Paper 256-2010

Let SAS® Do the Work: Correlation Crossroads

Theresa Gilligan[1], MS; Cheryl Coon[1], PhD; Lauren Nelson[1], PhD; Lori McLeod[1], PhD;
Yung-Jui Yang[2], MA
[1]RTI Health Solutions, RTP, NC
[2]University of Illinois at Urbana-Champaign

ABSTRACT

Patient-reported outcome (PRO) studies often require correlation analyses using multiple variable types. Producing tables that contain Pearson, polychoric, and polyserial correlations for ordinal and categorical variables is challenging in Base SAS®. This paper introduces a SAS macro called %PROCORR that routes variables to the appropriate correlation computation. The macro also identifies the appropriate output statistics for each type of correlation and creates a report-ready table. The analyses and output discussed were created with Base SAS Version 9.2 software and apply to analysts with macro experience.

INTRODUCTION

SAS code and statistics have at least one thing in common: there is always more than one way to complete the task. With bivariate measures of association in statistics, however, some procedures are more appropriate than others, and they may yield very different results.

Correlation coefficients are one of the most widely computed statistics across industries. The word “correlation” is usually associated with the Pearson product-moment correlation. Yet, there are many other types of correlations. A quick review of the literature highlights at least six different types of correlations outside of Pearson, including Spearman’s rho, point-biserial, the phi-coefficient, tetrachoric, polychoric, and polyserial. Each type describes the linear relationship between two variables. In SAS, there are many programming choices to produce the correlation coefficients between two variables. However, not all correlation types should be treated as interchangeable and the choice should not be made blindly. Depending on the types of analysis variables and their distributions, it is up to the analyst to choose the most appropriate correlation.

Questionnaires developed in survey research or patient-reported outcomes (PROs) often include questions with a mixture of response option styles. Sometimes the response (i.e., the variable) is easily captured on a continuous scale, such as measurements of height and weight, or as a summary score from a well-developed instrument composed of a battery of related items. Sometimes the focus is on single items and a categorical representation is more appropriate. Often, categorical items form ordinal variables, where the observed levels can be naturally ordered. For example, a Likert-type item that measures the extent to which a respondent agrees with a statement might measure agreement on a 5-point response scale with ordinal categories ranging from “1, strongly disagree” to “5, strongly agree.”

These Likert-type single items are often used in pharmaceutical- and biotech-based studies that assess PROs. The outcomes inform researchers of the impact of diseases on patients, and the drug benefits that are a high priority for patients. To assess construct validity (i.e., the ability of an instrument to measure what it is purported to measure) correlation coefficients are often computed between ordinal measures and continuous scales. Because the magnitude and sign of the correlation coefficients can be crucial to supporting the psychometric validity of the instrument, it is preferable to minimize bias by choosing the correlation computation method most appropriate for the variable types.

Given multiple variable types in PRO studies, you must be aware of the correlations chosen for analysis. For example, the popular Pearson product-moment correlation is not advised for use with ordinal variables because the resulting correlation is often artificially deflated (Nunnally & Bernstein, 1994). When the observed variables are assumed to be ordinal representations of an unobservable/latent continuous variable, which is often the case with PRO measures, then two ordinal variables would best be analyzed with polychoric correlations. Likewise, the correlation that best suits one ordinal variable and one continuous variable is a polyserial correlation. Pearson correlations are most appropriate for two normally-distributed continuous variables.

Currently, producing tables that contain Pearson, polychoric, and polyserial correlations is challenging in Base SAS. Although a matrix of Pearson correlations is easy to obtain, such a matrix of polychoric or polyserial correlations is not as easily obtained via SAS procedures. There are no SAS procedures specifically designed to create a polychoric correlation matrix, and there are no SAS procedures specifically designed to calculate a polyserial correlation or correlation matrix. SAS users and SAS staff have, however, designed "workarounds," such as the %POLYCHOR macro, documented on the SAS Support Web site (SAS Knowledge Base/Samples & SAS Notes: Sample 25010, 2009). In addition, a user-created macro, the %POLYSERIAL macro produces pairwise polyserial correlations (Yang, 2009). To complement these tools, a one-stop macro for producing a correlation table with various correlations selected based on the variable type is desirable for any researcher in the pharmaceutical or biotech industry who evaluates PRO measures. Further, these researchers often desire an uncluttered table of correlations instead of a correlation matrix because they are sometimes only interested in the correlations between already-validated items/scales and newly developed items/scales.

This paper introduces a SAS macro called %PROCORR that routes variables to the appropriate correlation computation given the variable type. The macro also identifies the appropriate output statistics for each type of correlation and creates a report-ready table. The aptly named "%PROCORR" macro was originally developed to analyze PRO outcomes. Therefore, while walking through the details of %PROCORR, we illustrated use of the macro in a fictitious PRO study.

THE DATA LANDSCAPE

Many psychometric analyses involving PRO data use multiple variable types. This paper focuses on two of the most widely used variable types in PROs: continuous variables and ordinal variables. PRO studies tend to yield more continuous and ordinal data because patients are asked questions in a survey format and are able to categorize their responses in these two forms with little burden. The variable types are important in defining which correlations are the most appropriate for analyses.

Nunnally and Bernstein (1994) have suggested that variables with 11 or more levels be considered continuous and those with 10 or less levels should be considered discrete. Although this rule is helpful for general guidance on choosing a variable type, it is the researcher's responsibility to consider distributions to determine data characteristics. It is entirely possible that the actual data values do not match the intended data type. For example, an analyst might consider a variable to be continuous when it actually reports a skewed cell count.

This paper describes a macro that directs the analyses toward the following three types, based on variable type:

1. ORDINAL VS. ORDINAL

When two variables are ordinal but assumed to have an underlying continuous nature, polychoric correlations are appropriate methods for analyses. Polychoric correlations are preferred because they estimate the correlation coefficient as if the variables had been measured on a continuous scale.

2. CONTINUOUS VS. ORDINAL

Polyserial correlations are appropriate for correlations between continuous and ordinal variables, when it is assumed that the ordinal variable has an underlying continuous dimension. Similar to polychoric correlations, polyserial correlations are preferred because they estimate the correlation coefficient as if the ordinal variable had been measured on a continuous scale.

3. CONTINUOUS VS. CONTINUOUS

The relationship between two continuous (and linear) variables is often described using Pearson product-moment correlations. Pearson product-moment correlations are inappropriate for categorical data because the values do not vary as expected on the usual -1 to +1 scale and are often underestimates of the true latent correlation (Muthén, 1983).

The guidelines above are only one aspect of the analysis process. Prior to choosing a correlation type, you should consider the following universal aspects of the data (Nunnally & Bernstein, 1994):

1. Is the variable a derived ordinal variable? If so, it is possible to incorrectly estimate *r* because categorization reduces *r*.
2. Is the latent variable supposed to be a continuous variable? If the underlying nature of a variable in a correlation is not continuous, correlations are not appropriate.
3. Is the total sample size sufficient? Without a proper sample size, there may not be enough power to produce unbiased coefficients.

Ideally, the %PROCORR macro would pull in the analysis variables and do all the work. Given the list above, it is clear that blindly running correlations is not advised and it is up to you to examine the context of each correlation.

In addition to the universal aspects of the data, univariate and bivariate considerations must also be reviewed prior to the analysis, including:

1. Low cell counts (i.e., few responses in one ordinal category can bias correlations).
2. The structure of continuous distributions
3. Sample sizes available for each bivariate analysis. The overall sample size may be large. But, respondents may have chosen to skip questions, leading to low n's for bivariate associations.

Although the %PROCORR macro could automatically detect the variable type (continuous vs. ordinal) based on the data values available in the analysis data set and thus reduce the required work by the analyst, it is possible that the situations listed above could result in SAS erroneously identifying a set of values as the wrong variable type. Therefore, the %PROCORR macro requires that each variable type be explicitly stated so that it can provide the most accurate results. You are the driver and Base SAS is the vehicle.

PRO EXAMPLE

Asthma is a chronic inflammatory lung disease that affects both children and adults. It remains a disease that is inadequately controlled, leading to lost productivity and sometimes traumatic asthma attacks. Medical professionals and researchers wish to emphasize monitoring asthma control by both physicians and patients to decrease the incidence of serious asthma attacks. Therefore, a short questionnaire has been developed to facilitate the assessment of the level of control of asthma on a regular basis. The questionnaire is self-administered by patients and only takes a few minutes to complete. Researchers wish to validate this questionnaire so that it can be used to accurately measure asthma control with little burden on the patient and, therefore, help prevent serious attacks.

The questionnaire was tested in a sample of 100 patients with asthma symptoms. The sample data set consists of 100 records, an identification (ID) variable, and eight analysis variables. The ID variable represents patients in a PRO study validating an asthma scale. The data set consists of one record per person.

The analysis variables are continuous and ordinal measures of asthma symptoms and severity, including three novel test survey instrument items and four validated scales/subscales.

The test survey questions are listed below. Patients were asked the following three questions about their asthma:

1. How often do you use extra puffs of your inhaler?
   1 = Never
   2 = Once a week
   3 = More than once a week
   4 = Once a day
   5 = More than once a day
2. How often do you experience breathing problems (such as coughing and/or wheezing) due to your asthma?
   1 = Never
   2 = Once a week
   3 = More than once a week
   4 = Daily
   5 = Continuous
3. How often does physical activity affect your breathing?
   1= Never
   2 = Rarely
   3 = Sometimes
   4 = Usually
   5 = Often

A total asthma score was calculated by summing the three test questions above.

Quantitative psychologists wish to validate these test items against four already-validated asthma measures, three subscales from the St. George's Respiratory Questionnaire (SGRQ) and the Clinician Global Impression of Severity (CGI-S). The SGRQ consists of three continuous subscales with values ranging from 1 to 100: 1) symptoms (frequency and severity), 2) activity (activities that cause or are limited by breathlessness), and 3) impacts (social functioning and psychological disturbances resulting from airway disease). The CGI-S is an ordinal measure of a patient's current asthma severity, in which clinicians are asked to rate patients on a scale from 1 to 7 (1 = "not at all ill", 7 = "extremely ill).

The data set is called DER.Asthma and is found in the "ANALYSIS" library. (A requirement of the %PROCORR macro is that the analysis data set is permanent.) The first four records and the last record of ANALYSIS.Asthma are displayed in Table 1. Q1, Q2 and Q3 are the variable names for the asthma test questions, and AsthmaTotal is the sum of these test questions. The last four variables contain the CGI-S and the SGRQ responses.

Table 1. Sample Records from ANALYSIS.Asthma

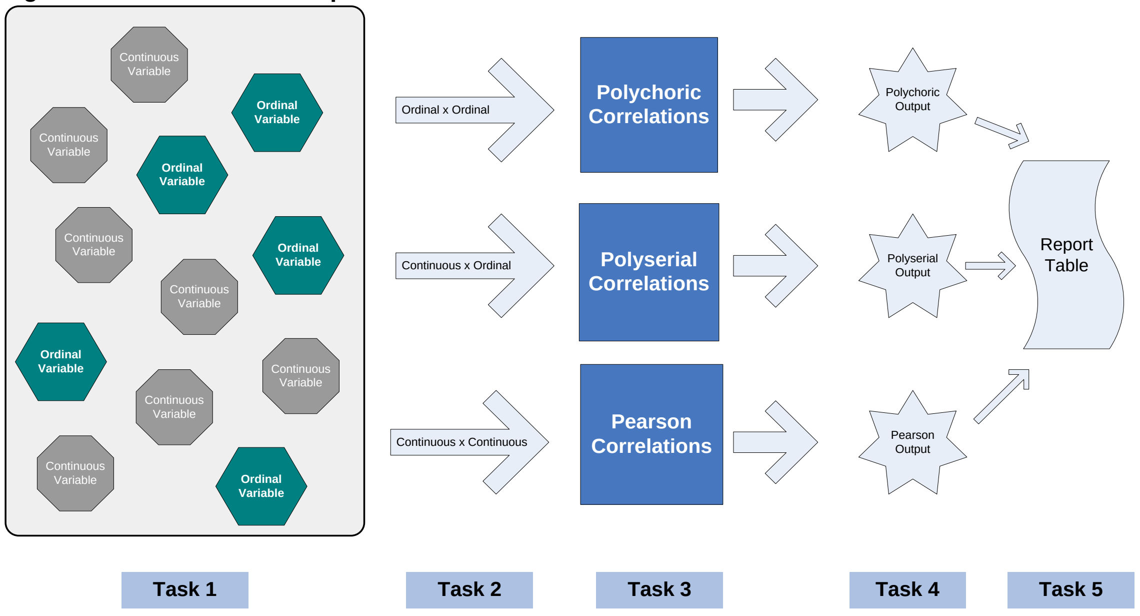
PatientID	Q1	Q2	Q3	AsthmaTotal	CGIS	SGRQSymptoms	SGRQActivity	SGRQImpacts
1	4	2	2	8	1	13	5	32
2	2	2	2	6	3	21	30	39
3	2	2	2	6	1	51	6	19
4	2	2	2	6	1	35	10	3
↓	↓	↓	↓	↓	↓	↓	↓	↓
100	3	2	2	7	2	39	14	43

One of the first steps in validating test items/scales is to correlate all test items/scales with validated scales. Given that the sample data set is composed of both ordinal and continuous variables, the %PROCORR macro is chosen as the analysis tool for the correlation analyses.

THE ROADMAP

Figure 1 illustrates the tasks that the macro accomplishes from starting with raw data to creating a report table.

Figure 1. Illustrated Road Map



TASK 1: PACKING THE MACRO SUITCASE (NAMING PARAMETERS)

As illustrated in the gray box in Figure 1, you start with a permanent SAS data set, which includes multiple variables of various types. Task 1 requires you to identify the variables for analysis and their respective variable types. The %PROCORR macro consists of nine parameters, but the analyst is required to provide only six parameters in the macro call (i.e., numbers 7 through 9 are optional):

%PROCORR Parameters

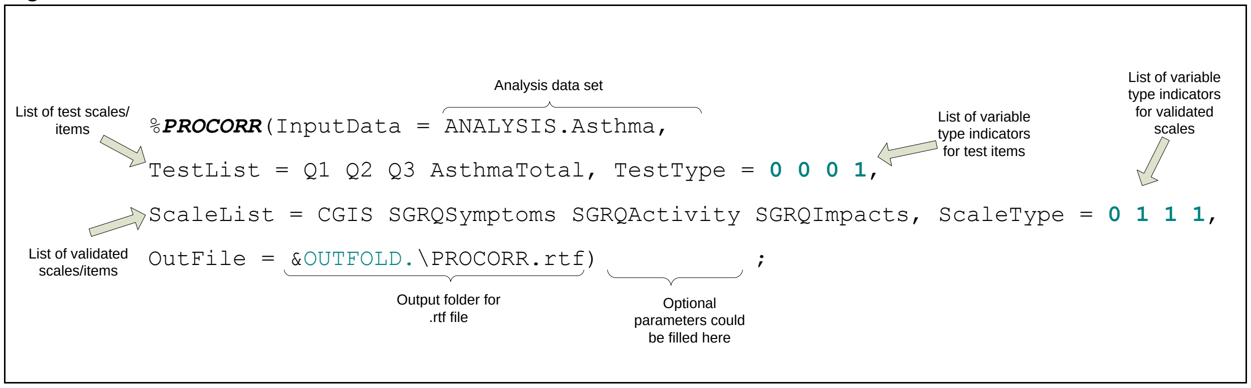
1. InputData – permanent SAS data set to be used for analysis (cannot be located in the WORK directory)
2. TestList - space-delimited list of items or scales being validated
3. TestType - space-delimited binary indicator of TestList variable-type, 0 = ordinal, 1 = continuous
4. ScaleList - space-delimited list of items or scales used for validation
5. ScaleType - space-delimited binary indicator of ScaleList variable-type, 0 = ordinal, 1 = continuous
6. OutFile - path to the location (and file name) in which the output .rtf file should be stored. A global variable (discussed later) fills the path location. The analyst needs to fill the file name.
7. Subset - subset specifications for analysis data set. The code entered in &Subset is entered into WHERE statements throughout %PROCORR. Therefore, it is required that you use the %STR macro function to mask any comparisons or logical operators (i.e., =, <, >, etc.) If no subsetting required, can be left blank.
8. Nval – critical N value, If N < &nval then correlation is flagged with ^ in the final table. The default n is 10.
9. Pval- p-value , If p < &pval then correlation is flagged with * in the final table. The default p-value significance cut-off is .05.

The %PROCORR is a keyword macro program. The %macro statement lists the nine parameters as illustrated below.

```
%macro PROCORR(InputData=, TestList=, TestType=, ScaleList=, ScaleType=,
OutFile= , Subset= , nval = 10, pval = .05);
```

In Figure 2, the macro call was written to analyze the test and validated scale variables for ANALYSIS.Asthma.

Figure 2. Macro Call



```
%PROCORR(InputData = ANALYSIS.Asthma,
TestList = Q1 Q2 Q3 AsthmaTotal, TestType = 0 0 0 1,
ScaleList = CGIS SGRQSymptoms SGRQActivity SGRQImpacts, ScaleType = 0 1 1 1,
OutFile = &OUTFOLD.\PROCORR.rtf)          ;
```

Analysis data set
List of test scales/ items
List of variable type indicators for test items
List of variable type indicators for validated scales
List of validated scales/items
Output folder for .rtf file
Optional parameters could be filled here

Q1, Q2, Q3 and AsthmaTotal are listed in &TestList and will be correlated with CGIS, SGRQSymptoms, SGRQActivity, and SGRQImpacts in &ScaleListbased on their corresponding variable types in &TestType and &ScaleType,. The %PROCORR macro identifies Q1 as an ordinal variable because the first value in "TestType" parameter is a "0."

In addition to the macro parameters discussed, %PROCORR utilizes two user-defined global macro variables to identify the analysis data set library and the .RTF output location. &DATFOLD allows you to tell SAS the location of the analysis data set. The contents of &DATAFOLD are used to create a generic library called "ANALYSIS." &OUTFOLD is the location of the final output .RTF file, which is included in the OutFile parameter of %PROCORR. Sections of the code related to these user-defined global macro variables are presented below.

```
        /* Identify folder location for data and output table */
        %let DATFOLD = C:\Documents and Settings;
        %let OUTFOLD = C:\Documents and Settings\Output;
        LIBNAME ANALYSIS "&DATFOLD";
```

TASK 2: CROSSROADS (VARIABLE × VARIABLE)

In Task 2, the macro decides which correlation to use. It then routes variables to the appropriate code for computing either a polyserial, polychoric, or Pearson correlation coefficient.

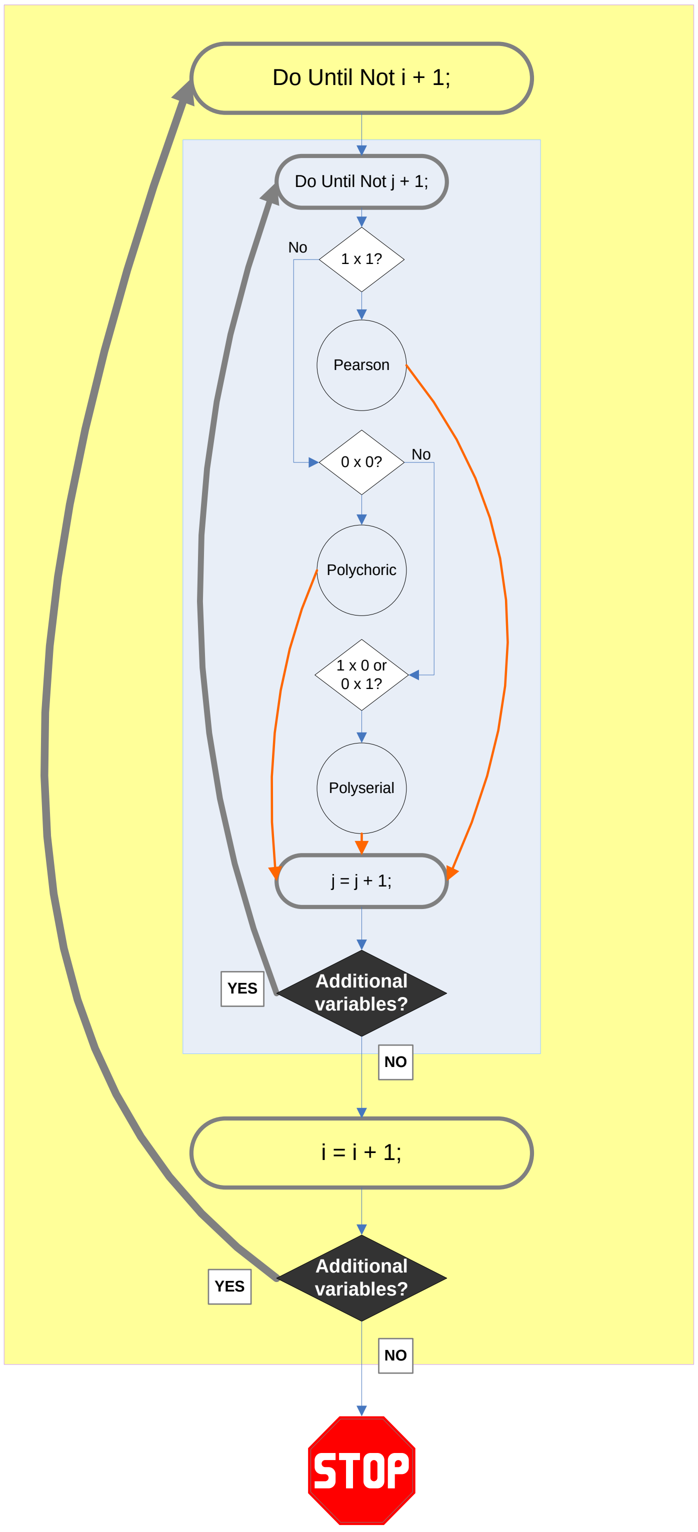
The %PROCORR macro comprises three main sections, one for each correlation type. %PROCORR routes variables to the appropriate section based on the variable type indicators in the "TestType" and "ScaleType" parameters.

In the asthma example, the list of test items includes Q1, Q2, Q3, and AsthmaTotal, and the validated scale list includes CGIS, SGRQSymptoms, SGRQActivity, and SGRQImpacts. The quantitative psychologists in the study would like to correlate each item in the test item list with each variable in the validated scale list. It is not necessary to correlate the variables within each list between each other. Therefore, two do loops work together to ensure that each variable in the "TestList" and "ScaleList" parameters are matched and circulated for analysis. The macro starts with a loop circulating through i-variables in "TestList." Within each i-loop, the macro loops through j-variables in "Scale List." The i-loops represent the test items (test loop) and the j-loops represent the validated scales (scale loop).

For example, in the first i-loop, the %PROCORR macro starts with Q1, matches it with CGIS in the first j-loop and runs the appropriate calculations. Next, it matches Q1 with the second variable in the scale list, SGRQSymptoms (i.e., second j-loop) and runs the appropriate calculations. In the third j-loop, it matches Q1 with SGRQActivity. After all the variables in loop j (the scale loop) have been matched, the i–loop (test loop) increments to the second variable (i.e., Q2) in the test item list.

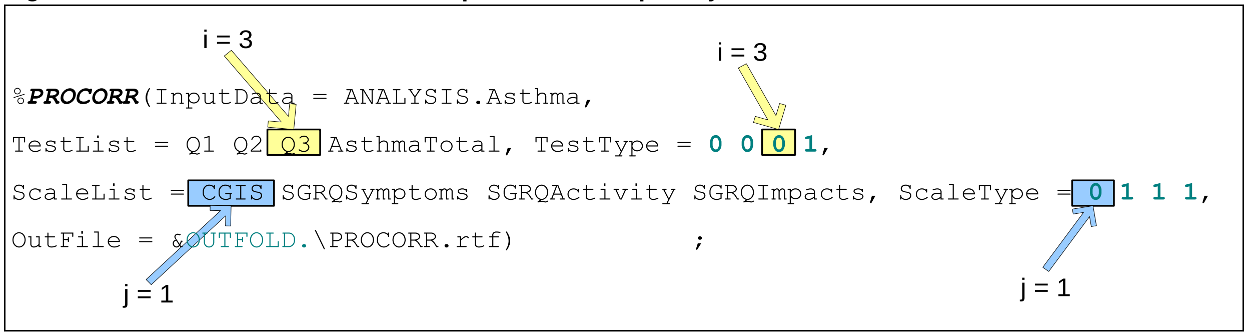
The thick grey lines in Figure 3 illustrate the looping process for the test and validated item parameters. The circles represent the code required to compute each correlation. The white diamonds symbolize the decision process that %PROCORR uses to route to and around the correlation code. After a correlation has been computed (characterized by the orange lines), an output data set is created and the loop begins again. Black diamonds represent the decision process that %PROCORR employs to determine whether there are any additional variables in &TestList and &ScaleList.

Figure 3. Looping Process for the Test and Validated Item Parameters



As each variable pair from ANALYSIS.Asthma circulates through the do loops, the %PROCORR macro decides which correlation is most appropriate. Conditional macro statements (illustrated as white diamonds in Figure 3) route variables to—or around—each correlation section in the macro. As i and j increment, %PROCORR moves from one variable to the next, and from one variable type indicator to the next. As illustrated in Figure 4, while %PROCORR is on the third loop of i (i.e., Q3) and the first loop of j (i.e., CGIS), it also locates the analogous location in &TestType and &ScaleType.

Figure 4. %PROCORR Macro Call at Loop 3 of i and Loop 1 of j



```
%PROCORR(InputData = ANALYSIS.Asthma,
TestList = Q1 Q2 Q3 AsthmaTotal, TestType = 0 0 0 1,
ScaleList = CGIS SGRQSymptoms SGRQActivity SGRQImpacts, ScaleType = 0 1 1 1,
OutFile = &OUTFOLD.\PROCORR.rtf)          ;
```

The syntax to identify variables in each loop utilizes a handful of macro functions including %Let statement, %Length, and %Scan. The first %Let statement identifies a starting value for the position in the variable and type lists. A value other than 1 would not be recommended because the macro would not be directed to pull all of the variables/types in the lists by starting with position 1. The next few functions (%Scan and %Length) are used to identify the i-th and the j-th positions in the type and test/validated scale lists. %SCAN can identify words depending upon their position in a string. In this circumstance, %SCAN only requires two pieces of information: 1) the character string to search (&TestList or &ScaleList) and 2) the position of the word to return (&i or &j). After identifying the position of the variable in the &TestList or &ScaleList lists, %LENGTH returns the length of the string (i.e., the number of variables in each list) and signals to the %DO %UNTIL statement when the list is complete. %DO %UNTIL tells the macro to continue repetitively until a condition is true. The NOT operator, in conjunction with the %DO %UNTIL statement continues to iterate until the length of the text pulled from the %SCAN macro function is 0, indicating the end of the test or scale list. Prior to each loop closing, i and j are incremented by 1 and then the process starts again at the top of the loop with the i-th + 1, and j-th + 1 position.

```
%Let i = 1; /* Loop over test items */
%Do %Until (NOT %LENGTH(%SCAN(&TestList, &i)));

       %Let j = 1; /* Loop over scale items */
       %Do %Until (NOT %LENGTH(%SCAN(&ScaleList, &j)));
```

Correlations occur here…

```
       /*** Close Loops ***/
       %let j = %EVAL(&j + 1);
       %End;  /* End loop over scale items */

%let i = %EVAL(&i + 1);
%End;  /* End loop over test items */
```

As noted above, the diamonds in Figure 3 represent the conditional logic to route to ---or around---each correlation section. The %SCAN function identifies the i-th and j-th variable pair types. If the variables pulled from &TestList and &ScaleList are both continuous, %PROCORR computes Pearson correlations by routing to the Pearson section. An example of this routing code is listed below for the Pearson section:

```
%if %SCAN(&TestType, &i) = 1 and %SCAN(&ScaleType, &j) = 1 %then %do;

/* Pearson section code */

%end;
```

Analogous code is again used to wrap both the Polychoric and Polyserial sections of the code:

```
%if %SCAN(&TestType, &i) = 0 and %SCAN(&ScaleType, &j) = 0 %then %do;

/* Polychoric section code */

%end;


%if (%SCAN(&TestType, &i) = 1 and %SCAN(&ScaleType, &j) = 0) OR
(%SCAN(&TestType, &i) = 0 and %SCAN(&ScaleType, &j) = 1)%then %do;

/* Polyserial section code */

%end;
```

The %SCAN function is used to identify variable names/types for the correlation code procedures and DATA steps. To simplify the code producing correlations, the identification function was replaced with a macro variable called &TEST or &SCALE to represent the test or scale variable names respectively.

```
%let test = %SCAN(&TestList, &i);
%let scale = %SCAN(&ScaleList, &j);
```

TASK 3: THE ROAD TRIP

Task 3 is when Base SAS does all the hard work. In this task, computations are completed and the macro runs through loops to pull all the variables. The final goal of this task is to create a unique output data set for each correlation run, which includes the test and validated item/scale names, a correlation coefficient, p-value, and sample n for the correlation. Most importantly, each correlation data set includes a character variable, which includes a formatted version of the correlation coefficient (2 decimal places), a significant p-value indicator, and a low-n indicator that appears in the final data set.

Pearson Correlations

The Pearson section uses a SAS procedure, PROC CORR, for analysis and selects necessary output for the final report table. A segment of the Pearson section is illustrated below. As mentioned in Task 2, the &TEST and &SCALE took the place of the longer `%SCAN(&TestList, &i)` and `%SCAN(&ScaleList, &j)` functions, respectively. &SUBSET is the optional %PROCORR parameter, and simply subsets the analysis based on user-defined statements.

```
title2 "Pearson Output: Validated Scale = &Scale, Test Item = &Test";
ods output PearsonCorr = PearOut;
ods output FisherPearsonCorr = CorrN; /* Workaround to pull n's for pearson
correlations: http://support.sas.com/kb/14/469.html */
proc corr data=&InputData PEARSON FISHER nosimple;
       where &SubSet;
       var &scale;
       with &test;
run;
```

After running the Pearson correlation, a series of simple DATA steps pull only the necessary information and create a standard data set that will later be used to merge and append to other correlation outputs. Each correlation calculation produces a temporary data set made up of one record with five variables. The example data set shown in Table 2 is output generated from the 4th i loop and 2nd j loop, which crosses the AsthmaTotal (continuous test scale) and the SGRQSymptoms (continuous validated scale), producing a Pearson correlation coefficient. ITEM holds the name of the test scale variable. The next four variable names depend upon the name of the validated scale. Each data set contains one variable for the correlation value, which is given the name of the validated scale variable; in this case, “SGRQSymptoms.” The variables that contain the n and p-value are prefixed with “N” and “P,” respectively, followed by the validated scale variable name, which differentiates these variables from other output variables (i.e., NFinalSGRQSymptoms and PFinalSGRQSymptoms). The last variable, “FinalSGRQSymptoms” displays the formatted correlation (two decimal points) to be printed in the report-ready table. An asterisk (*) accompanies any significant p-values (default is alpha = 0.05) and a carrot (^) that accompanies any correlation n’s below a specified number (default is 10).

Table 2. Example Data Set

Item	**SGRQSymptoms**	**PSGRQSymptoms**	**NSGRQSymptoms**	**FinalSGRQSymptoms**
ASTHMATOTAL	0.654420203	0.041202037	100	0.65*

Polychoric Correlations

Much like the Pearson section, the polychoric section of the macro is built from a key SAS procedure, PROC FREQ to create the necessary output for the final report table. A section of the polychoric code is displayed below.

```
title2 "PolyChor Output: Validated Scale = &Scale, Test Item = &Test";
proc freq data=&InputData;
  where &SubSet;
  tables &test * &scale / plcorr maxiter=1000;
  output out = FreqOut PLCORR N;
run;
```

The output data set is formatted exactly as the Pearson data set example above, but contains the correlation, p-value, n, and variable names for only the appropriate variable pairs.

Polyserial Correlations

%POLYSERIAL_OUTPUT is based on a macro originally written by Yung-Jui Yang that incorporates PROC IML

(Yang, 2009) and modified by Cheryl Coon to save the polyserial correlation (as calculated by Olsson's ML algorithm), the p-value, and the n to a data set named WORK.Polyserial (Olsson, 1982). The call for %POLYSERIAL_OUTPUT looks like this:

```
%polyserial_output(data= ,co= ,or= );
```

The first parameter, "DATA," is filled with the analysis data set name. "CO" is filled with a continuous variable name, and "OR" is filled with an ordinal variable name. The %PROCORR macro circulates the appropriate variables through %POLYSERIAL_OUTPUT by inserting the &TEST and &SCALE macro variables into the "CO" and "OR" parameters.

A similar series of simple DATA steps are used to create a data set that is just like the Pearson example above (automatically filling in the appropriate variable names) so it can later be merged and appended with other analogous temporary data sets. %Polyserial is available online (see references) and the modifications used to create %POLYSERIAL_OUTPUT are located in the appendix.

TASK 4: ARE WE THERE YET?

Each iteration of the i- and j- do-loops identifies a pair of variables for analysis. As mentioned in Task 3, the output data sets are uniform and produce five variables, including the text version of the test item name, three numeric variables holding the correlation coefficient, the p-value, and n, as well as a character variable intended for printing in the output table that displays the formatted correlation-coefficient and low n and p-value indicators. The individual correlation output data sets are saved in the work library and have uniform names so that they are easily identifiable. The data set names begin with the correlation type (Pearson, Polyserial, or Polychor), an underscore, the i-th loop number, an underscore and the j-th loop number. For example, the name of the Pearson correlation data set example in Task 3 is Pearson_4_2 because we pulled the 4th i-loop test item and the 2nd j-loop validated scale.

Each of these data sets is merged together by the variable "ITEM." This merge produces a data set that holds one test item variable (rows) crossed with the validated scale (columns) and the formatted correlation coefficients. This data set is called CORR and is found in the WORK library.

If you identified variable labels for the analysis variables, %PROCORR replaces the variable names in WORK.CORR with those labels by pulling this information from PROC CONTENTS. If you did not identify labels, the variable names continue to be displayed as row and column headers. Variable type identifiers (1 = continuous, 0 = ordinal) are appended to the end of the variable names/labels so that analysts can determine which correlation type was used produce each correlation coefficient. This data set is called CORRSLABELED and is also found in the WORK library.

The code used to label row and column headers is provided in the appendix with detailed comments. However, it is out of the scope of this paper to walk through the code in detail.

TASK 5: WHO'S GOT THE CAMERA? (SNAPSHOTS OF THE DATA)

Using ODS OUTPUT and PROC PRINT, %PROCORR outputs WORK.CORRSLABELED into a .rtf file name and in a location specified in the "OutFile" parameter and "OUTFOLD" global macro variable. The code to create the table is as follows:

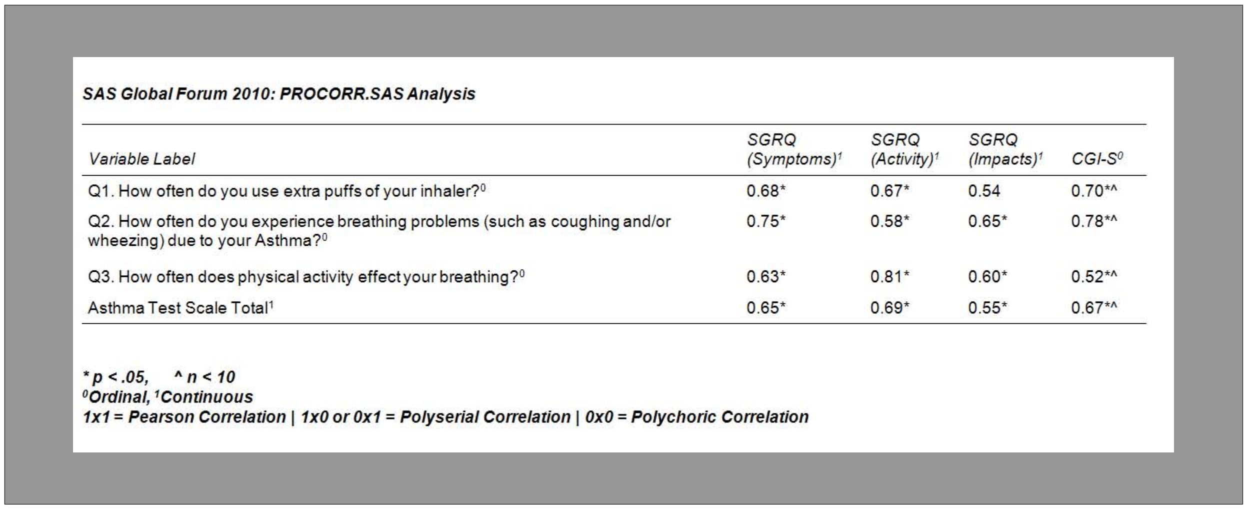
```
footnote1 "* p < &pval,   ^ n < &nval";
ods escapechar='~';
footnote2 "~{super 0}Ordinal, ~{super 1}Continuous";
footnote3 "1x1 = Pearson Correlation | 1x0 or 0x1 = Polyserial Correlation |
0x0 = Polychoric Correlation";
options label;

ods rtf file="&OutFile" style=styles.journal bodytitle;
proc print data = CorrsLabeled noobs label;
       VAR label FINAL: ;
run;
ods rtf close;
```

ODS escapechar, in conjunction with the "~{super 0}" and "~{super 1}" text in the footnote, create superscripts to identify the variable type as specified by the user in &TestType and &ScaleType.

The table is composed of one column and one row per variable. The output from the PRO asthma example is displayed in a table presented in Figure 5. The final table displays all relevant correlation coefficients, indicators for low correlation n's, significant p-values, variable type, and correlation type. The significant p-value and n cut-offs are user-defined in global macro variables, with defaults .05 and 10, respectively.

Figure 5. RTF File Created With ODS Output



SAS Global Forum 2010: PROCORR.SAS Analysis

Variable Label	*SGRQ (Symptoms)*^{1}	*SGRQ (Activity)*^{1}	*SGRQ (Impacts)*^{1}	*CGI-S*^{0}
Q1. How often do you use extra puffs of your inhaler?^{0}	0.68*	0.67*	0.54	0.70*^
Q2. How often do you experience breathing problems (such as coughing and/or wheezing) due to your Asthma?^{0}	0.75*	0.58*	0.65*	0.78*^
Q3. How often does physical activity effect your breathing?^{0}	0.63*	0.81*	0.60*	0.52*^
Asthma Test Scale Total^1	0.65*	0.69*	0.55*	0.67*^

p < .05, ^n < 10
^{0}Ordinal, ^{1}Continuous
1x1 = Pearson Correlation | 1x0 or 0x1 = Polyserial Correlation | 0x0 = Polychoric Correlation

Analysts can now review all three correlation-types in one table. Correlation type can be deduced by crossing the variable type, represented as a superscript, in the row and column headers. For example, readers would identify the first row variable (Q1) as ordinal because it contains a superscript of “0”. Q1 was correlated with SGRQ (Symptoms), a continuous variable, which is characterized by a superscript of “1” to produce a correlation of 0.68. Readers would simply review the footnote to determine the type of correlation that was produced. In this case, an ordinal and a continuous were correlated, producing a polyserial correlation coefficient.

Given the relatively high correlations produced between the test Asthma questions and the validated scales, the researchers are pleased to conclude that the new scale is performing quite well. High correlations were observed between pairs that assess similar constructs. For example, crossing the test asthma question about breathing problems (Q2) and the validated scale assessing symptoms (SGRQ Symptoms) resulted in a significant correlation of 0.75. Significant values were observed across the board for most correlation pairs (noted with superscript stars). The table also indicates a low n in the correlations involving the validated CGI-S scale (noted with superscript carrots). The CGI-S was printed on the back of a page and many of the physicians did not realize that they had not completed the questionnaire, resulting in many missing values for this item.

A VACATION FROM THE VACATION: VARIATIONS AND NEXT STEPS

%PROCORR is currently limited to only Pearson, polychoric, and polyserial correlations, but additional correlation types could be added by simply adding variable type indicators and conditional macro statements to route variable pairs to the appropriate correlation section. The authors are in the process of creating additional code to calculate Spearman correlations appropriate for use with count data.

As noted at the beginning of the paper in the “Data Landscape” section, knowing your data is extremely important. %PROCORR outputs rudimentary plots from PROC UNIVARIATE prior to running correlations. However, the authors usually review additional information prior to running correlation analyses. The authors are in the process of creating additional code to output a preliminary .rtf file, which includes information on bivariate and univariate cell counts and more aesthetic graphical displays of continuous distributions.

One of %PROCORR’s strengths is the ability to output only the correlation coefficients that you want to view. The asthma example provides an illustration of creating output with a subset of the correlation coefficients. However, the %PROCORR macro can be used more universally to create an inter-item correlation matrix if an identical list of variables is listed in &TestList and &ScaleList.

CONCLUSION

The %PROCORR macro can be a valuable tool if you often compute correlations among a large number of ordinal and continuous variables and have need for compiling these correlations in a report-ready table. %PROCORR is currently designed to package three correlation types into one routine to eliminate your need for three separate correlation programs (sometimes, run several times in the case of polyserials). Additionally, %PROCORR could be enhanced by safeguarding against violations of correlation assumptions, such as minimal sample size. The routing technique described can be expanded to include additional correlations or analysis applications.

REFERENCES

Muthén B. 1983. “Latent Variable Structural Equation Modeling With Categorical Data.” Journal of Econometrics 22:43-65.

Nunnally J and Bernstein I. 1994. Psychometric Theory, Third Edition. New York, NY: McGraw-Hill, Inc.

Olsson U, Drasgow F, Dorans N. 1982. "Polyserial Correlation Coefficient." Psychometrika 47:337-47.

Yang YJ. “SAS Macros.” <http://sites.google.com/site/yangyungjui/academic_home/statistics/sas-macros> (Oct. 23, 2009).

RECOMMENDED READING

Garson D. Correlation. 2008. Available at: http://faculty.chass.ncsu.edu/garson/PA765/correl.htm. Accessed February 22, 2010.

Jöreskog, K. G. 1990. “New Developments in LISREL: Analysis of Ordinal Variables Using Polychoric Correlations and Weighted Least Squares.” *Quality and Quantity* 24:387-404.

Mislevy, R. J. 1986. “Recent Developments in the Factor Analysis of Categorical Variables.” *Journal of Educational Statistics* 11:3-31.

SAS Institute Inc. 2007. SAS Note 25010. "Create a Polychoric Correlation or Distance Matrix." Available at www.support.sas.com.

SAS Institute Inc. 2009. SAS Problem Note 14469. “Variables reporting N in the ODS OUTPUT data sets are not present if no variable used to calculate the correlations contains a missing value.” Available at http://support.sas.com/kb/14/469.html

Uebersax J. Introduction to the Tetrachoric and Polychoric Correlation Coefficients, 2008. Available at: http://www.john-uebersax.com/stat/tetra.htm. Accessed February 22, 2010.

CONTACT INFORMATION

Your comments and questions are valued and encouraged.

Contact the author at:

Theresa Gilligan, MS
RTI Health Solutions
200 Park Office Drive
PO Box 12194
Work Phone: 919.316.3843
Fax: 919.541.7222
E-mail: gilligan@rti.org
Web: www.rtihs.org

APPENDIX

```
*******************************************************************************************;
* Program: PROCORR.sas
* Authors:Theresa Gilligan, Cheryl Coon, Lauren Nelson, Lori McLeod, Yung-Jui Yang
* Purpose: Calculates three types of correlations for Item-level Validity Correlations
* Macros called:POLYSERIAL_OUTPUT
* SAS version: 9.2
*******************************************************************************************;

options nodate nonumber nocenter nofmterr mprint mlogic symbolgen label
               topmargin=1in bottommargin=1in leftmargin=1.25in rightmargin=1.25in;

title1 "SAS Global Forum 2010: PROCORR.SAS Analysis";
footnote "Date: &SYSDAY &SYSDATE  |  SAS Version: &SYSVER";

/************************************************************/
/*                    USER INPUT START                      */
/************************************************************/
/* Identify folder location for data and output table */
%let DATFOLD = C:\Documents and Settings;
%let OUTFOLD = C:\Documents and Settings\Output;
/************************************************************/
/*                    USER INPUT END                        */
/************************************************************/

LIBNAME ANALYSIS "&DATFOLD";

/* Include Polyserial Macro */
/* Modified from http://sites.google.com/site/yangyungjui/academic_home/statistics/sas-macros */
/* Version: 0.2 2006.6.14 by Yung-jui Yang */
%include "C:\Documents and Settings\macros\polyserial_output.sas";

%macro PROCORR(InputData=, TestList=, TestType=, ScaleList=, ScaleType=, OutFile= , Subset= ,
nval = 10, pval = .05);

proc datasets library=work kill;
run;
quit;

/* Print univariate descriptive statistics to allow checking of frequencies, distributions, and
sample sizes */
title2 "Descriptives";
proc univariate data = &InputData plot;
       where &SubSet;
       var &TestList &ScaleList;
run;
title2;

/*** Start Loops ***/
%Let i = 1; /* Loop over test items */
%Do %Until  (NOT %LENGTH(%SCAN(&TestList, &i)));

       /* Pull the TEST type into dataset TYPE */
       data temp;
              length var $32. type $1.;
              var="%SCAN(&TestList, &i)";
              type="%SCAN(&TestType, &i)";
              ORDER = &i;
       run;

       proc append base=type data=temp;
       run;

       %Let j = 1; /* Loop over scale items */
       %Do %Until  (NOT %LENGTH(%SCAN(&ScaleList, &j)));

              /* Pull SCALE type into a dataset TYPE (only needed for 1st loop over j) */
              %if &i=1 %then %do;
                     data temp;
                            length var $32. type $1.;
```

```
                var="%SCAN(&ScaleList, &j)";
                type="%SCAN(&ScaleType, &j)";
                ORDER = .;
        run;

        proc append base=type data=temp;
        run;
%end;

/*********************************************************************/
/*** Pearson Correlations (Continuous vs Continuous) ***/
/*********************************************************************/
%if %SCAN(&TestType, &i) = 1 and %SCAN(&ScaleType, &j) = 1 %then %do;
        %let test=%SCAN(&TestList, &i);
        %let scale=%SCAN(&ScaleList, &j);
        title2 "Pearson Output: Validated Scale = &Scale, Test Item = &Test";
        ods output PearsonCorr = PearOut;
        /* Workaround to pull n's for Pearson corrs: SAS Problem Note 25010 */
        ods output FisherPearsonCorr = CorrN;
        proc corr data=&InputData PEARSON FISHER nosimple;
                where &SubSet;
                var &scale;
                with &test;
        run;

        /* Pull n's from here because ODS output does not output when = 100% */
        data CorrNOnly;
                set CorrN (KEEP = NOBS);
                rename NOBS = N&scale;
        run;

        data Pearson_&i._&j;
                length Item $50.;
                merge PearOut (keep = Variable &scale p&scale) CorrNOnly;
                /* Trim p-values to 2-decimal places, identify low n counts and
                significant p-values */
                if P&scale < &PVAL and N&scale < &NVAL then Final&scale =
                compress(put(&scale,7.2))||'*^';
                        else if P&scale < &PVAL and N&scale >= &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'*';
                        else if P&scale >= &PVAL and N&scale < &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'^';
                        else Final&scale = compress(put(&scale,7.2));
                Item = strip(upcase(Variable));
                Label
                        Item    = "Item"
                        N&scale = "N &scale x &test";
                drop Variable;
        run;

        proc datasets;
                delete PearOut CorrN CorrNOnly;
        run;
        quit;
%end;

/*************************************************************/
/*** Polychoric Correlations (Ordinal vs Ordinal) ***/
/*************************************************************/
%if %SCAN(&TestType, &i) = 0 and %SCAN(&ScaleType, &j) = 0 %then %do;
        %let test=%SCAN(&TestList, &i);
        %let scale=%SCAN(&ScaleList, &j);
        title2 "PolyChor Output: Validated Scale = &Scale, Test Item = &Test";
        proc freq data=&InputData;
                where &SubSet;
                tables &test * &scale / plcorr maxiter=1000;
                output out = FreqOut PLCORR N;
        run;

        data FreqOut2;
                length Item $50. ;
                set FreqOut (rename = ( _PLCORR_ = &scale N = N&scale));
                /* Calculate p-value */
                P&scale = 2*(1-probnorm(abs(&SCALE/E_PLCORR)));
                /* Trim p-values to 2-decimal places, identify low n counts and
                significant p-values */
                if P&scale < &PVAL and N&scale < &NVAL then Final&scale =
                compress(put(&scale,7.2))||'*^';
                        else if P&scale < &PVAL and N&scale >= &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'*';
                        else if P&scale >= &PVAL and N&scale < &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'^';
                        else Final&scale = compress(put(&scale,7.2));
                Item = %UPCASE("&test");
                Label
                        Item    = "Item"
                        N&scale = "N &scale x &test";
        run;

        proc append base = PolyChor_&i._&j data = FreqOut2;
        run;

        proc datasets;
                delete FreqOut FreqOut2;
        run;
        quit;
%end;

/*****************************************************************/
/*** Polyserial Correlations (Ordinal vs Continuous)   ***/
/*****************************************************************/
%if (%SCAN(&TestType, &i) = 1 and %SCAN(&ScaleType, &j) = 0) OR
 (%SCAN(&TestType, &i) = 0 and %SCAN(&ScaleType, &j) = 1)%then %do;
        %let test=%SCAN(&TestList, &i);
        %let scale=%SCAN(&ScaleList, &j);
        title2 "Polyserial Output: Validated Scale = &Scale, Test Item = &Test";
        data final;
                set &InputData;
                where &SubSet;
        run;

        /* Run polyserial macro based on variable type */
        %if %SCAN(&TestType, &i) = 1 and %SCAN(&ScaleType, &j) = 0 %then %do;
                %polyserial_output(data=final,co=&test,or=&scale);
        %end;
        %if %SCAN(&TestType, &i) = 0 and %SCAN(&ScaleType, &j) = 1 %then %do;
                %polyserial_output(data=final,co=&scale,or=&test);
        %end;

        data Polyserial_&i._&j /* polysclean */;
                length Item $50. ;
                set polyserial (rename = (poly = &scale polyp = P&scale n =
                        N&Scale));
                /* Trim p-values to 2-decimal places, identify low n counts and
                significant p-values */
                if P&scale < &PVAL and N&scale < &NVAL then Final&scale =
                compress(put(&scale,7.2))||'*^';
                        else if P&scale < &PVAL and N&scale >= &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'*';
                        else if P&scale >= &PVAL and N&scale < &NVAL then
                        Final&scale = compress(put(&scale,7.2))||'^';
                        else Final&scale = compress(put(&scale,7.2));
                Item = %UPCASE("&test");
                Label
                        Item    = "Item"
                        N&scale = "N &scale x &test";
                drop co or;
        run;

        proc datasets;
                delete polysclean polyserial poly polyp n final;
        run;
```

```
                        quit;
        %end;

        /*** Close Loops ***/
        %let j = %EVAL(&j + 1);
        %End;   /* End loop over scale items */

%let i = %EVAL(&i + 1);
%End;   /* End loop over test items */

/* Clean the TYPE dataset */
data type;
        set type;
        item = upcase(strip(var));
        drop var;
run;

/* Remove temp dataset */
proc datasets;
        delete temp;
run;
quit;

/* Pull the dataset names below */
ODS OUTPUT Members = WorkDatasetInfo;
proc datasets library=work memtype=data;
run;
quit;

/* Exclude the TYPE-related datasets from being merged into the final CORR dataset */
data WorkDatasetInfo2;
        set WorkDatasetInfo (keep = name) ;
        if name in("TYPE") then delete;
run;

/* Determine the number of datasets to merge */
data _null_;
        set WorkDatasetInfo2 end=lastrec;
        if lastrec then do;
                call symput('total',_n_);
        end;
run;

/* Store each dataset name as a macro variable */
%do n=1 %to &total;
        %global dataset&n;
        data _null_;
                set WorkDatasetInfo2;
                if &n = _n_; /* keep only the nth data record */
                call symput("dataset&n",name);
        run;
%end;

/* Merge Pearson, polychoric, and polyserial datasets into one file */
data CORRS;
        merge
        %do L = 1 %to &total;
                &&dataset&L    /* Pull each dataset name into one merge line */
        %end;
        ;
        by item;
run;

/* Remove work dataset info - only needed these datasets to produce a list of correlation
datasets. */
proc datasets;
        delete WorkDatasetInfo WorkDatasetInfo2;
run;
quit;

/* Add Labels to CORRS dataset if they do exist, otherwise label is variable name */
```

```
/*** First, add labels to rows ***/

/* Pull label names from input dataset */
proc contents data = &InputData out = LabelInfo (keep = name label);
run;

/* Make variable name match that of correlation dataset, and set label to variable name when
missing */
data LabelInfo2;
        set LabelInfo;
        Item = upcase(strip(name));
        if label="" then label = Item;
        drop name;
run;

proc sort data = LabelInfo2;
        by item;
run;

proc sort data = type;
        by item;
run;

/* Merge correlation type with labels and include correlation type as superscript in the label */
data LabelInfo3;
        merge LabelInfo2 (in = in1) type;
        if in1;
        by item;
        label=left(strip(label))||"~{super "||left(strip(type))||"}";
run;

proc sort data=corrs;
        by item;
run;

/* Merge labels into correlation dataset for row values */
data labeled;
        merge corrs (in = in1) LabelInfo3 (in = in2);
        if in1 and in2;
        by item;
run;

/*** Second, add labels to columns ***/

/* Pull column variable names from correlation dataset (with row labels, column labels still
missing) */
proc contents data = labeled out = FinalList (keep = name);
run;

/* Remove prefix "Final" from variable names to match label dataset */
data FinalList2;
        set FinalList;
        Item = upcase(substr(name,6));
        if substr(name,1,5) = "Final";
run;

proc sort data = FinalList2;
        by item;
run;

proc sort data = LabelInfo3;
by item;
run;

/* Merge labels into list of column variable names and create text for 'variable = "label"' */
data FinalList3;
        merge FinalList2 (in = in1) LabelInfo3 (in = in2);
        by Item;
        if in1 and in2;
        TextString = left(strip(Name))||" = '"||left(strip(Label))||"' ";
        keep TextString;
```

```
run;

/* Create a macro with a space-delimited list of labeling commands */
proc sql noprint;
       select TextString into :LabelList separated by ' '
       from FinalList3;
quit;

%put _user_;

/* Label the final correlation dataset columns using the LabelList macro variable */
data CorrsLabeled;
       set labeled;
       label &LabelList;
       keep order label Final: ;
run;

/* Remove all datasets required to label the corrs dataset because they are not needed anymore */
proc datasets;
       delete FinalList FinalList2 FinalList3 Labeled LabelInfo LabelInfo2 LabelInfo3;
run;
quit;

/* Sorts final dataset in order of test list */
proc sort data = CorrsLabeled;
  by order;
run;

/* Print the correlation table to ODS */
title2;
footnote1 "* p < &pval,   ^ n < &nval";
ods escapechar='~';
footnote2 "~{super 0}Ordinal, ~{super 1}Continuous";
footnote3 "1x1 = Pearson Correlation | 1x0 or 0x1 = Polyserial Correlation | 0x0 = Polychoric
Correlation";
options label;

ods rtf file="&OutFile" style=styles.journal bodytitle;
proc print data = CorrsLabeled noobs label;
       VAR label FINAL: ;
run;
ods rtf close;

%mend PROCORR;

/* FOR USER TO FILL */
/* 0 = Ordinal, 1 = Continuous */
/* PROCORR(InputData, TestList, TestType, ScaleList, ScaleType, OutFile, Subset, nval, pval) */

/* Example without subset */
%PROCORR(InputData = ANALYSIS.Generated,
TestList = Q1 Q2 Q3 AsthmaTotal, TestType = 0 0 0 1,
ScaleList = CGIS SGRQSymptoms SGRQActivity SGRQImpacts, ScaleType = 0 1 1 1,
OutFile = &OUTFOLD.\PROCORR.rtf);

/* Example with subset
%PROCORR(InputData = ANALYSIS.Generated,
TestList = Q1 Q2 Q3 AsthmaTotal, TestType = 0 0 0 1,
ScaleList = CGIS SGRQSymptoms SGRQActivity SGRQImpacts, ScaleType = 0 1 1 1, Subset = %str(q1>1),
OutFile = &OUTFOLD.\PROCORR.rtf); */



/*******************************************************************************/
/*  Modifications for %POLYSERIAL to create %POLYSERIAL_OUTPUT  (Cheryl Coon) */
********************************************************************************/
```

1) Immediately after the macro definition "%macro polyserial(data=,co=,or=,bins=3);" (i.e., immediately before "proc iml"), paste:

```
proc datasets library=work;
  delete polyserial poly polyp n;
```

```
run;
quit;
```

2) Immediately after the 'print "Note."' piece at the end of the macro (i.e., immediately before "quit" for proc iml), paste:

```
poly=rho[3];
polyp=pvalue[3];

create poly from poly[colname="poly"];
append from poly;
close poly;

create polyp from polyp[colname="polyp"];
append from polyp;
close polyp;

create n from n[colname="n"];
append from n;
close n;

data polyserial;
  length co or $50.;
  merge poly polyp n;
  co="&co";
  or="&or";
run;
```