Paper 3321-2015

Data Summarization for a Dissertation: A Grad Student 'How-To' Paper

Elisa L. Priest, DrPH, Baylor Scott & White Health; Ashley W. Collinsworth, ScD, MPH, Baylor Scott & White Health and Tulane University School of Public Health and Tropical Medicine

ABSTRACT

Graduate students often need to explore data and summarize multiple statistical models into tables for a dissertation. The challenges of data summarization include coding multiple, similar statistical models, and summarizing these models into meaningful tables for review. The default method is to type (or copy and paste) results into tables. This often takes longer than creating and running the analyses. Students might spend hours creating tables, only to have to start over when a change or correction in the underlying data requires the analyses to be updated. This paper gives graduate students the tools to efficiently summarize the results of statistical models in tables. These tools include a macro-based SAS/STAT® analysis and ODS OUTPUT statement to summarize statistics into meaningful tables. Specifically, we summarize PROC GLM and PROC LOGISTIC output. We convert an analysis of hospital-acquired delirium from hundreds of pages of output into three formatted Microsoft Excel files. This paper is appropriate for users familiar with basic macro language.

INTRODUCTION

Graduate students often need to explore data and summarize multiple statistical models into tables for a dissertation. The challenges of data summarization include coding multiple, similar statistical models, and summarizing these models into meaningful tables for review. The default method is to type (or copy and paste) results into tables. This often takes longer than creating and running the analyses. Students might spend hours creating tables, only to have to start over when a change or correction in the underlying data requires the analyses to be updated.

This paper gives graduate students the tools to efficiently summarize the results of statistical models in tables. These tools include a macro-based SAS/STAT® analysis and ODS OUTPUT statement to summarize statistics into meaningful tables. Specifically, we summarize PROC GLM and PROC LOGISTIC output. We convert an analysis of hospital-acquired delirium from hundreds of pages of output into three formatted Microsoft Excel files. This paper is appropriate for users familiar with basic macro language.

PROCESS

1. Create analytic datasets
2. Create sample analysis code
3. Create tables with ODS
4. Convert to %MACRO
5. Format summary tables

1. CREATE ANALYTIC DATASETS

The first step in the process is to create the datasets that will be used in the analysis. This includes becoming familiar with all of the data that you plan to use. Take time to check numeric variables for out of range and missing values, and to check categorical variables for invalid and missing values. Refer to Cody's Data Cleaning Techniques Using SAS for more information. Depending on your analysis, you may

need to recode or categorize variables. It is highly probable that as you go through the analytic process, that you will make modifications to the analytic datasets. Figure 1 describes this process.

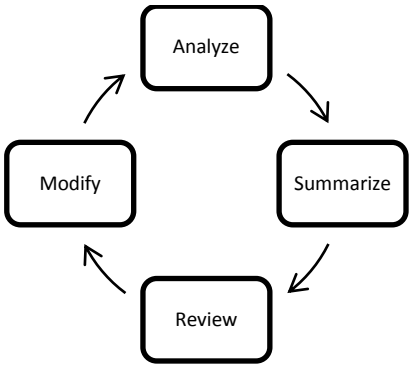


Figure 1. Iterative Process for Data Analysis

2. CREATE SAMPLE ANALYSIS CODE

The next step is to plan out your analysis. In our case, we want to use PROC GENMOD for some outcomes and PROC LOGISTIC for others. We have multiple analyses to perform, so I laid them out in a table to stay organized. In this project, we are examining multiple outcomes, predictors, and control variables. Table 1 is a simplified example of the entire analysis plan. For simplicity, I didn't list all the control variables below- but in your table you would want to list everything out.

Analysis	Outcome	Predictor	Control variables
GENMOD	Delirium days (del_days)	Adherence to intervention (total_adbcde_rate)	n/a
GENMOD	Delirium days (del_days)	Adherence to intervention (total_adbcde_rate)	Yes Unit, month, pat_age….
GENMOD	Coma days (coma_days)	Adherence to intervention (total_adbcde_rate)	n/a
GENMOD	Coma days (coma_days)	Adherence to intervention (total_adbcde_rate)	Yes Unit, month, pat_age….
LOGISTIC	Delirium incidence (delirium_incidence)	Adherence to intervention (total_adbcde_rate)	n/a
LOGISTIC	Delirium incidence (delirium_incidence)	Adherence to intervention (total_adbcde_rate)	Yes Unit, month, pat_age….

Table 1. Example table to organize analyses

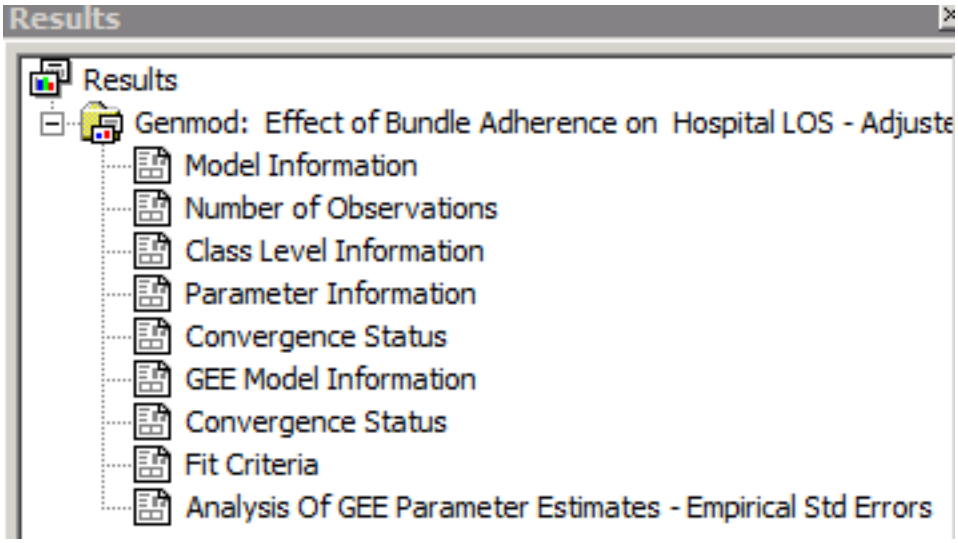
For each analysis, create the code. I'm going to focus on the PROC GENMOD. We have two types of analyses using PROC GENMOD: a basic (crude) analysis and then an analysis controlling for potential

confounding variables. In the crude analysis we are modeling the impact of adherence to an intervention (total_abcde_rate) on total days of delirium (del_days).

```
*Basic Crude analysis;
proc genmod data=work.analysis;
  class unit;
  model del_days= total_abcde_rate / dist=poisson;
  repeated subject=unit / type=ind;
run;
quit;

*Analysis controlling for confounders;
proc genmod data=work.analysis;
  class race insurance (ref='Medicare') unit (ref='Unit 1')/param=ref;
  model del_days= total_abcde_rate unit month pat_age male race hispanic
    insurance charlson APR_DRG_severity sum_vent_los_days icu_los_days
    currentsmoker alcohol dementia surg quality/ type3 dist=poisson;
  repeated subject=unit / type=ind;
run;
quit;
```

Running the PROC GENMOD produces several pages of output. In the results section, you see that each section of the output is listed and you can go directly to that section (Output 1). We are interested in the section "Analysis of GEE Parameter Estimates.." shown in Output 2.



Output 1. Results of PROC GENMOD

Analysis Of GEE Parameter Estimates
Empirical Standard Error Estimates

Parameter	Estimate	Standard Error	95% Confidence Limits		Z	Pr > |Z|
Intercept	-1.2397	0.4550	-2.1314	-0.3480	-2.72	0.0064
total_abcde_RATE	0.4276	0.2812	-0.1235	0.9787	1.52	0.1283

Output 2. PROC GENMOD Analysis of GEE Parameter Estimates

3. CREATE TABLES WITH ODS

ODS TRACE

Out of all of the pages of code, we are primarily interested in the values of the parameter estimates. We want to compare these estimates across our different statistical models. As each PROC GENMOD runs, raw data are produced. The Output Delivery System (ODS) formats these data with table and style definitions to produce output. That means that there are raw data tables that contain all of the statistical parameters displayed in the output window. To identify where these raw data tables are, use the ODS TRACE statement.

```
*Basic Crude analysis;
ODS trace on;
proc genmod data=work.analysis;
  class unit;
  model del_days= total_abcde_rate / dist=poisson;
  repeated subject=unit / type=ind;
run;
quit;
```

CHECK THE LOG

ODS TRACE produces the standard output window along with log messages with all of the ODS object names used to produce the output. The log shows 9 datasets created from the PROC GENMOD: Model Information, Number of observations, Class Level Information, Parameter Information, Convergence Status, GEE Model Information, Convergence Status, Fit Criteria, and Analysis of GEE Parameter Estimates. These correspond to each of the sections of output listed in the Results window (Output 1). Since we already know we are interested in “Analysis of GEE Parameter Estimates..” (Output 2), we look in the log to see what the name of the raw data table is: GEEEmpPEst. (Output 3)

```
Output Added:
-------------
Name:         GEEEmpPEst
Label:        Analysis Of GEE Parameter Estimates - Empirical Std Errors
Template:     stat.genmod.GEEEst
Path:         Genmod.GEEEmpPEst
```

Output 3. ODS TRACE Results in the LOG for PROC GENMOD Analysis of GEE Parameter Estimates

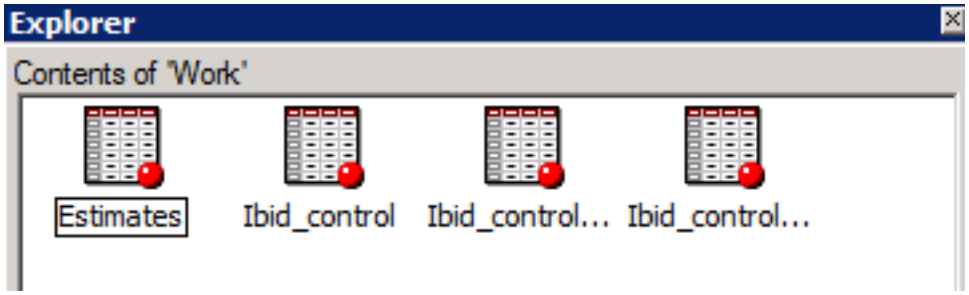
MODIFY CODE TO GET THE RAW DATA

To access the raw data tables, we need to make a small modification to our basic code. We add one line to create a new dataset work.estimates when the code is run:

```
*Basic Crude analysis;
ods output GEEEmpPEst=work.estimates;

proc genmod data=work.analysis;
  class unit;
  model del_days= total_abcde_rate / dist=poisson;
  repeated subject=unit / type=ind;
run;
quit;
```

When the code is run, we see a new table work.estimates in the Explorer window (Display 1). When we open this table (Output 4), we see variables and values that correspond to Output 2. These values are in table format and can modified or used as needed.



Display 1. Creation of da ta table from ODS

	Parameter	Estimate	Empirical Standard Error Estimates	95% Lower Confidence Limit	95% Upper Confidence Limit	Z	Pr > |Z|
1	Intercept	-1.2397	0.4550	-2.1314	-0.3480	-2.72	0.0064
2	total_abcde_RATE	0.4276	0.2812	-0.1235	0.9787	1.52	0.1283

Output 4. GEE Parameter Estimates

4. CONVERT TO %MACRO

To complete our analysis, we would have to code multiple PROC GENMODs. Instead, use macros to condense the repetitive code. We decide to create two different versions of the PROC GENMOD within our %MACRO. The first is the basic crude analysis and the second is the more complex analysis controlling for the confounding variables. Refer to Carpenter's Complete Guide to the SAS macro language for background information on macros.

CONVERT TO %MACRO

This %MACRO is a generalization of PROC GENMOD. We decide to pass several parameters: model (title), option (tells macro which analysis to perform: crude or adjusted), outcome (outcome), pred (predictor), var (confounders to control for), dist (distribution to use), datain (input data set). Because we knew our class variables and their reference categories were not going to change, we hard-coded those in the %MACRO. However, they could have been converted to %MACRO variables along with the others. Every time the %MACRO is called, it creates a new dataset work.estimates with all the values that we are interested in (Output 4).

```
  %macro models (model=, option=, outcome=, pred=, var=, dist=, datain=);
  ods output GEEEmpPEst=work.estimates;

  %if &option.=1 %then %do;

  proc genmod data=work.&datain.;
  class unit;
  model &outcome. = &pred. &var. /dist=&dist.;
  repeated subject=unit / type=ind;
  output out=work.&outcome_&model. Pred=predicted reshi=reschi
  stdreschi=stdreschi;
  run;
  quit;
  %end;

  %if &option.=2 %then %do;

  proc genmod data=work. &datain.;
  class race insurance (ref="Medicare') unti (ref="Unit 1')/param= ref;
  model &outcome. =&pred. &var. /type3 dist=&dist.;
  repeated subject=unit / type=ind.;
  output out=work.&outcome_&model. Pred=predicted reshi=reschi
  stdreschi=stdreschi;
  run;
  quit;
  %end;
%mend models;
```

CALL %MACRO

To call the macro:

```
  %models   (model=CRUDE,
            option=1,
            outcome=del_days,
            pred=total_abcde_rate,
            var=,
            dist=poisson,
            datain=work.analysis,
            );

  %models   (model=TOTAL,
            option=2,
            outcome=del_days,
            pred=total_abcde_rate,
            var= unit month pat_age male race hispanic insurance charlson
                  APR_DRG_ROM APR_DRG_severity sum_vent_los_days icu_los_days
                  Currentsmoker alcohol dementia surg quality,
            dist=poisson,
            datain=work.analysis,
            );
```

5. FORMAT SUMMARY TABLES

The final step is to reformat the values in work.estimates (Output 4) into a final table.

REORGANIZE AND FORMAT

The code below uses the DATA step within the %MACRO to reformat work.estimates (Output 4) into work.estimates1 (Output 5). This new table lists the model, predictor, outcome, effect, confidence limit, and p-value. If the p-value is <0.05, there will be a '*' on the effect variable.

	model	Parameter	outcome	effect	effectCL	p
1	CRUDE	TOTAL_ABCDE_RATE	del_days	0.43	(-0.12, 0.98)	0.1283

Output 5. Reformatted data

```
Data work.estimates1;
    Length model $15. Parm$50. Outcome $50. Level1 $50. Effect $12.
          Sigflag $2. ;
    Merge work.estimates;
    Outcome="&outcome.";
    Parm=upcase (parm);
    If parm="intercept" then delete;
    If parm="Dispersion" then delete;
    If parm="Scale" then delete;
    Model="&model.";
    Effect_=round ((estimate), 0.01);
    effectLL=round((lowerCL,0.01);
    effectUL=round((uppercl),0.01);
    if probz<0.05 then do;
          sigflag="*";
          end;
    effect=strip(effect_)||strip (sigflag);
    effectCL="("|| strip(effectLL)|| ", " ||strip (effectul || ")";
    p=probz;
    if parm=upcase("&pred.");
    keep model parm outcome effect effectCL p;
    format p pvalue6.4;
run;
```

COMBINE ACROSS %MACRO CALLS

The final section of the %MACRO code combines the latest %MACRO call summary estimates with any previous %MACRO calls. Each type of model (Crude, Total, etc) is summarized together in a separate table (Output 6.)

```
proc datasets library=work force noprint;
    Append base=&model. Data=estimates1;
 run;
 quit;
```

The output below shows the crude results across multiple outcomes. Notice how the effect for coma_days is -1.35* . The "*" indicates that the p-value is significant. Also note that we used the p-value format pvalue6.4 to format the values.

	model	Parameter	outcome	effect	effectCL	p
1	CRUDE	TOTAL_ABCDE_RATE	del_days	0.43	(-0.12, 0.98)	0.1283
2	CRUDE	TOTAL_ABCDE_RATE	DEL_FREE_DAYS_ALL	0.02	(-0.16, 0.21)	0.7953
3	CRUDE	TOTAL_ABCDE_RATE	coma_days	-1.35*	(-1.65, -1.04)	<.0001
4	CRUDE	TOTAL_ABCDE_RATE	coma_free_days_all	0.15	(-0.05, 0.34)	0.1373
5	CRUDE	TOTAL_ABCDE_RATE	all_del_coma	0.13	(-0.07, 0.32)	0.2106
6	CRUDE	TOTAL_ABCDE_RATE	sum_vent_los_days	-0.34*	(-0.5, -0.18)	<.0001
7	CRUDE	TOTAL_ABCDE_RATE	icu_los_days	0.05	(-0.13, 0.23)	0.6057
8	CRUDE	TOTAL_ABCDE_RATE	los_calc	0.25*	(0.1, 0.41)	0.0014

Output 6. Reformatted data

CONCLUSION

In conclusion, this paper gives graduate students the tools to efficiently summarize the results of statistical models in tables. These tools include a macro-based SAS/STAT® analysis and ODS OUTPUT statement to summarize statistics into meaningful tables. Specifically, we summarized PROC GLM and the same process can be followed for PROC LOGISTIC. We converted an analysis of hospital-acquired delirium from hundreds of pages of output into formatted Microsoft Excel files. This method can save hours of manual typing and summarizing into tables.

REFERENCES

Carpenter, A. Carpenter's complete guide to the SAS macro language. SAS Institute. Cary, NC, 2004.

Cody, R. Cody's data cleaning techniques using SAS, Second Edition. SAS Institute. Cary, NC 2008

Haworth, L. Output Delivery System: The basics. Chapter 6: Exploring ODS Output. SAS Institute, Cary NC, 2001.

Haworth, L. Output Delivery System: The basics. Chapter 7: Output Data Sets. SAS Institute, Cary NC, 2001.

ACKNOWLEDGEMENTS

This project was supported by a grant (R18HS021459) from the Agency for Healthcare Research and Quality (AHRQ).

CONTACT INFORMATION

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

Elisa L. Priest
elisapriest@hotmail.com