Macros to Report Missing Data: An HTML Data Collection Guide
Patrick Thornton, University of California San Francisco, SF, California

ABSTRACT
This paper presents SAS® macro programs that calculate missing data in a data set and produce reports of missing data in HTML. The reports are useful for informing analysts, managers and guiding data collection activities. The macro programs create DATA step and ARRAY syntax to calculate the number and percent of missing data points in a SAS data set. Reports summarize the missing data points overall, by pre-defined variable and observation groups, and by individual variables and observations. The macros also allow the calculation of missing data in changing data sets where only observations and variables due at the time of the report are included in the calculations.

INTRODUCTION
A challenge in reporting missing data from data sets with many variables is to create reports that are clear, concise, and applicable to various stakeholders. For example, an executive summary may be short, summarize over many variables, and present missing data relative to expectations. The following shows an example of an executive summary that might be generated from a data set storing data from a completed research study.

<table>
<thead>
<tr>
<th>Survey</th>
<th>Time</th>
<th>Observations Due</th>
<th>Variable Count</th>
<th>Expected Points</th>
<th>Missing Points</th>
<th>Percent Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>100</td>
<td>20</td>
<td>2000</td>
<td>20</td>
<td>1%</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>3-Months</td>
<td>100</td>
<td>25</td>
<td>2500</td>
<td>650</td>
<td>25%</td>
</tr>
</tbody>
</table>

In contrast, an analyst report may be much longer, include all variables, and highlight problem variables. The SAS macro programs presented here can perform missing data calculations on nearly any data set. Additional macro programs, including some presented here, may be used to report the missing data results to a variety of stakeholders.

In addition to reporting from static data sets, such as those from a completed study, the examination of missing data is a common activity in monitoring a changing data set, such as in an on-going study. As an example, clinical research protocols may dictate data collection on participants at study intake and at 3, 6, 9, and 12-months after intake. Thus, an on-going study would have a changing data set where only a subset of observations may be expected to have completed data for a given time period. The SAS macro programs may be used to monitor missing data in changing data sets by determining the subset of observations and variables that are due at the time.

The following shows the report above as it might be generated from a study data set where the study is ongoing. In the example, 100 participants have entered the study while only 75 participants have been in the study long enough for a 3-month survey to be expected.

<table>
<thead>
<tr>
<th>Survey</th>
<th>Time</th>
<th>Observations Due</th>
<th>Variable Count</th>
<th>Expected Points</th>
<th>Missing Points</th>
<th>Percent Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>100</td>
<td>20</td>
<td>2000</td>
<td>20</td>
<td>1%</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>3-Months</td>
<td>75</td>
<td>25</td>
<td>1875</td>
<td>25</td>
<td>1.3%</td>
</tr>
</tbody>
</table>

Since 75 observations rather than 100 are expected to have data points on 25 variables, the expected points in the second table are 1875 rather than 2500. In addition, missing points on 25 observations and 25 variables are not counted toward the count of missing data points, thus rightly omitting 625 missing points that could not possibly be completed at the time of reporting. The result is that 1.3% missing more accurately reflects the current status of missing data than does the 25% missing reported in the first table.

OVERVIEW
The first section of this paper shows several missing data reports that may be generated within the context of a clinical study, including reports for executives, project managers, analysts, and data collection staff. The second section discusses the execution of the SAS macro programs and the third section presents core strategies, SAS syntax, and macros used to calculate the missing data results. The fourth section lists the complete macro programs, followed by concluding comments.
MISSING DATA REPORTS
This section reviews in more detail the types of reports that may be generated using the macro programs herein.

ANALYST REPORT
By default the SAS macro programs calculate missing observations for each variable in the target data set. The following is a shortened report showing the percent missing for each variable.

**Analyst Report**

Percent Missing=20-39 % Missing

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Total Variables</th>
<th>Observations Due</th>
<th>Expected Points</th>
<th>Missing Points</th>
<th>Obs. Without Missing</th>
<th>Missing Points/Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>jja3_2</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>11</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>jib2</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>10</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>mhb2</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>10</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>jja3_4</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>8</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>jib3</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>8</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>jja3_3</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>7</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>mhb3</td>
<td>1</td>
<td>35</td>
<td>35</td>
<td>7</td>
<td>28</td>
<td>1</td>
</tr>
</tbody>
</table>

This report includes information that might be produced by analysts who are responsible for checking and validating each variable in a data set. Viewing the variables by percent missing, analysts may be able to easily identify problem variables and investigate the cause for missing observations. For example, variables may be missing due to lack of data collection, issues in translation from other software, and/or errors in SAS syntax.

PROJECT MANAGEMENT REPORT
While analysts are often interested in viewing missing data for each variable, a project manager may be interested in the overall percent missing, and the performance of each data collection staff. The following is a portion of a report showing missing data results according the data collector responsible.

**Observation Group = Smith**

<table>
<thead>
<tr>
<th>Survey</th>
<th>Time</th>
<th>Total Variables</th>
<th>Observations Due</th>
<th>Expected Points</th>
<th>Missing Points</th>
<th>Percent Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>17</td>
<td>16</td>
<td>272</td>
<td>41</td>
<td>15.0 %</td>
</tr>
<tr>
<td>All</td>
<td>Intake</td>
<td>9</td>
<td>16</td>
<td>144</td>
<td>25</td>
<td>17.3 %</td>
</tr>
<tr>
<td>All</td>
<td>3 Months</td>
<td>8</td>
<td>16</td>
<td>128</td>
<td>16</td>
<td>12.5 %</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>All</td>
<td>9</td>
<td>16</td>
<td>144</td>
<td>25</td>
<td>17.3 %</td>
</tr>
<tr>
<td>Family</td>
<td>All</td>
<td>8</td>
<td>16</td>
<td>128</td>
<td>16</td>
<td>12.5 %</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>5</td>
<td>16</td>
<td>80</td>
<td>15</td>
<td>18.7 %</td>
</tr>
</tbody>
</table>

The categories of variables, “Survey and Time”, must be defined before the SAS macros can produce this report.
The creation of variable categories will be discussed in a later section “Using the Macros.”

**DATA COLLECTOR/ENTRY REPORT**

Data collectors and data entry staff may benefit from a participant level report of missing data, so that they may identify participants who are missing data that are expected. The following shows only a portion of variables for observation 9.

**Observation Group=9**

<table>
<thead>
<tr>
<th>Survey</th>
<th>Time</th>
<th>Variable Name</th>
<th>Total Variables</th>
<th>Observations Due</th>
<th>Expected Points</th>
<th>Missing Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>jjb1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>jjb2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>jjb3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>jjb4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>jjb0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**EXECUTIVE SUMMARY**

Research directors may be interested in monitoring the overall performance of many studies, thus he/she may only be interested in viewing the percent of missing data across all observations and variables, and within logical categories of variables.

**Executive Summary**

<table>
<thead>
<tr>
<th>Survey</th>
<th>Time</th>
<th>Total Variables</th>
<th>Observations Due</th>
<th>Expected Points</th>
<th>Percent Missing</th>
<th>Obs. Without Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>All</td>
<td>17</td>
<td>35</td>
<td>595</td>
<td>14.2 %</td>
<td>3</td>
</tr>
<tr>
<td>All</td>
<td>Intake</td>
<td>9</td>
<td>35</td>
<td>315</td>
<td>14.2 %</td>
<td>8</td>
</tr>
<tr>
<td>All</td>
<td>3 Months</td>
<td>8</td>
<td>35</td>
<td>280</td>
<td>14.2 %</td>
<td>12</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>All</td>
<td>9</td>
<td>35</td>
<td>315</td>
<td>15.8 %</td>
<td>10</td>
</tr>
<tr>
<td>Family</td>
<td>All</td>
<td>8</td>
<td>35</td>
<td>280</td>
<td>12.5 %</td>
<td>12</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>Intake</td>
<td>5</td>
<td>35</td>
<td>175</td>
<td>13.7 %</td>
<td>18</td>
</tr>
<tr>
<td>Juvenile Justice</td>
<td>3 Months</td>
<td>4</td>
<td>35</td>
<td>140</td>
<td>18.5 %</td>
<td>17</td>
</tr>
<tr>
<td>Family</td>
<td>Intake</td>
<td>4</td>
<td>35</td>
<td>140</td>
<td>15.0 %</td>
<td>18</td>
</tr>
<tr>
<td>Family</td>
<td>3 Months</td>
<td>4</td>
<td>35</td>
<td>140</td>
<td>10.0 %</td>
<td>22</td>
</tr>
</tbody>
</table>
STATISTICAL REPORT
The strategy of categorizing variables may be extended to examine missing observations on variables used in statistical models. For example, models examining differences in variables across time in a pre-test post-test design may require that only observations contributing at all time points are included in the analysis. Thus, variables may be categorized by model, and the “Obs Without Missing” column may be useful in determining completed observations that may be used in the model.

USING THE MISSING DATA MACROS
In the simplest and default case, a missing data report may be generated for each variable in a target data set. The following is required:

1. The data set must have a variable without missing that uniquely identifies each observation;
2. The formats MISP, DIMF, and DMIS must be defined;
3. The initial 8 macro parameters must be given a value where the data set and unique ID variable names are assigned and the macro variable storing a label, gl2, is given the value “Variable”

The two formats DIMF and DIMS are required to exist, but they are not important until variables are grouped into variable categories, in which case the formats should provide a meaningful value for the variable categories. In the following example, the data set from which to report missing results is called ALLC and the unique ID variable is called PART. The execution of the following will list all the variables in the data set ALLC and produce an analyst report.

```sas
proc format;
  value misp 1-19 = "1-19 % Missing"
           20-39 = "20-39 % Missing"
           40-59 = "40-59 % Missing"
           60-79 = "60-79 % Missing"
           80-99 = "80-99 % Missing"
           100 = "100 % Missing";
run;
proc format; value dimf . = "All"; run;
proc format; value dims . = "All"; run;
*set initial parameters;
%let d = allc; * 1: SAS data set;
%let obs = part; * 2: variable name storing unique id in SAS data set;
%let gl1 = ; *label 3: first variable group;
%let gl2 = Variable;  *4: label of second variable group;
%let form1 = dimf; *5: format for first variable group (ITEMD);
%let form2 = dims;  *6: format for second variable group (ITEMD2);
%let by = jjb0; *7: optional: variable used to group observations;
%let f = rg; *8: optional: format for variable used to group observations;
%let dfp = c:\projects\sas presentation\wuss\wuss12\root\;
odds html body="&dfp.body.html" contents="&dfp.contents.html"
frame="&dfp.index.html" newfile=page style=fancyPrinter;
%group(&d,&obs); *if groups not specified default to listing all variables;
%getmis(&d,form1=&form1,form2=&form2,glab1=&gl1,glab2=&gl2);
%analyst(form1=&form1,form2=&form2,bycat=,title=%str (Analyst Report));
odds html close;
```

CREATING AND REPORTING CATEGORIES OF VARIABLES
Summarizing missing data results over categories of variables requires that the variable names be assigned a value on two grouping variables. The macro programs depend on a data set called CATS1 to define the variable categories. CATS1 is a data set where each variable name in the original data set is mapped to a category or value on 2 variables (ITEMD, ITEMD2). Variables TEM and TEM2 are in most cases duplicates of ITEMD and ITEMD2. The following is an example of a CATS1 data set showing 3 variable names in the first column from a larger data set where 2 categories are defined on ITEMD (i.e. 2, 5) and 3 categories on ITEMD2 (i.e. 2, 3, 4):

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>VARNUM</th>
<th>ITEMD*</th>
<th>ITEMD2*</th>
<th>TEM*</th>
<th>TEM2*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jja3_1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Jja6_1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Mhb3</td>
<td>2</td>
<td>14</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

*Note, that the number 1 is reserved and should not be used to code a category for the variable names

The information found in the variables name, type, and varnum may be obtained using PROC CONTENTS:

```sas
proc contents data=allc noprint out=cats(keep=varnum name type);
run;
```
The CATS data set produced by the OUT= option in the CONTENTS procedure may be used to create the CATS1 data set discussed above. The most straightforward way to create CATS1 is to export CATS to an Excel spreadsheet and add columns named ITEMD, ITEMD2, TEM, and TEM2. For each row in Excel type in numbers, beginning with the number 2, to indicate the category to which each variable name belongs where TEM and TEM2 are duplicates of ITEMD and ITEMD2. The two formats DIMF and DIMS must be re-defined to reflect the categories that you assigned to each variable name. For example, in this illustration a variable name with ITEMD=2 and ITEMD2=3 would be categorized as a “Juvenile Justice 3-month” variable using the following formats. Once all variable names are categorized, import the Excel sheet back into SAS and name it CATS1.

```
*variable categories group 1;
proc format;
  value dimf
    1 = "All"
    2 = "Juvenile Justice"
    3 = "Education"
    4 = "Alcohol/Drug"
    5 = "Family";
run;
```

A programmatic way to categorize the variable names is possible if the variables were named according to the categories. For example, the DATA step to the right is a simplified example of creating CATS1 and the variables ITEMD and TEM. The full macro is found in later sections, and similar syntax was used to categorize variables on ITEMD2 and TEM2. The syntax to the right correctly categorizes the variable names because the first 2 characters of each variable name are a consistent indicator of the category of the variable on ITEMD (e.g. JJ indicates juvenile justice).

```
data cats1(keep= itemd itemd2 tem tem2 name type varnum);
  set cats;
  length vg $2;
  itemd = .; itemd2=.
  vg = substr(name,1,2);
  if vg = "jj" then itemd = 2;
  else if vg = "ed" then itemd = 3;
  tem = itemd;
*Add Syntax to code ITEMD2 and TEM2;
run;
```

Whether you create CATS1 via Excel or through a DATA step, the macro %group is designed to accomplish the task of incorporating CATS1 into missing data calculations. By default the groups= parameter of the macro %group is set to "no" which results in all variables being given the same category. When groups= is set to "yes" syntax within the %if %end block is executed. You will need to add SAS syntax to obtain or create the CATS1 data set within the %if %end block in order to create missing data results by category using %group.

```
%macro group(d,uid,anchor=,cats=no,groups=no);
  proc contents data=&d noprint out=cats(keep=varnum name type label nobs);
  run;
  %if &cats = yes %then %do ;
    data cats2;
    /*user defined categories using cats*/
    /*place your own syntax to categorize variable names on ITEMD and ITEMD2*/
    run;
  %end ;
  %else %do ;
    /*default categories*/
    data cats1(keep= itemd itemd2 tem tem2 name type varnum);
    set cats;
    itemd = 2; itemd2=2; tem = 2; tem2=2;
    run;
  %end ;
  /*additional macro code as shown in the last section of this paper*/
%mend;
```

Assuming all the initial macro parameters are set from the previous example, and the CATS1 data set exists that categorizes the variables, the following macro calls will produce a summary of missing data by the variable categories that you specified. The calls of the macros differ from the default case by the addition two parameter specifications in the %group macro call, groups=yes and cats=yes, and one in the %getmis macro call, groups=yes. The report created will be an executive summary.
REPORTING ONLY OBSERVATIONS DUE

It’s also possible to report missing data based on the observations due at the time of a report. The data set ANCHOR is used to define the information needed to limit the count of missing points. Each type 4 missing variable category is defined with the values of ITEMD and ITEMD2. The variable anchor is the name of the date variable in the original data set to be used as a reference date for the determination of variables in the category being due. In this example, the date may be the date of completing the first survey in the category. The variable time is the number of days from the reference date when the additional surveys are due. Once ANCHOR is created, the anchor= parameter in the calls to %getmis and %execut should be set to the value anchor.

data anchor;
  input tem tem2 anchor $ time timel & $;
cards;
  2 3 jjbl 90 JJ Intake
  5 3 mhbl 90 MH Intake
;
run;
proc sort data=anchor;
  by tem tem2;
run;
%let d = allc; *data set;
%let obs = part; *variable storing unique id in data set;
%let gl1 = Survey; *label of first variable group;
%let gl2 = Time; *label of second variable group;
%let form1 = dimf; *format for first variable group;
%let form2 = dims; *format for second variable group;
%let dfp = c:\projects\sas presentation\wuss\wuss12\root\;
%let gl1 = Survey; *label of first dimension;
%let gl2 = Time; *label of second dimension;
%group(&d,&obs,groups=yes,cats=yes); *groups;
%getmis(&d,form1=&form1,form2=&form2,glab1=&gl1,glab2=&gl2,groups=yes,anchor=anchor);
%execut(form1=&form1,form2=&form2,title=%str(Executive Summary),anchor=anchor);

The following shows an exert of syntax generated by the macro programs, where missing calculations are limited by a conditional IF statement that tests whether the due date for each observation is greater than or equal to today’s date. Observations not due are flagged in the ELSE condition. The following section discusses in more detail the use of arrays to calculate missing data.

tod= date();
w_5_3 = 0; /*missing count*/
d_5_3 = 0; /*date due*/
rc_5_3 = 0; /*indication of no missing*/
label w_5_3 = "Family 3 Months";
due_5_3 = 1; /*indication of due*/
d_5_3 = mhbl + 90; /*scheduled due date based on days from reference date*/
if d_5_3 => tod then do; /*
  array xng5_5_3(2) mha3_1 mha3_2;
d  k = 1 to dim(xng5_5_3);
if xng5_5_3(k) = . then do;
w_5_3 = w_5_3 + 1;
end;
end;
array xsg5_5_3(2) mha3_3 mha3_4;
d  k = 1 to dim(xsg5_5_3);
if xsg5_5_3(k) = "" then do;
w_5_3 = w_5_3 + 1;
end;
end;
end;
else due_5_3 = 0; /*not due*/
if w_5_3 = 0 and due_5_3=1 then rc_5_3 = 1; /*obs complete if no missing and due*/

SAS SYNTAX AND MACRO STRATEGY
The SAS programs in this paper use array processing to generate the counts of missing data found in the reports. Simplified excerpts of SAS syntax and macro programs will be presented to make the basic strategy clear. The full macro programs are shown in the last section.

data allc2;
set allc;
w_1_1 = 0;
label w_1_1 = "Total";
array xng1_1_1(13) jjb0 jjb1 jjb2 jja3_1 jja3_2 jja6_1 jja6_2 mhb1 mhb2 mha3_1 mha3_2 mha6_1 mha6_2;
do k = 1 to dim(xng1_1_1);
if xng1_1_1(k) = . then do;
w_1_1 = w_1_1 + 1;
end;
end;
array xsg1_1_1(12) jjb3 jjb4 jja3_3 jja3_4 jja6_3 jja6_4 mhb3 mhb4 mha3_3 mha3_4 mha6_3 mha6_4;
do k = 1 to dim(xsg1_1_1);
if xsg1_1_1(k) = "" then do;
w_1_1 = w_1_1 + 1;
end;
end;
run;

In this example, the variables in the data set ALLC are assigned to array xng1_1_1 if they are numeric and array xsg1_1_1 if they are string. The variable w_1_1 is set to 0 and two DO LOOPS are executed for each observation in data set ALLC. The DO LOOPS iterate through the variables assigned to each array respectively. An IF statement inside each DO LOOP tests the condition that the specific variable in the array is missing (xng1_1_1(k) = . or xsg1_1_1(k) = ""). If a variable is missing, then variable w_1_1 is increased by 1, thus w_1_1 is the count of total missing variables, both numeric and string, for each observation.

The variable w_1_1 can be used to determine the total missing points in the data set:

```sas
proc means data=allc2 noprint;
var w_1_1;
output out=sums sum=w_1_1;
run;
```

The array strategy can be extended to create a count of variables missing for categories of variables other than the total.

data allc2;
set allc;
w_2_2 = 0;
label w_2_2 = "Juvenile Justice Intake";
array xng2_2_2(3) jjb0 jjb1 jjb2;
do k = 1 to dim(xng2_2_2);
if xng2_2_2(k) = . then do;
w_2_2 = w_2_2 + 1;
end;
end;
array xsg2_2_2(2) jjb3 jjb4;
do k = 1 to dim(xsg2_2_2);
if xsg2_2_2(k) = "" then do;
w_2_2 = w_2_2 + 1;
end;
end;
run;

In this example, the variables in ALLC defined as storing items from a juvenile justice intake survey are assigned to array xng2_2_2 if they are numeric and array xsg2_2_2 if they are string. The variable w_2_2 is set to 0 and the two DO LOOPS are executed for each observation in data set ALLC. The DO LOOPS iterate through the variables assigned to each array respectively. An IF statement inside each DO LOOP tests the condition that the specific variable in the array is missing (xng2_2_2(k) = . or xsg2_2_2(k) = ""). If a variable is missing, then variable w_2_2 is increased by 1, thus w_2_2 is the count of total missing intake juvenile justice variables for each observation.

The SAS macro programs in this paper build the SAS syntax to count missing data in a data set using SAS arrays. The programs systematically build arrays based on categories of variables. The following section discusses the naming conventions and some of the logic that the macro programs follow.
STANDARDIZING MISSING TYPES AND VARIABLE NAMING

After you have categorized variable names and created the CATS1 data set (see "Creating and Reporting Categories of Variables"), macros generate 4 types of missing data categories indicated in column 5 in the following table: 1) all variables, 2) all variables in each category in ITEMD, 3) all variables in each category of ITEMD2, and 4) all variables in each combination of categories. The table shows an example of 5 variables categorized on ITEMD and ITEMD2 within each of the 4 missing types. The type 4 missing categories are created when you create the CATS1 data set, and the macro %getmis uses that information to add observations for categories shown in missing types 1-3. Observations for all missing types 1-4 as illustrated below are stored in the CATS2 data set.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JJb1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>W_itemd_itemd2_i</td>
<td>W_1_1_1</td>
<td>Jjb1 1</td>
<td></td>
</tr>
<tr>
<td>JJb2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mhb1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>C_itemd_itemd2_C1_1</td>
<td>5</td>
<td>W_1_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X_type</td>
<td>X1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_type_x_type</td>
<td>I_1_1_1</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>JJb1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJb2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>C_itemd_itemd2_C2_1</td>
<td>4</td>
<td>W_2_1</td>
<td></td>
</tr>
<tr>
<td>mhb1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>C_itemd_itemd2_C5_1</td>
<td>1</td>
<td>W_5_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X_type</td>
<td>X2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_type_x_type</td>
<td>I_2_1_1</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_2_2_1</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>JJb1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJb2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mhb1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>C_itemd_itemd2_C1_2</td>
<td>3</td>
<td>W_1_2</td>
<td></td>
</tr>
<tr>
<td>JJa3_1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>C_itemd_itemd2_C1_3</td>
<td>2</td>
<td>W_1_3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X_type</td>
<td>X3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_type_x_type</td>
<td>I_3_1_1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_3_2_1</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJb1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJb2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>C_itemd_itemd2_C2_2</td>
<td>2</td>
<td>W_2_2</td>
<td></td>
</tr>
<tr>
<td>JJa3_1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JJa3_2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>C_itemd_itemd2_C2_3</td>
<td>2</td>
<td>W_2_3</td>
<td></td>
</tr>
<tr>
<td>mhb1</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>C_itemd_itemd2_C5_2</td>
<td>1</td>
<td>W_5_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X_type</td>
<td>X4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_type_x_type</td>
<td>I_4_1_1</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_4_2_1</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I_4_3_1</td>
<td>52</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cross</td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Key
- w_itemd_itemd2_i stores the variable name and type and is created for each observation
- Cross: count of missing types (usually 4)
- X_type: count of categories in each missing type
- I_type_x_type: value of ITEMD and ITEMD2
- C_itemd_itemd2: Count of variable names in the category

The primary goal %getmis is to translate the information in the CATS2 data set into the creation of SAS syntax that is needed to count missing data. Columns 1-4 show the basic information found in CATS2. Column 7 shows the macro naming conventions used to capture CATS2 information, column 8 shows the macro variables created in this
example, and column 9 shows the macro variable values resolved in this example. The last column shows the variables for this example that would be created using arrays to count missing points in the original data set.

By way of summarizing the macro variable naming conventions in column 7, the macro variable cross (near the bottom) resolves to the number of missing types, usually 4. Macro variables x1 etc. each resolve to the number of categories in each missing type. For example, missing type 1 has 1 variable category because all variables are assigned a 1 on both ITEMD and ITEMD2, thus x1 resolves to 1. The macro variable x2 resolves to 2 because there are two categories of variable names in missing type 2. The macro variables with a prefix of “c” have indexes of the values of ITEMD and ITEMD2 and resolve to the count of variables found in each category. For example, in missing type 2 there are 2 categories of variables, one having 4 variable names (ITEMD=2, ITEMD2=1) and 1 having 1 variable name (ITEMD=5, ITEMD2=1). The macro variable with the prefix of “i” stores the value of ITEMD and ITEMD2 for each missing type and category. The “i” macro variables are resolved to obtain the ITEMD and ITEMD2 values in order to resolve the “c” macro variables.

Column 10 shows the variables created to count missing data in the original data set. These variables all have a prefix of w and the indexes of ITEMD and ITEMD2 for each unique combination of ITEMD and ITEMD2 values. Each variable will contain the count of missing points in the original data set that are found across the variables in the category. For example, w_1_1 will contain a count of missing on all 5 variables because all 5 are categorized as 1 on ITEMD and ITEMD2 in missing type 1. Variable w_5_1 will count missing on 1 variable (mhb1) and w_2_1 will count missing on 4 variables.

SAVING SAS DATA SET INFORMATION TO SAS MACRO VARIABLES

Using the previous naming conventions, the macro program %getmis uses the CATS2 data set to create variables and macro variables. The following section presents simplified highlights of the full syntax found later in the paper. The macro %getmis determines the maximum number that is used to categorize variable names on ITEMD and ITEMD2. The following shows the creation the macro variable dim1 to store the max category number for ITEMD.

```sas
proc sql noprint;
  select max(itemd) into: dim1 from cats1;
quit;
```

The max category numbers from ITEMD and ITEMD2 are used to determine both dimensions of a 2-dimensional array w, where variables in the array are created with the prefix w, the value for ITEMD and ITEMD2 (e.g. w_1_1).

```sas
data cats3 (drop=i j tcats) tcats (drop=itemd itemd2 name type i c j);
  set cats2 end=last;
  by gt itemd itemd2;
  array w(&dim1,&dim2)
    %do k = 1 %to &dim1;
      w_&k._1 - w_&k._&dim2
    %end;
    %str();
  retain %do k = 1 %to &dim1; w_&k._1 - w_&k._&dim2 %end; %str(0);
```

The variables are used to count the number of variable names that are categorized in each combination of ITEMD and ITEMD2.

```sas
w(itemd,ITEMD2) = w(itemd,ITEMD2) + 1;
```

Macro variables are also created using the same naming convention, where each macro variable stores the variable type and variable name.

```sas
call symput("w_",""||trim(left(itemd))||"_\n"||trim(left(ITEMD2))||"_\n"||
  trim(left(w(itemd,ITEMD2))),left(type|"_\n"||name));
```

On the last observation of the CATS2 DATA step, DO looping is used to go through all dimensions of the 2-dimensional array and test if each variable is greater than zero, that is if at least one variable name is found in the category. Missing type 1-4 is determined for each observation. Using the macro naming conventions discussed earlier the following information is stored in macro variables: (a) the missing type 1-4, (b) the count of variables in the category, (c) the format for the category, and (d) the values of ITEMD and ITEMD2.

```sas
if last then do;
  c = 0;
  do i = 1 to &dim1; /*each category in group 1*/
```
do j = 1 to &dim2; /*each category in group 2*/
if w(i,j) > 0 then do; /*at least one variable in dimension*/
    if i = 1 and j = 1 then c = 1;
    else if i = 1 and j > 1 then c = 2;
    else if i > 1 and j = 1 then c = 3;
    else if i > 1 and j > 1 then c = 4;
    dc(c) = dc(c) + 1;
    /*save total variables in category*/
call symput("c_"||trim(left(i))||"_"||trim(left(j)), trim(left(w(i,j))));
    /*format of category*/
call symput("f_"||trim(left(i))||"_"||trim(left(j)),
        trim(left(put(i,& form1..)||" "||trim(left(put(j,& form2..))))));
    /*category indexes*/
call symput("i_"||trim(left(c))||"_"||trim(left(dc(c))), trim(left(i))||""
        ||trim(left(j)));
end;
end;
end;

Finally, the count of variable categories within each missing type is saved, and the total number of missing types.

CREATING ARRAYS
The main purpose of storing the previous information from CATS2 to macro variables is to create a DATA step with
the original data set where the variables are assigned to arrays according the categories of variables in CATS2. The
creation of the arrays is accomplished by going sequentially through each missing data type, determining the
number of unique categories, determining the number of variables in each category, and obtaining each variable
type and name in the category. Within each missing type and category, variables are placed in string or numeric
arrays. The macro %chktype determines the variable type, and %barry creates arrays.

%let sep = _;
%let v =;
%do k = 1 %to &cross; /*for each missing data type*/
%do dd = 1 %to &x&sep&k; /*count of categories*/
    %let d1 = %scan(&i&sep&k&sep&dd,1); /*category on group 1*/
    %let d2 = %scan(&i&sep&k&sep&dd,2); /*category on group 2*/
    %let s = 0; %let n = 0; %let nn =;
    %let vc = w&sep&d1&sep&d2;
    &vc = 0; /*variable to count missing points */
    d&sep&d1&sep&d2 = 0;
    rc&sep&d1&sep&d2 = 0;
    label &vc = "&f&sep&dl&sep&d2"; /*format for the categories*/
%do j = 1 %to &c&sep&dl&sep&d2; /*count of variables in the category*/
    %let wv= &w&sep&dl&sep&d2&sep&j;
    %let type = %trim(%left(%scan(&wv,1))); /*variable type*/
    %let name = %trim(%left(%scan(&wv,2))); /*variable name*/
    %chktype(&name,&type) /*check type of variable*/
%end;
    due&sep&dl&sep&d2 = 1; /*count of variables due*/
    if &a&sep&dl&sep&d2 ne %then %do; /*date variable*/
        d&sep&dl&sep&d2 = &a&sep&dl&sep&d2 + &t&sep&dl&sep&d2; /*date that
dimension is due*/
    if &sep&dl&sep&d2 <= totd then do;
    %end;
/*build array for string or numeric*/
%barry(xsg&dl&sep,xng&dl&sep,&vc,&d1,&d2)
    if &a&sep&dl&sep&d2 ne %then %do; /*date variable*/
end;
else due\&sep\&dl\&sep\&d2 = 0; /*due data vs. today*/
@end;
if w\&sep\&dl\&sep\&d2 = 0 and due\&sep\&dl\&sep\&d2 = 1 then rc\&sep\&dl\&sep\&d2 = 1;
@end;

MISSING DATA MACRO PROGRAMS

/*define variable groups*/
%macro group(d,uid,anchor=,cats=no,groups=no);
proc contents data=&d noprint out=cats(keep=varnum name type label nobs);
run;
@if &cats = yes %then %do;
/*place your own syntax to categorize variable names on ITEMD and ITEMD2*/
data cats1(keep= itemd itemd2 tem tem2 name type varnum);
length _name_ $8 vg $2;
set cats end=last;
if name ne "&uid" then do;
   _name_ = put(name,$8.);
   itemd = .; itemd2 = .;
   vg = substr(name,1,2);
k = k + 1;
   if vg = "jj" then itemd = 2;
   else if vg = "ed" then itemd = 3;
   else if vg = "ad" then itemd = 4;
   else if vg = "mh" then itemd = 5;
   else if vg = "ca" then itemd = 6;
   else if vg = "cb" then itemd = 7;
   else if vg = "cs" then itemd = 8;
   else if vg = "ro" then itemd = 9;
   else if vg = "ys" then itemd = 10;
   per = substr(name,3,1);
i = index(name,"_");
   if i > 0 then do;
      t = substr(name,1,i-1);
      l = length(t)-3;
      tp = substr(name,4,1);
   end;
   if per = "b" then tp = 0;
   if tp = 0 then itemd2 = 2;
   else if tp = 3 then itemd2 = 3;
   else if tp = 6 then itemd2 = 4;
   else if tp = 9 then itemd2 = 5;
   else if tp = 12 then itemd2 = 6;
   tem = itemd; tem2=itemd2;
   if itemd2 ne . and itemd ne . then output cats1;
end;
run;
@end;
%else %do;
/*default categories*/
data cats1(keep= itemd itemd2 tem tem2 name type varnum); 
set cats;
   itemd = 2; itemd2=2; tem=2; tem2=2;
run;
@end;
@if &groups = no %then %do;
/*make each variable its own group*/
data cats1;
   set cats1;
   itemd = 1; /*no groups on 1*/
   itemd2 = varnum; /*each varaible as its own group*/
run;
%end;
proc sort data=cats1;
 by itemd itemd2;
run;
%mend;
%macro chktype(name,type);
 %if &type = 1 %then %do;
   %let nn = &nn &name;
   %let n = %eval (&n + 1);
 %end;
 %else %if &type = 2 %then %do;
   %let sn = &sn &name;
   %let s = %eval (&s + 1);
 %end;
%mend;
%macro barry(pres,pren,cp,g1,g2);
 %if &n > 1 %then %do;
   array &pren&g1&sep&g2(&n) &nn;
   do k = 1 to dim(&pren&g1&sep&g2);
     if &pren&g1&sep&g2(k) = . then do;
       &cp = &cp + 1; /*count all missing for part*/
     end;
   end;
 %end;
 %else %if &n = 1 %then %do;
   if &nn = . then &cp = &cp + 1; /*count all missing for part*/
 %end;
 %if &s > 1 %then %do;
   array &pres&g1&sep&g2(&s) &sn;
   do k = 1 to dim(&pres&g1&sep&g2);
     if &pres&g1&sep&g2(k) = "" then do;
       &cp = &cp + 1; /*count all missing for part*/
     end;
   end;
 %end;
 %else %if &s = 1 %then %do;
   if &sn = "" then &cp = &cp + 1; /*count all missing for part*/
 %end;
%mend;
%macro vars(cross,p,b=,a=);
 %do k = 1 %to &cross; /*1:all all, 2:all dim2, 3:dim1 all, 4:dim1 dim2*/
   %do dd = 1 %to &&x&sep&k; /*count of index pairs in each cross*/
     %let d1 = %scan(&&i&sep&k&sep&dd,1); /*this is a count of i_k variables each with a pair of indexes*/
     %let d2 = %scan(&&i&sep&k&sep&dd,2);
   &b %p&sep%d1&sep%d2 &a
 %end;
%end;
%mend;
%macro sumar(d,cross,pre,npre,outn,by=);
 proc means data=&d noprint;
 %if &by ne %then %do;
   class &by;
 %end;
 var %vars(&cross,&pre);
 output out=&outn sum=%vars(&cross,&npre)
 ;
run;
%mend;
%macro getmis(d,form1=,form2=,by=,glab1=,glab2=,f=,groups=no,anchor=);
 proc sql noprint;
select max(itemd) into: dim1 from cats1;
quit;
proc sql noprint;
    select max(itemd2) into: dim2 from cats1;
quit;
%let dim1 = %trim(%left(&dim1));
%let dim2 = %trim(%left(&dim2));
%let dim = %eval (&dim1*&dim2);
/*case designed to create variable list*/
    %if &dim1 = 1 %then %do;
        %if &dim2 > 1 %then %do;
            data cats2;
            set cats1;
            gt = 1;
            output cats2;
        %end;
    %end;
%else %do; /*creates all categories*/
        %if &dim2 > &dim1 %then %do;
            %let dim1 = &dim2;
        %end;
        %if &dim > 1 %then %do;
            data cats2;
            set cats1(in=o) cats1(in=tw) cats1(in=th) cats1(in=f);
            if o then do;
                itemd2 = 1; itemd=1; gt=1;
            end;
            else if tw then do;
                itemd2 = 1; gt = 2;
            end;
            else if th then do;
                itemd = 1; gt = 3;
            end;
            else do;
                gt=4;
            end;
            run;
        %end;
%else %do; /*case a single dimension on each group*/
        data cats2;
        set cats1;
        gt = 1;
        run;
%end;
proc sort data=cats2;
    by tem tem2;
run;
%if &anchor ne %then %do;
    data cats2;
    merge cats2 anchor;
    by tem tem2;
    if time = . then time = 0;
run;
%end;
%else %do;
    data cats2;
    set cats2;
time =0;
anchor=""
run;
%end;
proc sort data=cats2;
    by gt itemd itemd2;
run;
/*save variable and data set information to macros*/
data cats3 (drop=i j) tcats (drop=itemd itemd2 name type i c j);
    set cats2 end=last;
    by gt itemd itemd2;
    /*save diminsional indexes*/
    array dc(4) dc1 dc2 dc3 dc4;
    retain dc1 dc2 dc3 dc4
    array w(&dim1,&dim2)
    %do k = 1 %to &dim1;
        w_&k._1 - w_&k._&dim2
    %end;
    %str(); retain %do k = 1 %to &dim1; w_&k._1 - w_&k._&dim2 %end;
    %str(0);
    w(itemd,itemd2) = w(itemd,itemd2) + 1; /*count variables falling in each
dimension of group 1 and 2*/
call symput("w_" ||trim(left(itemd))||"_"||trim(left(itemd2))||"_"||trim(left(w(itemd,item d2)))),left(type)||" _"||name);
call symput("a_" ||trim(left(itemd))||"_"||trim(left(itemd2)),trim(left(anchor)));
call symput("t_" ||trim(left(itemd))||"_"||trim(left(itemd2)),trim(left(time)));
if last then do;
c = 0;
do i = 1 to &dim1; /*each dim group 1*/
do j = 1 to &dim2; /*each dim group 2*/
if w(i,j) > 0 then do; /*at least one variable in dimension*/
    if i =1 and j = 1 then c = 1;
    else if i = 1 and j > 1 then c = 2;
    else if i > 1 and j = 1 then c = 3;
    else if i > 1 and j > 1 then c = 4;
    dc(c) = dc(c) + 1;
    /*save total variables in dims*/
call symput("c_" ||trim(left(i))||"_"||trim(left(j)),trim(left(w(i,j))));
call symput("f_" ||trim(left(i))||"_"||trim(left(j)),trim(left(put(i,& form1.)))||"_"||trim(left(put(j,& form2.))));
call symput("i_" ||trim(left(c))||"_"||trim(left(dc(c))), trim(left(i))||"_"||trim(left(j)));end;
end;
do i = 1 to c;
call symput("x_" ||trim(left(i)),trim(left(dc(i))));
call symput("cross",trim(left(c)));
end;
output tcats;
end;
output cats3;
run;
%if &by ne %then %do;
proc sort data=allc;
    by &by;
run;
%end;
%let sep = _;
data allc2 tall (keep =gc tobs tote itobs);
set &d end=last;
  label gc="Observation Group";
%if &by ne %then %do;
  by &by;
%end;
retain tobs itobs gc 0;
%if &by ne %then %do;
  if first.&by then do;
    itobs = 0; /*counts obs within each value of by variable */
  end;
%end;
  itobs = itobs + 1;
totd= date();
tobs = tobs + 1;
tote = &&c_1_1; /*total variables in data set*/
%let v =;
%do k = 1 %to &cross; /*1:all all, 2:all dim2, 3:dim1 all, 4:dim1 dim2*/
%do dd = 1 %to &&x&sep&k; /*count of index pairs in each cross*/
  %let d1 = %scan(&&i&sep&k&sep&dd,1); /*this is a count of i_k variables
  each with a pair of indexes*/
  %let d2 = %scan(&&i&sep&k&sep&dd,2);
  %let s = 0; %let n = 0; %let nn =;
  %let vc = w&sep&d1&sep&d2; %vc = 0; /*counts missing points */
  %let dc&sep&d1&sep&d2 = 0;
  label &vc = "&&f&sep&d1&sep&d2";
  %do j = 1 %to &&c&sep&d1&sep&d2;
    %let wv= &&w&sep&d1&sep&d2&sep&j;
    %let type = %trim(%left(%scan(&wv,1)));
    %let name = %trim(%left(%scan(&wv,2)));
    %chktype(&name,&type)
  %end;
  due&sep&d1&sep&d2 = 1; /*count of variables due*/
%if &&a&sep&d1&sep&d2 ne %then %do; /*date variable*/
  d&sep&d1&sep&d2 = &&a&sep&d1&sep&d2 + &&t&sep&d1&sep&d2; /*date that
  dimension is due*/
  if d&sep&d1&sep&d2 => totd then do;
%end;
  %barry(xsg&d1&sep,xng&d1&sep,&vc,&d1,&d2)
  %if &&a&sep&d1&sep&d2 ne %then %do; /*date variable*/
    %end;
  else due&sep&d1&sep&d2 = 0; /*due data vs. today*/
%end;
  if w&sep&d1&sep&d2 = 0 and due&sep&d1&sep&d2=1 then rc&sep&d1&sep&d2 = 1;
%end;
%end;
%if &by ne %then %do;
  if last.&by then do;
    /*stores group value, and total obs in each group*/
    call symput("itobs","trim(left(itobs))");
    gc = &by;
    output tall;
  end;
%end;
if last then do;
  call symput("itobs0","0 ",trim(left(tobs))); /*total obs for data set*/
  gc = .;
  itobs = tobs;
  output tall; /*data set with total variables per dimension w_l_l...*/
end;
output allc2;
run;
%sumar(allc2,&cross, due, w, obsdue, by=&by);
%sumar(allc2,&cross, rc, w, obsnm, by=&by);
%sumar(allc2,&cross, w, w, sums, by=&by);
data tcats2 (keep=label gc itobs
%vars(&cross,w)
%str( ));
set tcats; /*total variables in each dimension*/
do i = 1 to t;
  set tall nobs=t; /*total observations, or within obs group*/
  label = 1;
  output tcats2;
end;
run;
data tcats3 (drop=_type_ _freq_);
set tcats2 (in=tv) obsdue (in=tod
%if &by ne %then %do;
rename=(&by=gc)
%end;
) sums (in=tm
%if &by ne %then %do;
rename=(&by=gc)
%end;
) obsnm (in=onm
%if &by ne %then %do;
rename=(&by=gc)
%end;
); if tv then label =1;
if tod then label =2;
if tm then label =4;
if onm then label =6;
run;
proc sort data=tcats3;
by gc;
proc transpose data=tcats3 name=variable out=p;
var
%vars(&cross,w)
;if &by ne %then %do;
by gc;
%end;
id label;
run;
data pp;
set p;
label _1 = "Total Variables";
label _2 = "Observations Due";
label _3 = "Expected Points";
label _4 = "Missing Points";
label _5 = "Percent Missing";
label _6 = "Obs. Without Missing";
label _7 = "Missing Points/Obs";
%if &by ne %then %do;
label gc = "Observation Group";
%end;
_8 = _2 - _6;
if _8 > 0 then do;
  _7 = _4/(_2 - _6);
end;
else do;
  _7 = 0;
end;
_3 = _1* _2;
if _3 > 0 then do;
$5 = \frac{4}{3} \times 100$
end;
else do;
$5 = .$
end;
itemd = scan(variable,2,'_')\*1;
itemd2 = scan(variable,3,'_')\*1;
label itemd="glab1";
label itemd2 = "glab2";
drop _label_ variable;
run;
@if &groups = no %then %do;
proc sort data =pp;
by itemd itemd2;
run;
proc sort data =cats1;
by itemd itemd2;
run;
data pp;
merge pp cats1;
by itemd itemd2;
label itemd= "&glab1";
label itemd2 = "&glab2";
run;
%end;
%mend;
*executive summary;
%macro execut(form1=,form2=,title=,anchor=);
proc print data=pp label noobs;
var itemd itemd2 _1-_7;
@if &form1 ne %then %do;
format itemd & form1.
@end;
@if &form2 ne %then %do;
format itemd2 & form2.
@end;
@if &anchor ne %then %do;
where itemd > 1 and itemd2 > 1;
@end;
format _5 pctfmt.
title "&title";
run;
%mend;
*project management report;
%macro pm(form1=,form2=,byf=,title=);
proc print data=pp label noobs;
var itemd itemd2 _1-_7;
@if &form1 ne %then %do;
format itemd & form1.
@end;
@if &form2 ne %then %do;
format itemd2 & form2.
@end;
@if &byf ne %then %do;
by gc;
pageby gc;
format gc & byf.
@end;
format _5 pctfmt.
title "&title";
run;
%mend;
*analyst report;
CONCLUSION

The macro programs presented in this paper were created to standardize the common activity of determining missing data in a data set. The default use of the macro programs allows a simple listing of all variables and percent observations missing. The macro programs also allow variables in the original data set to be categorized on two grouping variables, thus allowing missing data reports to be generated over variable categories. Reports may also be generated within observational groupings, and for only observations with variable categories expected to be complete at the time of reporting. The flexibility allows reporting of missing data in a variety of formats that may be of interest to stakeholders.
CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Patrick Thornton, Ph.D.
Department of Psychiatry, UCSF
1388 Sutter Street Suite 503
San Francisco, CA 94109
Phone: 415 502-8004
Fax: 415 502-6177
Email: Patrick@wolfstrategies.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.