MI is that all subgroups of variables that might be used in analyses of the study dataset (*analysis models*) should be nested within the imputation model (Collins et al 2001). The objective is to assure that analyses carried out on subsets of the imputation variable list result in unbiased and efficient estimates.

An important assumption of MI is that the missing data are *missing at random* (MAR), a less stringent assumption than *missing completely at random*. The MAR assumption implies that observed data can predict the missing values. Hence, we include final and interim measures of outcomes and covariates, as well as design variables, predictors of dropout, and variables that code a reason data are missing (Schafer & Graham 2002). In addition, any possibly meaningful interactions should be in the model. Alternatively, the imputation can be carried out separately on subgroups expected to have distinct means or covariance matrix.

Prepare data for imputation

In addition to the usual effort of setting up an analysis dataset, such as auditing and cleaning data values, additional planning and preliminary steps are needed before carrying out a MI. The aim of the following series of steps is to assure that the data meet the assumptions required to obtain unbiased and efficient imputed values:

1. Evaluate variables in the dataset for missing pattern, violations of normality assumption, collinearity.
2. As needed, eliminate redundant variables and indicators with rare events.
3. Recode categorical variables to k-1 indicators, where k is the number of categories.
4. Assign interactions as product of variables involved.
5. Repeat step 1 on the transformed variables, and verify no new issues have appeared.
6. If adjustments are needed, return to step 2.

We use PROC MI to check the missing data pattern. We print the **MISSPATTERN** table into an .rtf file, then copy and paste this into a Microsoft Excel spreadsheet to analyze the missing pattern. We look for three things: a) a monotone missing pattern (which affects choice of

imputation procedure), b) any unexpected missing values possibly due to data processing error, and c) variables with excessive missing?, e.g., a question applicable only to pregnant women.

Example 2. Code used to obtain missing-data pattern

```
proc mi data=pattern_check
  nimpute=0 /*0 to get the missing pattern only*/;
  ods output misspattern=mspattern;
  var var1-var50;
run;
```

Variables that are highly correlated may result in a singular covariance matrix, while not adding much information, so one variable could be omitted. Binary variables with very unbalanced frequencies (e.g., <2% in a category) and no missing values should be excluded from the imputation data, though they should be retained in the analysis dataset. Features such as these appear to increase the likelihood of failure to converge or of a singular covariance matrix.

PROC MI currently assumes that data are distributed as normal, and we found that failure to transform variables to (at least) approximate normality could result in imputed values that are not plausible, such as large negative values in a physical measurement.

Highly skewed quantitative variables should be transformed using a mathematical function, if possible. Generally, a skew statistic greater than $\text{abs}(1.0)$ is used as the criterion for attempting a normalizing transformation. Since exponential transformations can result in extremely small or large numbers, we routinely linearly rescale the results to the original range. If a variable cannot be transformed to normality using a mathematical function, we group values into an ordinal scale with approximately symmetric frequencies. It may also be necessary to combine levels on data collected as an ordinal scale (e.g., income categories), to increase the density and approximate symmetry.

Categorical variables must be recoded as $k-1$ indicators, where k is the number of categories. In addition, once variables are recoded and transformed, the interactions to be included in the model are calculated by multiplying the variables involved. For instance, an interaction between a 2-value and a 3-value categorical variable will add $(2-1)*(3-1) = 2$ variables to the imputation model.

We use the following naming conventions to keep track of variables that have been changed from the original during the imputation process:

- `_tr` as a suffix attached to a variable to indicate it has been transformed.
- `_btr` as a suffix attached to a variable with imputed values to indicate it has been back-transformed
- `_i` as a suffix attached to a variable with imputed values that has not been transformed.

Labels also show the status of the new variables (back-transformed or just imputed).

Example 1. Variable transformation code

```
data transform;
  set all;
  **** mathematical transformations ****;
  var1_tr=var1**(1/4);
  var3_tr=0.1*(var3/10)**3;

  **** transform quantitative measure to ordinal ****;
  var5_tr=.;
  if var5 = 9 then var5_tr=1;
  if 9 < var5 < 12 then var5_tr=2;
```

```

if 11 < var5 < 16 then var5_tr=3;
if 15 < var5 < 21 then var5_tr=4;
if 20 < var5 < 27 then var5_tr=5;
if 26 < var5 < 44 then var5_tr=6;

```

Next, we examine intercorrelations of complete variables to determine whether any are redundant (i.e., correlation > .80) and hence likely to cause numerical problems during estimation of posterior parameters. MI requires an invertible covariance matrix. We sometimes compute the rank of the final variance-covariance matrix to assure that it is full rank (using PROC FACTOR). If the matrix is not full rank, we make further adjustments to the variable list to eliminate the redundancy.

Carry out multiple imputation procedure

We carry out MI using SAS PROC MI, which implements algorithms given by Schafer, 1997. After an initial ("prior") estimate of mean and covariance matrix is obtained, the imputed values of missing data points are updated, and parameters are re-estimated (Schafer, 1997, pp 68-87). Once the iterative estimation of parameters converges, each missing value is sampled from the final ("posterior") joint distribution and then further perturbed with random error. Imputed values are drawn several times (we start with NIMPUTE=5), thus creating several "complete" versions of the dataset.

PROC MI carries out 2 iterative processes, and either can encounter problems. These problems include failure to converge, a singular variance-covariance matrix, or generation of values that are far outliers. We avoid putting minimum/maximum or rounding constraints on the imputation process, and instead use post-processing (see below) to handle these, since these constraints may result in failure to complete the imputation.

Also, if problems occur, further adjustments to problematic variables may be necessary, such as dropping low-frequency binary variables, recoding to eliminate low-frequency or empty cross-variable cells, or improving the transformation to normality. Starting values can be specified for the estimation of parameters, but we have not tried this.

Example 3. Multiple imputation code

```

*** fixed seed to create replicable results, ***
***printed output to monitor process          ***;
proc mi data=for_imputation nimpute=5 seed=56986 out=imputed;
  mcmc initial=em(itprint) nbiter=1000 niter=500 timeplot(wlf)
      acfplot(wlf);
  var var1-var50 site_1 site_2 site_4;
run;

```

Another consideration is the plausibility of imputed values. For example, large negative values for a variable that cannot be negative (e.g., body mass index) may indicate that the dataset does not sufficiently constrain the iterative estimation procedure. For instance, in a dataset with fewer than 5% missing primary outcomes we included several binary indicators of reasons for missed final outcome, each of which had no more than 5 positive instances out of more than 1000 cases. Some imputed values of participant weight were negative and large. When the indicator variables were dropped, the imputed values were in an acceptable, positive range.

We examine the efficiency indicators in the output from PROC MI to determine whether enough imputations have been created. A key indicator of the efficiency of the imputation is the "df" for posterior parameter estimates.

Example 4. Excerpts from PROC MI output showing the DF and Relative Efficiency

Multiple Imputation Parameter Estimates	Multiple Imputation Variance Information
--	---

Variable	...	DF	...	Relative Efficiency
risk	...	86.86	...	0.975137
benefit	...	45.30	...	0.959009
risk_ben_r_tr	...	82.92	...	0.974056
...
Q101_tr	...	109.21	...	0.980423
race_i1	...	204.19	...	0.999536
race_i2	...	204.55	...	0.999726
race_i3	...	204.64	...	0.999777

The DF is a direct function of the efficiency of the estimate. In our experience, variables with smaller DF appear to be those that have other problematic features, e.g., unsatisfactory internal consistency, many missing values, low incidence, or extreme skew. Allison (2001) suggests using the rule-of-thumb that if the DF for a variable is less than 100, there is excessive variability among the imputations for that variable, and more imputations are needed to improve the efficiency of the imputation.

Post-process variables after imputation

Once the number of imputations is satisfactory, we carry out back-transformation of variables that have imputed values and had been transformed to normality solely to meet the needs of the imputation process. Variables transformed using a mathematical function (square root, log, exponential) are back-transformed with the inverse of that function. Imputed values that are outside the range of the variable are reset to the limit, and imputed values are rounded to the unit of the measure.

Categorical variables need special handling, because the imputation generates a value for all of the indicators. The code shown in Example 5 for back-transforming the race indicators takes account of the fact that three values have been imputed for persons whose race is missing. The highest value is identified and the person with missing race is assigned to that category.

Example 5. Code used to back transform variables after imputation.

```
* BACK TRANSFORMATIONS OF IMPUTED VARIABLES;
data dsn;set out_mi2;
* set imputed negative values to 0;
array trans ... waisscore_tr ..... Q101_tr race_i1 race_i2 race_i3 ...;
do i = 1 to dim(trans);
  if trans[i] < 0 then trans[i] = 0;
end;
* inverse transform of imputed continuous variable;
q101_btr = round(((exp (q101_tr)-1)/11),.01);

* inverse transform of categorical variable;
array races race_i1 race_i2 race_i3;
do i = 1 to 3; *recode race category indicators;
  if races[i] > 1 then races[i] = 1;
end;
other = 1 - race_i1 - race_i2 - race_i3;
if other > race_i1 and other > race_i2 and other > race_i3
  then do;race_i1_btr=0;race_i2_btr=0;race_i3_btr=0;
end;
else if race_i1 > race_i2 and race_i1 > race_i3
  then do; race_i1_btr=1;race_i2_btr=0;race_i3_btr=0;
end;
else if race_i2 > race_i3
  then do; race_i2_btr=1;race_i3_btr=0;race_i1_btr=0;
```

```
end;  
else do; race_i1_btr=0;race_i2_btr=0;race_i3_btr=1;  
end;  
drop other;  
Race_cat_i = sum(1*(race_i1_btr=1),2*(race_i2_btr=1),3*(race_i3_btr=1));  
if race_cat_i = 0 then race_cat_i = 4;
```

Continuous variables transformed to binary or ordinal categories must stay that way after imputation; these typically were extremely skewed before recoding. After back transformation, we truncate out-of-range values and round to the number of decimal points in the original measure.

Analyze multiple versions of imputed data

Every analysis is repeated in exactly the same way in each of the imputed datasets, usually by using `BY _IMPUTATION_` as a statement in any SAS PROC. However, it is not obvious how to do this when an analysis plan calls for model selection.

Model selection requires deciding which variables in a set of interest should be included in a predictive model, using statistical criteria to compare a model containing the variable[s] of interest to another model nested within it. We have tried two approaches with imputed data:

1. Automated stepwise modeling (e.g., PROC GLMSELECT or PROC REG) on an arbitrarily selected version of the dataset (e.g., imputation 3). The final model is then obtained on all versions of the dataset and the output is used to obtain combined results. This is a fairly straightforward method, but leaves a small element of doubt that a different version might have yielded a different final model.
2. Stepwise “by hand” on all versions, with results combined over versions at each step and formal comparison by F test with the previous step, with the result used to determine the variables included in the next step. This is extremely time consuming, but avoids any doubts about potential variability between dataset versions.

Combine analysis results across multiple versions

Finally, in order to produce results that can be used in papers or presentations by study investigators, we combine data over imputations. Point estimates can be simply averaged over imputations, whereas statistical tests or estimates of confidence limits require combining the results, usually according to Rubin’s (1987) rules. These rules adjust the standard error and the degrees of freedom for variance between imputations. Setting up output for this final step often requires extra steps, typically using ODS or other output statements to create an output dataset that matches PROC MIANALYZE input specifications.

Appendix 1 contains a summary of how output from various SAS PROCs is handled. We prefer to use the newer PROCs, such as MIXED and GLIMMIX, for modeling imputed data because their output is more likely to match PROC MIANALYZE input requirements. One unanticipated situation is that variable names differ between the parameter dataset and the covariance dataset output from PROC GLM, but PROC MIANALYZE requires that they be consistent. Variables can be renamed in a data step, but this requires customized coding every time a model or variable name changes. Instead, the code we use automates the match between analysis output and PROC MIANALYZE input.

For instance, in order to obtain combined F-test results from PROC GLM, PROC GLMMOD is used to set up effect parameterization for the CLASS variables. The NOINT option is used on the MODEL statement, because GLM subsequently adds an intercept term by default, and the presence of two intercept terms produces an error. The output from GLMMOD is used in PROC

GLM, where no CLASS statement is necessary. The ODS tables are then input into PROC MIANALYZE to obtain combined parameter estimates. We also obtain combined F-tests for main effects and contrasts between age groups. Note that columns of the design matrix that are reference values (parameter estimate=0) are omitted from the PROC MIANALYZE MODELEFFECTS statement. The MULT option on the TEST statement produces the F test. For single effects, this is just the square of the t-test on the combined parameter (automatically produced by PROC MIANALYZE), but the TEST statements are included for convenience in displaying the results. The resulting table can be dropped from the .rtf file directly into a paper.

The sequence of steps needed for obtaining the least-squares means from imputed data is also shown in Appendix 1. The PROC MIXED output can be input directly into PROC MIANALYZE, but the combined results will not contain data for cells (defined by the combination of CLASS variables) that contain no imputed values. These are identified in the PROC MEANS output as having a standard deviation of the estimate equal to 0. Results for the two types of data are combined into a single table and printed, for use in papers or presentations.

Further examples and comments are provided in Appendix 1.

CONCLUSION

MI can be very useful for estimating missing data in multivariate clinical trial data. One of the advantages of the procedure is that, when done correctly, it generates valid inferences from a complex and incomplete clinical study dataset while also accounting for the uncertainty involved in imputing missing values. However, users of these methods must be aware of the extra data preparation needed to successfully use MI, and of the extra processing needed to analyze the imputed multiple data sets. In this paper we have provided detailed information about the steps required to perform such an analysis.

REFERENCES

- Allison PD 2001. Missing Data. *Sage University Papers Series on Quantitative Applications in the Social Sciences*. 07-136. Thousand Oaks, CA., 07-136. Sage.
- Collins LM, Schafer JL, and Kam CM 2001. "A comparison of inclusive and restrictive strategies in modern missing data procedures." *Psychological Methods* 6:330-351.
- Williams TR 2001. "Flexible matching imputation: combining hot-deck imputation with model-based methodology." In Proc. Ann. Meeting of the Am. Statistical Assoc., 5-9.
- Rubin DB 1987. *Multiple Imputation For Nonresponse In Surveys*. New York: John Wiley & Sons, Inc.
- Rubin DB 1980. *Handling Nonresponse in Sample Surveys by Multiple Imputations*, monograph, U.S. Department of Commerce, Bureau of the Census
- Schafer JL 1997. *Analysis of Incomplete Multivariate Data*. Boca Raton: Chapman & Hall/CRC.
- Schafer JL, and J. W. Graham. 2002. Missing data: Our view of the state of the art. *Psychological Methods* 7:147-177.
- Schafer JL & Olsen MK 1998. Multiple imputation for multivariate missing-data problems: A data analyst's perspective. *Multivariate Behavioral Research* 33: 545-571.

CONTACT INFORMATION

Christina M. Gullion, PhD

The Center for Health Research

3800 N. Interstate Ave

Portland OR 97227

Work Phone: 503-335-6506

E-mail: christina.gullion@kpchr.org

Appendix 1. Approaches to combining imputed results from SAS PROCs

Statistic estimated, comments	Sample code
<p>Point estimates can be combined by taking the mean over imputations</p> <p><i>This approach does not provide standard error or standard deviation or any hypothesis test result</i></p>	<pre>%macro combfreq(dsn, var1, var2); proc freq data=&dsn noprint; tables &var1 * &var2 /outpct out=pct_r; by _imputation_ ; run; title 'percentages.'; proc summary nway data=pct_r ; output out=summary mean=; class &var1 &var2; var percent pct_row pct_col ; run; proc print; var &var1 &var2 percent pct_row pct_col; format percent pct_row pct_col 6.2; run; %mend combfreq;</pre>
<p>Mean and standard deviation.</p> <p><i>First calculate means and standard errors. Provide edf (error degrees of freedom). Calculate standard deviation in separate data step.</i></p> <p><i>Meaningless to combine quantiles and percentiles.</i></p>	<pre>proc univariate data=dsn noprint; var var1; output out=outuni mean= var1_m stderr= var1_s; by _imputation_; run; proc mianalyze data=outuni edf=651; ods output parameterestimates=param; modeleffects var1_m; stderr var1_s ; run; * calculate std for the request; data param_std; set param; std= stderr*sqrt(df); run;</pre>
<p>Compare means in 2 independent groups (<i>t</i>-test), adjusting for unequal variances</p> <p><i>Use PROC MIXED with unequal variance option. Provide edf=effect df. Variable names in ODS output are completely compatible with PROC MIANALYZE.</i></p>	<pre>%macro t_test(dsn, class, var); ods listing close; proc mixed data=&dsn; by _imputation_; class &class; model &var=&class / solution ddfm=satterth; repeated /group=&class; ods output solutionf=parms; run; proc mianalyze parms(classvar=full)=parms; ods output parameterestimates=outparms; class &class; modeleffects &class; run; ods listing; title "t-test, class &class, variable &var" ; proc print data=outparms(obs=1) label; id parm;by parm; var estimate stderr tvalue probt; label estimate='diff (1-2)'; run; %mend t_test;</pre>
<p>Analysis of covariance, with F tests of main effects and contrasts between levels of the covariate</p>	<pre>ods listing close; proc glmmod noprint data = dsn outparm=Parm outdesign=Design; by _imputation_; class CAT agegrp ;</pre>

Statistic estimated, comments	Sample code
<p><i>GLMMOD is used to recode class variables as columns in the design matrix;</i></p>	<pre> model Score_i = CAT agegrp / noint; run; ods listing;title5 'Parameter list for ANCOVA model'; proc print data=parm;where _imputation_=1;run; ods listing close; proc glm data = Design; ods output ParameterEstimates=glmparms1 InvXPX=glmxxi1; by _imputation_; model Score_i = col1-col5 /inverse solution ; run;quit; proc mianalyze parms=glmparms1 xpxi=glmxxi1 edf=604; ods output TESTMULTSTAT=Score_i_TEST; modeleffects Intercept Col1 Col3 Col4 ; Numeracy: test col1/mult; Agegrp: test col3,col4/mult; Y_vs_00: test col3 /mult; Y0_vs_00: test col4 /mult; Y_vs_Y0: test col3 - col4/mult; run; ods rtf file="\\&dir\output\JBftests.rtf";ods listing; title4 " F-tests on combined results"; proc print data=Score_i_test label; id test; var numdf dendf fvalue probf; format probf pvalue8.4; label fvalue = 'F' test='Hypothesis test' probf = 'p <'; run; </pre>
<p>Least squares means</p> <p><i>Empty cells are generated for cells with no imputed values; second part of code fills in the empty cells automatically</i></p>	<pre> proc mixed data = dsn; ods output lsmeans=lsn; by _imputation_; class CAT agegrp ; model Score_i = CAT agegrp ; lsmeans agegrp ; lsmeans CAT ; run;quit; proc sort data=LSM; by cat agegrp ; run; proc mianalyze data=LSM edf=602; ods output parameterestimates=ls_means; by cat agegrp ; modeleffects estimate; stderr stderr; run; ** fill in missing values for non-imputed cells; proc means data=LSM noprint; by cat agegrp ; var estimate stderr DF ; output out=ls_std mean = std=sd_est; run; data ls_std;set ls_std; where sd_est = 0; run; data ls_means; merge ls_means ls_std(drop=_TYPE_ _FREQ_); by cat agegrp ; drop parm min max theta0; if sd_est=. then sd_est=1; label sd_est = 'Imputed? (1/0)'; </pre>

Statistic estimated, comments	Sample code
	<pre>run; ods listing; ods rtf file="\&dir\output\&prog_lsmeans.rtf"; title5 "Least squares means: Combined results"; proc print data=ls_means label; id cat;by cat; var agegrp estimate stderr sd_est; format estimate stderr 8.2; run;</pre>
<p>Odds ratios for contrasts in logistic regression.</p> <p><i>After determining final model, use GLMMOD to set up effects for input into PROC MIANALYZE</i></p>	<pre>proc glmmmod data = dsn outparm=parm outdesign=design; by _imputation_; class cat agegrp; model q05_i = cat agegrp / noint; run; ods listing; title5 'parameters for logistic models'; proc print data=parm; where _imputation_=1; run; ods listing close; proc genmod data=design descending; ods output parameterestimates=gmparms parminfo=gmpinfo covb=gmcovb; by _imputation_; model q05_i = col1-col5 / covb dist=b; run; data gmpinfo;set gmpinfo; where parameter in ('prm1', 'prm2', 'prm4', 'prm5'); run; title5 'contrasts, final model'; ods output testmultstat=test; proc mianalyze parms=gmparms covb=gmcovb parminfo=gmpinfo edf=604; modeleffects intercept col1 col3 col4 ; cat: test col1 /mult; agegrp: test col3,col4/mult; gp1vsgp2: test col3-col4/mult; gp1vsgp3: test col3 /mult; gp2vsgp3: test col4 /mult; run;</pre>
<p>PROC GLIMMIX is useful for virtually all models.</p> <p><i>Can do logistic regression with GLIMMIX, but class variables must be coded as binary indicators first. Variable names in ODS output from PROC GLIMMIX are completely compatible with PROC MIANALYZE.</i></p>	<pre>proc glimmix data=dsn order=internal; class aa female tx1 tx2; model wt_cat_i=aa female aa*female tx1 tx2 weight f109_tr f110_tr f111_tr /dist=binary solution; by _imputation_ ; ods output parameterestimates=mixparms; run; proc mianalyze parms=mixparms edf=1000; class aa female tx1 tx2; modeleffects intercept aa female aa*female tx1 tx2 weight f109_tr f110_tr f111_tr; run;</pre>
<p>"Combined" estimates for cells in class variable design that have no imputed data (no variance over imputations).</p>	<pre>ods listing close; title4 'percentages that chose option a'; proc means noprint data=dsn; by _imputation_; class cat agegrp ;var q05_i;</pre>

Statistic estimated, comments	Sample code
<p><i>Rows defined by class variables that have no imputed missing will be empty after PROC MIANALYZE, fill with secondary processing</i></p>	<pre> output out=q05_i n=n mean=estimate stderr=stderr lclm=lclmean uclm=uclmean/autoname; run; data q05_i ;set q05_i(drop= _type_ _freq_) ; if cat = . then cat = -1; if agegrp = '' then agegrp = 'all'; run; proc mianalyze data=q05_i edf=100; ods output parameterestimates=q05_means; by cat agegrp ; modeleffects estimate; stderr stderr; run; ** fill in cells with no imputed values; proc sort out=q05_i ;by cat agegrp;run; proc means data=q05_i noprint; by cat agegrp ; var estimate n; output out=q05_std mean= std=sd_est ; run; data q05_std;set q05_std(drop= _type_ _freq_); if sd_est > 0 then sd_est=.; run; data q05_means_2; update q05_std q05_means ; by cat agegrp ; if sd_est=. then sd_est=1; run; ods listing; proc print data= q05_means_2 label; id cat agegrp; var n estimate sd_est; format estimate percent8.1 cat fcat. n 3.; label estimate='q05_i chose a' agegrp = 'age group' cat = 'category'; sd_est='imputed? [1/0]'; run; </pre>