Paper 241-2009

Easier Exploratory Analysis for Epidemiology: A Grad Student 'How-To' Paper

Elisa L. Priest[1,2] Brian Adams[2], Lori A. Fischbach[1]
[1] University of North Texas School of Public Health, Epidemiology Department
[2] Institute for Health Care Research and Improvement, Baylor Health Care System

ABSTRACT

In epidemiology, exploratory analyses of existing data can screen new hypotheses and provide rationale for future studies. Graduate students often perform an exploratory analysis for a dissertation because it is cost- and time-efficient.

Exploratory analyses may examine multiple variables and outcomes across strata of data. Thousands of pages of statistical output may need to be examined for meaningful patterns. The challenges of an exploratory analysis are 1) Organization: Which variables do you want to examine?; 2) Execution: How do you code hundreds of statistical models?; and 3) Summarization: How do you combine hundreds of results across models?

This paper gives graduate students the tools to perform an efficient exploratory analysis. These tools include 1) an Excel control document for organization of variables; 2) a macro-based SAS/STAT® analysis to simplify analysis; and 3) ODS OUTPUT to summarize statistics in meaningful tables. This is appropriate for users familiar with basic macro language. We convert an analysis of *H. pylori* infection and gastroesophageal disease outcomes from 4,000 pages of output to three Excel files.

INTRODUCTION

In epidemiology, exploratory analyses of existing data can screen new hypotheses and provide rationale for future studies. Graduate students often perform an exploratory analysis for a dissertation because it is cost- and time-efficient.

Exploratory analyses may examine multiple variables and outcomes across strata of data. Thousands of pages of statistical output may need to be examined for meaningful patterns. The challenges of an exploratory analysis are 1) Organization: Which variables do you want to examine?; 2) Execution: How do you code hundreds of statistical models?; and 3) Summarization: How do you combine hundreds of results across models?

This paper gives graduate students the tools to perform an efficient exploratory analysis. These tools include 1) an Excel control document for organization of variables; 2) a macro based SAS/STAT® analysis to simplify analysis; and 3) ODS OUTPUT to summarize statistics in meaningful tables. This is appropriate for users familiar with basic macro programming.

For this paper, we convert output from an exploratory analysis into useful tables using macro code, an Excel control table, and ODS OUTPUT. This is an exploratory analysis data collected during a randomized controlled trial in Columbia. The primary analysis of the data was completed and the investigator wished to examine 148 variables in 5 categories: diet, non-modifiable, family history, medication, histological, and disease factors. All variables were to be looked at 7 different ways (overall, and stratified 6 different ways) and across three primary outcomes: progression of disease, initiation of disease, and presence of Barrett's Esophagus. The 270 pages of code for all the 3108 PROC PHREG models created 19,513 pages of SAS output. The investigator began to tabulate all the results from the models by hand and quickly became overwhelmed. For our example, we will focus on the 41 diet factors (861 models, 52 pages of code, and 4096 pages of output). The analysis can easily be expanded to include the remaining variables.

PROCESS

- Organize
  1. Gather requirements and create an Excel control document
- Execution and Summarization
  2. Basic statistics and ODS output
  3. Convert Statistics to %MACRO
  4. Combine datasets in %MACRO
  5. Reorganize, format and output

1. GATHER REQUIREMENTS AND CREATE AN EXCEL CONTROL DOCUMENT

Our program has three requirements: 1) the overall analysis for each of the 41 variables; 2) the analysis stratified by 6 different variables; and 3) all of the analyses done with three different outcome variables. The first will use our Excel control document to create a list of variables. A macro will then perform the overall analysis and use a loop to perform the 6 stratified analyses. To complete the third requirement, we will call the macro once for each outcome.

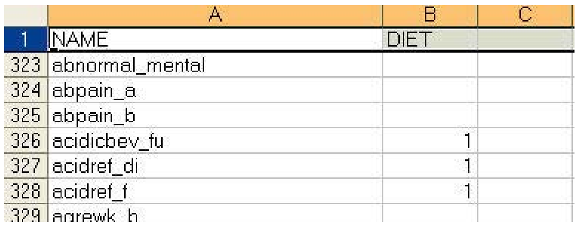
First, create a list of variables to examine. We used an Excel worksheet to organize the requirements for the analysis. To create the worksheet without retyping all the variables, use PROC CONTENTS with an OUTPUT statement to write the variables to a data set and PROC EXPORT to convert the list to Excel format (Code1).[1]

Code 1: Create list of variables in Excel

```
proc contents data=work.data varnum out=work.contents;
run;

data work.variables;
    set work.contents;
    keep name ;
run;

proc export DATA= work.variables
            OUTFILE= "C:\location\Variables.xls "
            DBMS=EXCEL REPLACE;
     SHEET="Variables";
run;
```

Next, identify the variables for analysis. There are at least two options: 1) only keep the variables for the analysis or 2) leave all the variables from the dataset and use a flag to indicate which ones are used. We chose the second option to allow us to expand the analysis in future iterations. The flag column is labeled ‘DIET’ and a flag of ‘1’ is used to indicate that the variable will be used in the diet analysis (Figure 1). This method can be used to organize variables into meaningful categories, or to indicate outcome variables, stratification factors, or variables to use during modeling (see Appendix A). It also provides essential documentation of the analysis.

Figure 1. Create flags for analysis in Excel



	A	B	C
1	NAME	DIET	
323	abnormal_mental		
324	abpain_a		
325	abpain_b		
326	acidicbev_fu	1	
327	acidref_di	1	
328	acidref_f	1	
329	agrewk_b		

2. BASIC STATISTICS AND ODS OUTPUT

Next, choose the statistical models for analysis. This analysis required PROC PHREG to account for person years of follow-up. Code 2 illustrates the PROC PHREG for the overall analysis of follow up acidic beverage consumption (variable acidicbev_fu) with the outcome disease progression (variable progress) and Code 3 shows the PROC PHREG for the same variables stratified by baseline atrophy in the corpus region of the stomach (variable corpusatro_b).

Code 2. Example overall analysis for the outcome progress

```
proc phreg data=work.data;
    model pyrs*progress(0) = acidicbev_fu / risklimits;
run;
```

Code 3. Example stratified analysis for the outcome progress

```
proc sort data=work.data; by corpusatro_b;
run;

proc phreg data=work.data; by corpusatro_b;
    model pyrs*progress(0) = acidicbev_fu / risklimits;
run;
```

Once the statistical models are chosen, select the portions of the output to examine. As each DATA and PROC step run, raw data are produced. The Output Delivery System (ODS) formats these data with table and style definitions to produce output.[2] As PROC PHREG runs, SAS creates raw data tables that contain all of the statistical parameters displayed in the output window. Because these raw data tables are not automatically available in the work directory, use the ODS TRACE statement to identify them (Code 4).

ODS TRACE produces the standard output window display along with log messages with all of the ODS object names used to produce the output. We could also use the ODS TRACE with the LISTING option to write the ODS object names directly to the output window[2] (Code 5). However, this increases the output. The log messages show 7 datasets created from the PROC PHREG: Model Information, Number of observations, Summary of event and censored observations, Convergence status, Model fit statistics, Test of global null hypothesis, and Maximum likelihood estimates of model parameters. Next, compare the output window to the log to identify which of these datasets contain the statistics of interest (Figure 2 and Figure 3). In this example, we select the Hazard Ratio, 95% CI for the Hazard Ratio, and the p-value. These statistics are located in the dataset labeled ‘Maximum Likelihood Estimates of Model Parameters’.

Code 4. ODS to identify raw data

```
ods trace on;

proc phreg data=work.data;
    model pyrs*progress(0) = acidicbev_fu / risklimits;
run;

ods trace off;
```

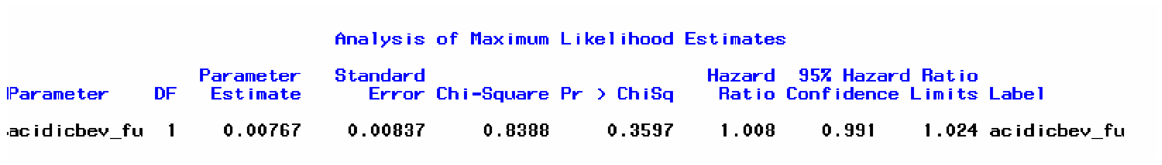
Code 5. ODS to identify raw data (alternative)

```
ods trace on/listing ;

proc phreg data=work.data;
    model pyrs*progress(0) = acidicbev_fu / risklimits;
run;

ods trace off;
```

Figure 2. Log for Code 4: ODS to identify raw data

```
-------------
Name:       ParameterEstimates
Label:      Maximum Likelihood Estimates of Model Parameters
Template:   Stat.Phreg.ParameterEstimates
Path:       Phreg.ParameterEstimates
-------------
```

Figure 3. Output for Code 4: ODS to identify raw data



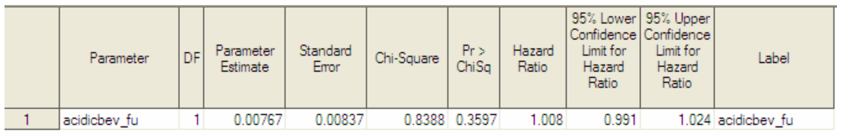
```
                         Analysis of Maximum Likelihood Estimates

                 Parameter   Standard                          Hazard   95% Hazard Ratio
Parameter    DF   Estimate      Error Chi-Square Pr > ChiSq     Ratio Confidence Limits Label

acidicbev_fu  1    0.00767    0.00837     0.8388     0.3597     1.008     0.991     1.024 acidicbev_fu
```

Next, access the raw data tables that were identified in the ODS output[3] (Code 6.). Use the ODS OUTPUT statement with PARAMETERESTIMATES=*dataset name* to create a data set called work.Overall. This data set contains 10 variables, including all the statistics we needed: Hazard Ratio, 95% CL for the Hazard Ratio, and p-value (Figure 4). The analysis variable called acidic beverage (acidicbev_fu) is indicated by the column header 'Parameter'. PROC CONTENTS gives the complete variable names and labels in the dataset (Figure 5). Repeating this process for the stratified analysis gives one additional variable in the output data set (Figure 6). This variable name indicates the variable used for the stratification (corpusatrophy_fu) and the values (0,1) indicate the levels of stratification. Also notice that there are two rows of statistics in this data set. Each row contains the statistics for one level of stratification.

Code 6. Access ODS raw data

```
ods output parameterestimates=work.Overall;

proc phreg data=work.data;
    model pyrs*progress(0) = acidicbev_fu / risklimits;
run;

ods output close;

proc contents data=work.overall varnum;
run;
```

Figure 4. Overall data set

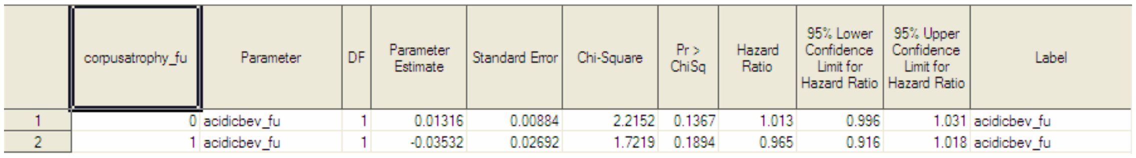


	Parameter	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio	95% Lower Confidence Limit for Hazard Ratio	95% Upper Confidence Limit for Hazard Ratio	Label
1	acidicbev_fu	1	0.00767	0.00837	0.8388	0.3597	1.008	0.991	1.024	acidicbev_fu

Figure 5. Output for Code 6: Variables in Overall data set

```
#     Variable       Type    Len    Format        Label
1     Parameter      Char     12
2     DF             Num       8    2.
3     Estimate       Num       8    D10.          Parameter Estimate
4     StdErr         Num       8    D10.          Standard Error
5     ChiSq          Num       8    10.4          Chi-Square
6     ProbChiSq      Num       8    PVALUE6.4     Pr > ChiSq
7     HazardRatio    Num       8    8.3           Hazard Ratio
8     HRLowerCL      Num       8    8.3           95% Lower Confidence Limit for HR
9     HRUpperCL      Num       8    8.3           95% Upper Confidence Limit for HR
10     Label          Char     12
```

Figure 6. Stratified data set



	corpusatrophy_fu	Parameter	DF	Parameter Estimate	Standard Error	Chi-Square	Pr > ChiSq	Hazard Ratio	95% Lower Confidence Limit for Hazard Ratio	95% Upper Confidence Limit for Hazard Ratio	Label
1	0	acidicbev_fu	1	0.01316	0.00884	2.2152	0.1367	1.013	0.996	1.031	acidicbev_fu
2	1	acidicbev_fu	1	-0.03532	0.02692	1.7219	0.1894	0.965	0.916	1.018	acidicbev_fu

3. CONVERT TO %MACRO

To complete the requirements for this analysis would require coding 861 PROC PHREG models. Instead, use macros to condense the repetitive code. We will start simple and build in complexity. Please refer to Carpenter's Complete Guide to the SAS macro language for background information on macros.[6]

OVERALL PHREG

Convert the overall PROC PHREG to a %MACRO. This %MACRO is a generalization of PROC PHREG and we need to pass two parameters: cvar (covariate) and outcome (outcome) to the %MACRO. Call %MACRO stats using the variables in Code 4 (cvar= acidicbev_fu, outcome=progress). This produces an output data set called work.Overall that includes the hazard ratio statistics for the association between acidicbev_fu (cvar) and progress (outcome).

Code 7. Overall %MACRO stats with macro call

```
%macro Stats (cvar=, outcome=);
*Stats Overall*;

ods output parameterestimates=work.Overall;

proc phreg data=work.data;
    model pyrs*&outcome.(0) = &cvar. / risklimits;
run;

ods output close;

%mend stats;

*call macro to test*;
%stats (cvar=acidicbev_fu, outcome=progress);
```

STRATIFIED PHREG

Add the analysis for a stratified variable to %MACRO stats. Because there are 6 different ways to stratify the data, create a macro variable (&strata) to indicate the stratification variable.

Code 8. Stratified %MACRO stats

```
*Stats Stratified*;
proc sort data=work.data;
   by &strata.;
run;

ods output parameterestimates=work.Overall;

proc phreg data=work.data;
    by &strata.;
    model pyrs*&outcome.(0) = &cvar. / risklimits;
run;
```

Next, modify the code to loop through this section once for each of the 6 ways to stratify the data. Because there are only a few variables, create a macro list (%LET stratalist=) and %SCAN to choose the appropriate variable. Refer to Fehd, 2007 for an overview of macro list processing and alternative methods.[4] The complete %MACRO stats including the %DO loop is below (Code 9). Notice that the ODS OUTPUT statement was modified to update the name of the data set depending on the count (&i). When %MACRO stats is called, 7 datasets are created: work.Overall and work.Stratified1-work.Stratified6.

Code 9. Complete Stratified %MACRO stats

```
%macro stats (cvar=, outcome=);

*Stats Overall*;
ods output parameterestimates=work.Overall;

proc phreg data=work.data;
    model pyrs*&outcome.(0) = &cvar. / risklimits;
run;

ods output close;

*Stats Stratified*;
*List of stratification variables*;
%let strataList= corpusatrophy_fu corpusatro_b corpuspoly_b corpuspmn_fu corpushp_fu
corpushp_b ;

*do the following 6 times, once for each strata var*;
%do i= 1 %to 6;

*assign macro var strata to the next in the list*;
          %let strata= %scan(&strataList, &i., " ");

          proc sort data=work.data;
             by &strata.;
          run;

          ods output parameterestimates=work.stratified&i.;

          proc phreg data=work.data;
                by &strata.;
                model pyrs*&outcome.(0) = &cvar. / risklimits;
          run;

          ods output close;
%end;

%mend stats;

*call macro to test*;
%stats (cvar=acidicbev_fu, outcome=progress);
```

CALL %MACRO STATS FOR EACH &CVAR

Now that the %MACRO stats is complete, call it one time for each diet variable (&cvar). Use PROC IMPORT to convert the Excel control document into a SAS data set and keep only the variables flagged in the 'DIET' column. PROC SORT the dataset by length so that the longest variable name is first. This becomes important when combining datasets. DATA _null_ counts how many variables are in the dataset and assigns this to the macro variable &numobs (Code 10).

The next section of code is the %DO %WHILE loop (Code 11). The macro variable &passcount is the counter for the %DO %WHILE loop which assigns the appropriate variable name in the list to the macro variable &cvar using CALL SYMPUT. Next, the macro call for the %MACRO stats is issued and &cvar and &outcome are passed to the PROC PHREGs to create ODS data sets. Finally, the macro variable &passcount is incremented so that the next variable name in the data set will be read during the next loop. When %MACRO diet is called, the %MACRO stats runs 41 times, creates over 1000 pages of output, and creates 7 datasets: work.Overall and work.Stratified1-work.Stratified6.

However, notice that the data sets contain only the observations from the last iteration of the %DO %WHILE loop. This will be modified in the next step.

Code 10. %MACRO diet to call %MACRO stats once for each &cvar

```
%macro diet (outcome=);

*import dataset of variables*;
PROC  IMPORT OUT= WORK.VARS
            DATAFILE= "C:\My Documents\SAS Presentation\VARIABLES.xls"
            DBMS=EXCEL REPLACE;
     SHEET="Variables";
     GETNAMES=YES;
     MIXED=YES;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;

*keep those vars for diet analysis*;
data work.vars_diet;
set work.vars;
    if diet=1;
    Length=Length(name);
    keep name Length;
run;

*sort by length so longest is first*;
proc sort data=work.vars_diet ;
     by descending length;
run;

*assign count of variable to macro variable*;
    data _null_;
      call symput ('Numobs', left(put(dsnobs,5.)));
      stop;
      set work.vars_diet nobs=dsnobs;
run;
```

Code 11. %DO %WHILE loop

```
*loop through entire list*;
%let passCount = 1;

    %do %while(&passCount. <= &numobs.);
           data _null_;
           set work.VARS_diet (firstobs=&passCOUNT.
                  obs=&passCOUNT.);
                  call symput ('CVAR', NAME);
           run;
           *double check macro vars in the log*;
           %put &cvar. ;
           %put &outcome.;

*call statistics macro for each variable/passcount*;
           %stats (cvar=&cvar., outcome=&outcome.);
    *increase counter to go to next diet variable in the list*;
           %let passcount=%eval(&passCount+1);
    %end;
%mend diet;

*Macro call for %macro diet ;
%diet (outcome=progress);
```

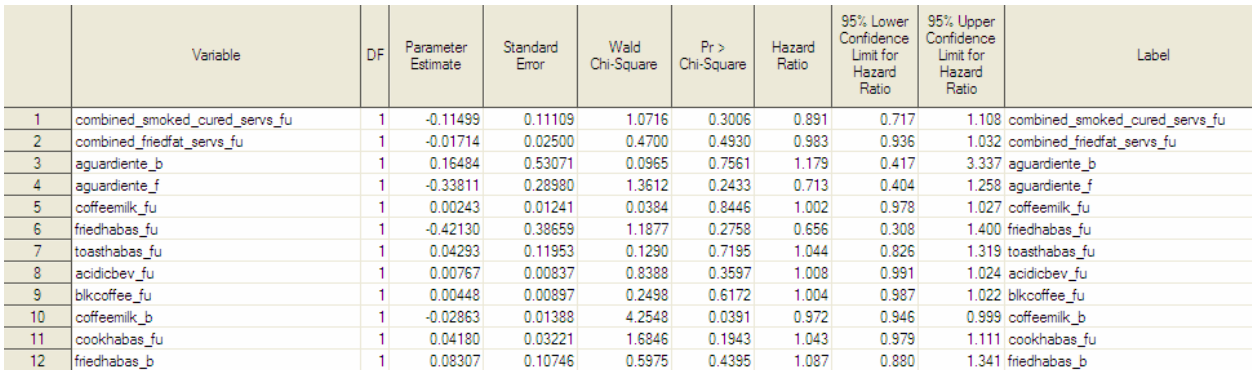
4. COMBINE DATASETS IN %MACRO

Next, %MACRO diet should be modified to roll up data across each of the iterations of the %DO %WHILE loop. One method to do this is illustrated for the Overall dataset. In the %DO %WHILE loop, create a data set that is either initialized at the first loop (&passcount=1) or added if the macro variable &passcount is greater than 1 (Code 12). Because the macro variable &outcome is assigned the value “progress” in our macro call (Code 11), the resulting data set is work.Overall_Progress and contains one row for each of the 41 diet variables (Figure 7). This method of rolling up datasets can be repeated for each of the 6 stratified analyses datasets. We nested our code for the stratified datasets inside a %DO loop to repeat once for each stratification variable (Code 13) to produce data set Stratified_Progress (Figure 8).

Code 12. Roll up Overall data set across iterations of %DO %WHILE loop

```
   *roll up into one dataset*;
             %if &passcount. =1 %then %do;
                     data work.Overall_&Outcome. ;
                             set work.Overall;
                     run;

             %end;

             %if &passcount.>1 %then %do;
                     data work.Overall_&Outcome. ;
                             set work.Overall_&Outcome. WORK.Overall;
                     run;
             %end;
```

Figure 7. Data set Overall_Progress showing the first 12 of 41 diet variables

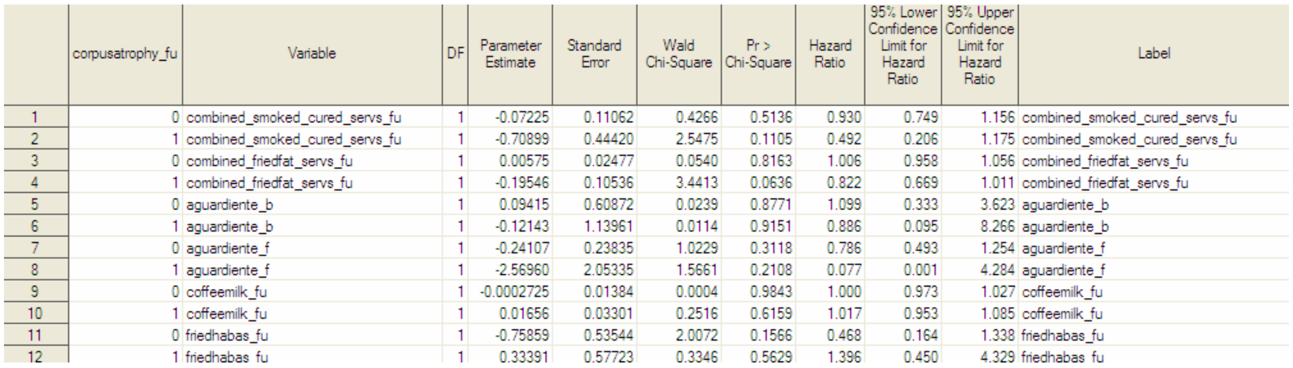


	Variable	DF	Parameter Estimate	Standard Error	Wald Chi-Square	Pr > Chi-Square	Hazard Ratio	95% Lower Confidence Limit for Hazard Ratio	95% Upper Confidence Limit for Hazard Ratio	Label
1	combined_smoked_cured_servs_fu	1	-0.11499	0.11109	1.0716	0.3006	0.891	0.717	1.108	combined_smoked_cured_servs_fu
2	combined_friedfat_servs_fu	1	-0.01714	0.02500	0.4700	0.4930	0.983	0.936	1.032	combined_friedfat_servs_fu
3	aguardiente_b	1	0.16484	0.53071	0.0965	0.7561	1.179	0.417	3.337	aguardiente_b
4	aguardiente_f	1	-0.33811	0.28980	1.3612	0.2433	0.713	0.404	1.258	aguardiente_f
5	coffeemilk_fu	1	0.00243	0.01241	0.0384	0.8446	1.002	0.978	1.027	coffeemilk_fu
6	friedhabas_fu	1	-0.42130	0.38659	1.1877	0.2758	0.656	0.308	1.400	friedhabas_fu
7	toasthabas_fu	1	0.04293	0.11953	0.1290	0.7195	1.044	0.826	1.319	toasthabas_fu
8	acidicbev_fu	1	0.00767	0.00837	0.8388	0.3597	1.008	0.991	1.024	acidicbev_fu
9	blkcoffee_fu	1	0.00448	0.00897	0.2498	0.6172	1.004	0.987	1.022	blkcoffee_fu
10	coffeemilk_b	1	-0.02863	0.01388	4.2548	0.0391	0.972	0.946	0.999	coffeemilk_b
11	cookhabas_fu	1	0.04180	0.03221	1.6846	0.1943	1.043	0.979	1.111	cookhabas_fu
12	friedhabas_b	1	0.08307	0.10746	0.5975	0.4395	1.087	0.880	1.341	friedhabas_b

Code 13. Roll up stratified data set across iterations of %Do %WHILE loop

```
             %do j= 1 %to 6;
                     %if &passcount. =1 %then %do;
                             data work.Stratified&J._&Outcome. ;
                                     set work.Stratified&J.;
                             run;

                     %end;

                     %if &passcount.>1 %then %do;
                             data work.Stratified&J._&Outcome. ;
                                     set work.Stratified&J._&Outcome.
                                     work.Stratified&J.;
                             run;
                     %end;
             %end;
```

Figure 8. Data set Stratified_Progress showing the first 6 of 41 diet variables



	corpusatrophy_fu	Variable	DF	Parameter Estimate	Standard Error	Wald Chi-Square	Pr > Chi-Square	Hazard Ratio	95% Lower Confidence Limit for Hazard Ratio	95% Upper Confidence Limit for Hazard Ratio	Label
1	0	combined_smoked_cured_servs_fu	1	-0.07225	0.11062	0.4266	0.5136	0.930	0.749	1.156	combined_smoked_cured_servs_fu
2	1	combined_smoked_cured_servs_fu	1	-0.70899	0.44420	2.5475	0.1105	0.492	0.206	1.175	combined_smoked_cured_servs_fu
3	0	combined_friedfat_servs_fu	1	0.00575	0.02477	0.0540	0.8163	1.006	0.958	1.056	combined_friedfat_servs_fu
4	1	combined_friedfat_servs_fu	1	-0.19546	0.10536	3.4413	0.0636	0.822	0.669	1.011	combined_friedfat_servs_fu
5	0	aguardiente_b	1	0.09415	0.60872	0.0239	0.8771	1.099	0.333	3.623	aguardiente_b
6	1	aguardiente_b	1	-0.12143	1.13961	0.0114	0.9151	0.886	0.095	8.266	aguardiente_b
7	0	aguardiente_f	1	-0.24107	0.23835	1.0229	0.3118	0.786	0.493	1.254	aguardiente_f
8	1	aguardiente_f	1	-2.56960	2.05335	1.5661	0.2108	0.077	0.001	4.284	aguardiente_f
9	0	coffeemilk_fu	1	-0.0002725	0.01384	0.0004	0.9843	1.000	0.973	1.027	coffeemilk_fu
10	1	coffeemilk_fu	1	0.01656	0.03301	0.2516	0.6159	1.017	0.953	1.085	coffeemilk_fu
11	0	friedhabas_fu	1	-0.75859	0.53544	2.0072	0.1566	0.468	0.164	1.338	friedhabas_fu
12	1	friedhabas_fu	1	0.33391	0.57723	0.3346	0.5629	1.396	0.450	4.329	friedhabas_fu

5. REORGANIZE, FORMAT, AND OUTPUT

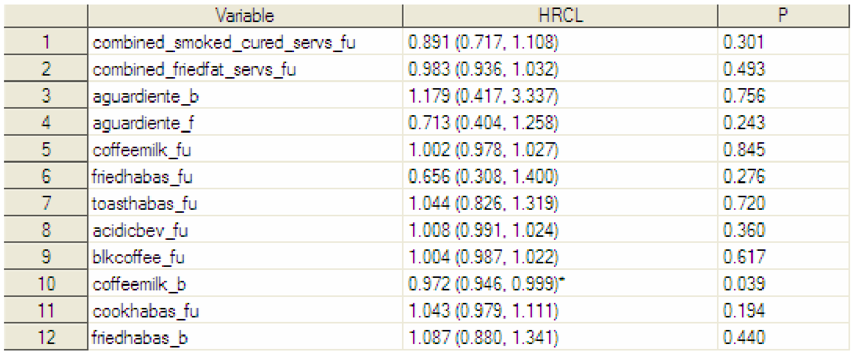
Now that all of the values are in datasets instead of the output window, we can manipulate them in a multitude of ways. We will use the Overall data set as an example. Character and numeric functions such as PUT and ROUND converted the hazard ratios and confidence limits to character values. Concantenation || creates a new variable, HRCL, which combines the hazard ratio and confidence limits. A special character, '*', indicates whether the p-value was less than 0.05. The resulting data set work.Overall_Progress_Final is exported to Excel for review. Additional ideas for manipulation of data to produce reports can be found in Bhaskar (2004).[5] This manipulation could also be generalized and placed into a macro.

Finally, we call our %MACRO diet one time for each of the three outcomes in the requirements (Code 14) to produce data set Overall_Progress_Final (Figure 9).

Code 14. Format data and export

```
*Format data*;
data work.Overall_Progress_Final;
set work.Overall_Progress;
  HR= put (round(hazardratio,0.001),10.3);
  UCL= put (round (hruppercl,0.001),10.3);
  LCL=put (round (hrlowercl,0.001), 10.3);
  if probChiSq<=0.05 then
  HRCL=strip(hr) || " (" || strip(lcl) || ", " || strip(ucl) || ")*";
  if probChiSq>0.05 then
  HRCL=strip(hr) || " (" || strip(lcl) || ", " || strip(ucl) || ")";
  P=strip(put (round(probchisq,0.001),10.3));
  keep variable hrcl p;
run;

proc export DATA= work.Overall_Progress_Final
            OUTFILE= "C:\location\Progress.xls"
            DBMS=EXCEL REPLACE;
     SHEET="Overall";
run;
```

Figure 9. Organized and formatted data set Overall_Progress_Final showing the first 12 rows



	Variable	HRCL	P
1	combined_smoked_cured_servs_fu	0.891 (0.717, 1.108)	0.301
2	combined_friedfat_servs_fu	0.983 (0.936, 1.032)	0.493
3	aguardiente_b	1.179 (0.417, 3.337)	0.756
4	aguardiente_f	0.713 (0.404, 1.258)	0.243
5	coffeemilk_fu	1.002 (0.978, 1.027)	0.845
6	friedhabas_fu	0.656 (0.308, 1.400)	0.276
7	toasthabas_fu	1.044 (0.826, 1.319)	0.720
8	acidicbev_fu	1.008 (0.991, 1.024)	0.360
9	blkcoffee_fu	1.004 (0.987, 1.022)	0.617
10	coffeemilk_b	0.972 (0.946, 0.999)*	0.039
11	cookhabas_fu	1.043 (0.979, 1.111)	0.194
12	friedhabas_b	1.087 (0.880, 1.341)	0.440

CONCLUSION

The challenges of an exploratory analysis are 1) Organization: Which variables do you want to examine?; 2) Execution: How do you code hundreds of statistical models?; and 3) Summarization: How do you combine hundreds of results across models?.

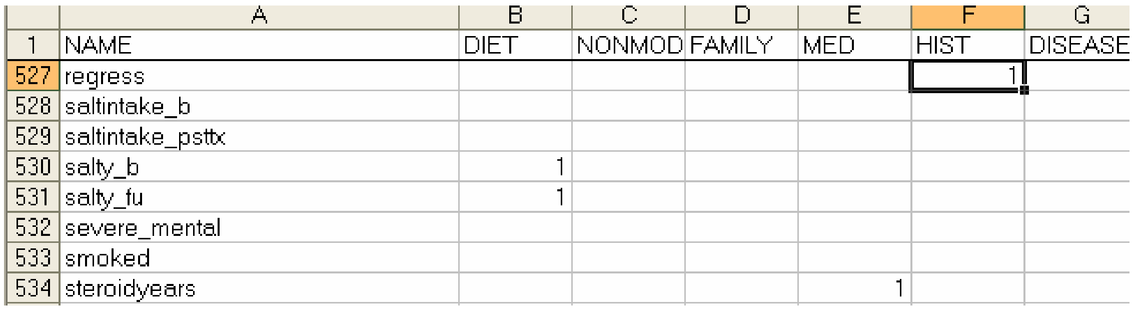
This paper provided the tools to perform an efficient exploratory analysis. These tools included 1) an Excel control document for organization of variables; 2) a macro based SAS/STAT analysis to simplify analysis; and 3) ODS OUTPUT to summarize statistics in meaningful tables.

The results of converting the exploratory analysis to a macro justified the time spent modifying the code. The 270 pages of code required originally for the 3108 PROC PHREG models was compressed to 6 pages and the 19,513 pages of SAS output was presented in three Excel files with three worksheets each.

This method is extremely flexible and can be combined with any ODS output objects including model fit statistics or FREQS and MEANS for descriptive statistics. The Excel control document can be as simple or complex as needed and provides essential documentation of the analysis. In addition, the resulting statistics can be manipulated with any available functions and could drive other SAS procedures like TABULATE or REPORT.

APPENDIX A

Excel control document with multiple variable categories



	A	B	C	D	E	F	G
1	NAME	DIET	NONMOD	FAMILY	MED	HIST	DISEASE
527	regress					1	
528	saltintake_b						
529	saltintake_psttx						
530	salty_b	1					
531	salty_fu	1					
532	severe_mental						
533	smoked						
534	steroidyears				1		

REFERENCES

1) Priest EL. Data requests, quickly. South Central SAS Users Group 2006 Conference, Irving, TX, October 15-18, 2006.

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

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

4) Fehd, RJ. List processing basics: Creating and using lists of macro variables. SAS Global Forum 2007.

5) Bhaskar B, and Murray K. Generating customized analytical reports from SAS procedure output. NESUG 17, 2004.

6) Carpenter, A. Carpenter's complete guide to the SAS macro language. SAS Institute. Cary, NC, 2004.
This is the text for the references.

ACKNOWLEDGMENTS

Thank you to Dr. Tim Church and Dr. Lori Fischbach for their data analysis projects which led to the development of the code in this paper.

CONTACT INFORMATION

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

Elisa L. Priest, MPH
elisapriest@hotmail.com