

Missing observations: Identification and resolutions using the SAS System

Doug Moore & Tahereh Nourbakhsh, both of Quintiles, Inc., Kansas City, MO

Abstract:

This paper will discuss identification of cases where missing values or missing records in a data set either create interpretation problems or generate results that do not meet the requirements of the report we are asked to produce. By missing, we mean situations where no value is provided for a variable in one or more records or when all variables corresponding to a key variable are missing in which case the record would usually not show up in the data set. Resolutions to address the problems identified will be provided.

Introduction:

It would be ideal to a SAS programmer if there were no missing observations in any data sets at all, but the reality dictates otherwise. Often missing observations problems emerge due to a variety of reasons including but not limited to: lack of any data or a poor data structure, which will be explained later in this paper. Sometimes, data are intended to be collected, but there is no opportunity to collect them due to uninterpretable results, medical device breakdown, mishandling of the specimen, or subject's death. This paper will shed some light on identifying and properly handling missing observation problems by addressing the following issues:

- I – Identification of
 - a - Key variables
 - b- Common variables
 - c- All possibilities in the planned report shell
- II - Establishment of the meaning of missing variables
- III– Representation of missing data
- IV– Data handling and data reporting

I-a Identification of Key variables

Before using the SAS transpose procedure, identify missing values in all by variables and try to get them corrected. This can be done preferably by the person who created the data set if at all possible. Otherwise, use local solutions,

as appropriate. Below is an example of transpose procedure where missing values in one of the by variables would create problems:

Data set name = x

Ptn o	Visi t	Visit_ M	Visit_D	Visit_ Y	Interva l	Visitdt	Examt m	Vsvar	Vsval
100	1	04	22	03	45	78-02-19	1730	Heart rate	78
100	1	04	22	03	45	78-02-19	1730	Systolic	100
100	1	04	22	03	45	78-02-19	1730	Diastoli c	52
100	1	ND	ND	ND	60	78-02-19	.	Heart rate	.
100	1	ND	ND	ND	60	.	.	Systolic	120
100	1	ND	ND	ND	60	.	.	Diastoli c	.
100	1	04	22	03	75	78-02-19	1800	Heart rate	70
100	1	04	22	03	75	78-02-19	1800	Systolic	124
100	1	04	22	03	75	78-02-19	1800	Diastoli c	72

```
Proc sort data = x; by ptno interval visitdt examtm; run;
```

```
Proc transpose data =x out = y(drop = name);
```

```
By ptno interval visitdt examtm;
```

```
Id vsvar;
```

```
Var vsval;
```

```
Run;
```

```
Title 'Problem with transposing, if visitdt is a by variable';
```

```
Proc print data =y;
```

```
Run;
```

The output will be:

Ptno	Interval	Visitdt	Examtm	Heart rate	Systolic	Diastolic
100	45	78-02-19	1730	78	100	52
100	60	.	.	.	120	.
100	60	78-02-19	.	.	.	.
100	75	78-02-19	1800	70	124	72

As shown in the output, two incorrect records are created for the interval 60 which corresponds to the records in the data set x where the visit date is missing for the systolic and diastolic, but not missing for the heart rate. In other words, a group of data (heart rate, systolic, diastolic) collected at the same interval do not have the same value for the visit date. This data should be cleaned up by the person who created the data set by assigning values for the missing visit dates. In cases, when this cannot be done, the use of local solutions which here for purpose of demonstration: the missing visit date is corrected in order to have a consistent visit date across the group under consideration.

```
Data fix;
  Set x;
  If visit_m = 'ND' and visit_d = 'ND' and visit_y = 'ND' then visitdt = '19FEB1978'd;
Run;

Proc transpose data = fix out = fix (drop = _name_);
  By ptno interval visitdt examtm;
  Id vsvar;
  Var vsval;
Run;

Title 'No more repeated records';
Proc print data = fix;
Run;
```

Ptno	Inteval	Visitdt	Examtm	Heart rate	Systolic	Diastolic
100	45	78-02-19	1730	78	100	52
100	60	78-02-19	.	.	120	.
100	75	78-02-19	1800	70	124	72

It is preferable to have the raw data corrected. If not possible to correct the raw data, it is suggested to create a derived data set with the correction. Then the correction becomes available for other uses: because we may need to join the above data with another data set, say meal schedule, where the variables interval and visit dates exist with the same problem.

I - b: Identification of common variables

When merging two data sets, the common variables (existing in multiple data sets) other than the key variables should probably not be used. A keep or drop statement should be used to limit the presence of the common variables to the preferred data set. The following example shows that the use of common variables is not a good practice at all.

Dataset name: AE

Preterm	Pts	Visit	Trt
Headache	407	1	.
Hyperglycemia	407	3	.
Vomiting	407	4	.
Nausea	407	4	.
Vomiting	410	2	.
Headache	410	4	.
Nausea	410	4	.

Dataset name: dosing

Pts	Visit	Trt
407	1	0
407	2	1
407	2	2
407	4	3
410	1	1
410	2	3
410	3	0
410	4	2

Data problem;

 Merge AE (in =a)

 Dosing;

 By pts visit;

 If a;

Run;

The output will be as follows:

Prefertem	Pts	Visit	Trt
Headache	407	1	0
Hyperglycemia	407	3	2
Vomiting	407	4	3
Nausea	407	4	.
Vomiting	410	2	3
Headache	410	4	2
Nausea	410	4	.

Every time there are multiple events(i.e., Vomiting, Nausea) on the same visit then the variable trt is populated only for the first event. Dropping the common variable trt from the data set AE when merging it with dosing data set would prevent the problem. In general, identify common variables and establish which data set is the master and which one is the transaction data set and consider eliminating the common variable(s) either from the master or the transaction data set.

I - c: Identification of all possibilities in the planned report shell

Sometimes, the data within a data set do not have all the possibilities, in which case, the intended report will not present all possibilities. Using a template that contains all the possibilities from the planned report shell and merging it with the data set would guarantee the inclusion of all possibilities in the report. Below, two examples for this case are illustrated.

Example 1:

Suppose using the following data set we are asked to create a table according to the planned report shell where categories other than those which appear in the data exist. We can do this by the way of joining a template, which embodies all possibilities with the data. This is illustrated below.

Data set name: adverse

Category	Counts
At least one ae	10

Data template;

Length category \$50;

category ='At least on ae'; row =1; Output;

category ='Ae leading to death'; row =2; Output;

```
category=death'; row =3; Output;
un;
```

```
Data adverse;
  Set adverse;
  Row +1;
Run;
```

```
Proc sort data = adverse; by row; run;
Proc sort data =template; by row; run;
```

```
Data adverse;
  Merge template
  Adverse;
  By row;
  If counts = . then counts =0;
Run;
```

The output will be:

Row	Category	Counts
1	With at least one ae	10
2	With ae leading to death	0
3	Death	0

Example 2:

Identification of missing classification variable(s) in GPLOT procedure can be crucial for proper identification of the group value within the legend. Sometimes a classification variable in a data set for some records is missing. For instance, a subject may leave the study, in which case the future treatment (a classification variable) will be missing in the data set. Suppose the researcher using the SAS GPLOT procedure created a plot for each treatment group for each and every subject. As will be seen, the plot for the subject with missing treatment group will be incomplete and even inaccurate. The following example is a case in point. (Note the missing group=1 for subject 2).

Data set name: problem

Subject	Group	X	Y
1	1	1	1

1	1	2	2
1	1	3	3
1	2	1	1
1	2	2	2
1	2	3	3
2	2	1	1
2	2	2	2
2	2	3	3

The following code would create an inaccurate legend and inappropriate symbols for subject 2.

Data problem;

Set problem;

If group =2 then y =y + .1; /* for presentation */

Run;

Proc print ; run;

Symbol1 C =black V =star;

Symbol2 C=red V=circle;

Legend1 position =(top right) across frame value= (tick=1 "Group 1" tick =2 "Group 2");

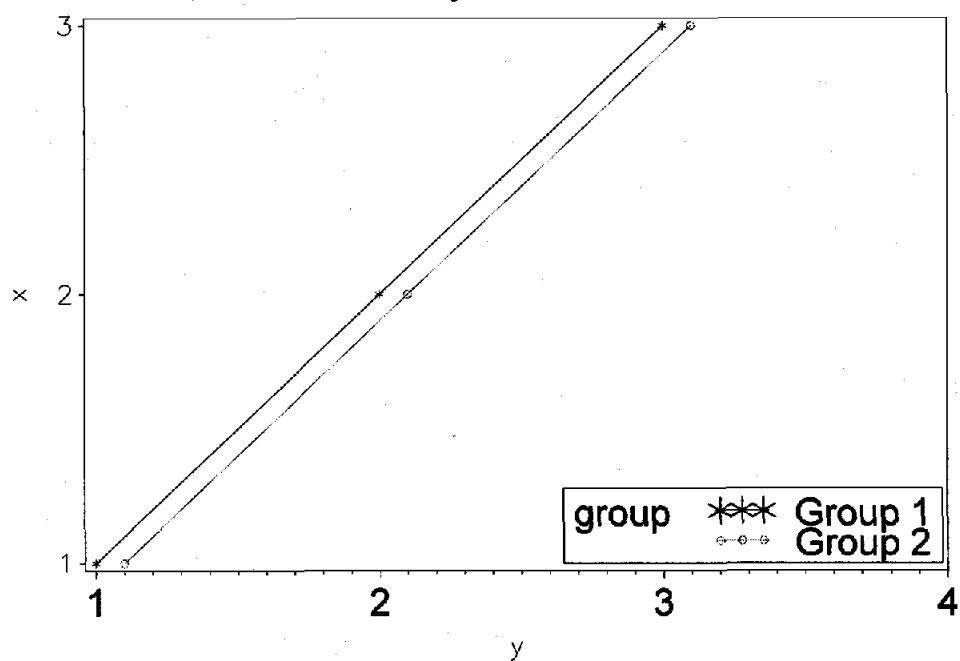
Proc gplot data = problem;

Plot x *Y = group/legend = legend1;

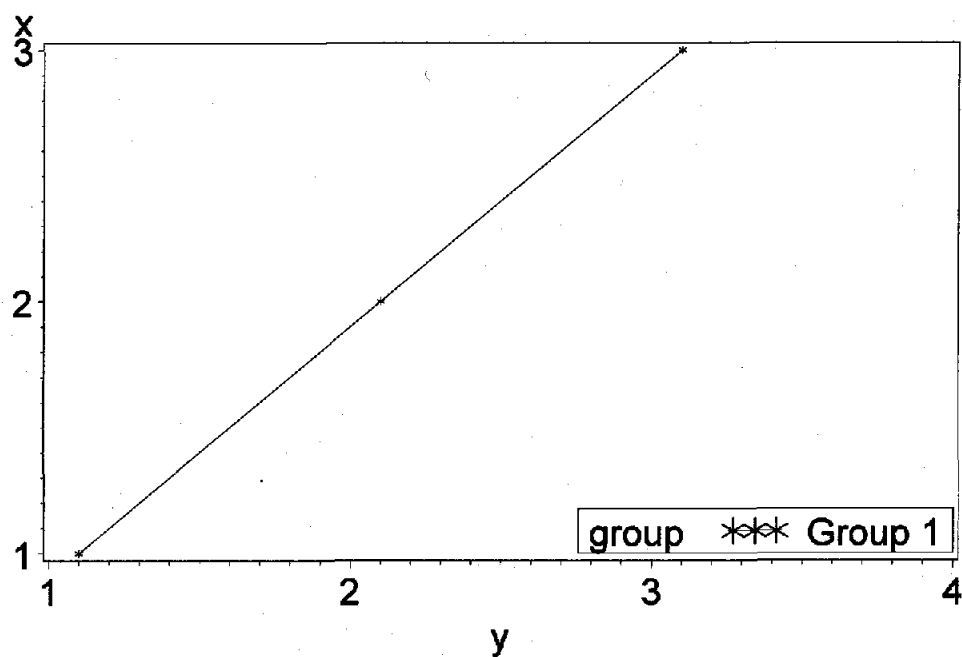
By subject;

Run;

plot O.K. because all groups present
Subject =1



Plot wrong because not all groups present
Subject =2



Notice that the legend for subject 2 with missing treatment group and the intended choice of symbols is not correct. The legend for the subject with missing treatment has one instead of 2 treatments with an incorrect symbol, which is designated to identify the treatment group. To correct these problems, create a template as follows:

```
Data template;
```

```
  Do subject = 1 to 2;
```

```
    Do group = 1 to 2;
```

```
      Output;
```

```
    End;
```

```
End;
```

```
Run;
```

```
Data correct;
```

```
  Merge template
```

```
  Problem;
```

```
By subject group;
```

```
Run;
```

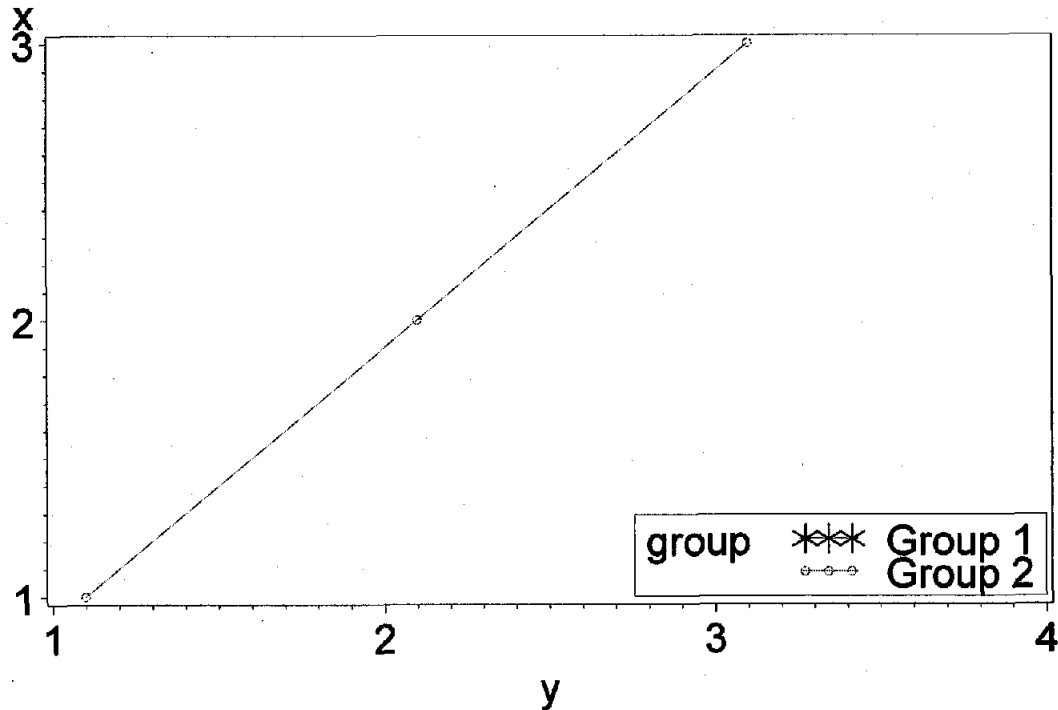
```
Proc gplot data = correct;
```

```
Plot x *Y = group/legend = legend1;
```

```
By subject;
```

```
Run;
```

Plot corrected
Subject =2



Clearly the legends in all plots have 2 treatment groups with correct symbols now.

II – Establishment of the meaning of missing variables

Sometimes a poor data collection design causes for a record not to show up in the data set. Consider the following example:

Two subjects exist in the demography data set; however, the adverse events data set only shows one event for one subject.

Simply because there is no record for both subjects in the adverse data set does not necessarily mean that the second subject did not have an adverse event. The true outcome, whether or not each subject had an adverse event, should be established and documented in a derived data set, which contains a variable indicating the presence or absence of adverse events for the subject.

Another example for establishing the meaning of missing observations would be when using categorical variables to describe occurrences of some events and if there are some missing observations for those variables in the data set. In this case, to avoid underestimation of the counts and percentages, make sure about

the meaning of the blanks so that they will be accounted for when using the SAS frequency procedure.

III – Representation of missing data

Using the SAS options statement, we can set the missing observations to say for example, to zero for the sake of the report we are asked to create, but it could be misleading. Consider the following example:

```
Options missing =0;  
Data x;  
X =.;  
Run;  
Title 'data is 0?';  
Proc print ; run;
```

```
Title 'or is it?';  
Proc univariate;  
Var x;  
Run;
```

Raw data which have “special” assignment for missing data using real numbers such as -1,-9,-99, etc., can be misleading and would affect the descriptive statistics of the variables adversely. So, a good programming practice would be to check formats or the underlying program that created the data set to identify and properly assign the missing observations.

In order to avoid symbols such as -1,-9,-99,'nd', or 'unk' to adversely affect the descriptive statistics of a variable, use special missing values as follows:

```
If value = -99 then value = .u;  
If value ='nd' then x =.n;
```

IV: Data handling and reporting:

When an assignment is used instead of a SUM function (see example below), the result will be missing since the variable w has a missing value regardless of the values of the other two variables.

```
Data x;  
W =.; x = 1; y =2;  
Z =SUM(of w x y);
```

```

A = w + x + y;
Run;
Title 'Careful how you sum';
Proc print ; run;

```

The "missing" option needed in the proc report statement if observations with missing group or order values need to be listed. The following example shows the outcome of not adding this option.

Data set : one

X	Y	Z
1	1	1
2	2	2
3	.	3
4	4	4
5	5	5

```

Title 'report OK?';
Proc report data = one headline nowd;

```

```

Columns x y z;

```

```

define x / group 'X';
define y / group 'Y';

```

```

define z / group 'z';

```

```

run;

```

The record with missing values will not appear in the report.

X	Y	Z
1	1	1
2	2	2
4	4	4
5	5	5

Another important topic in data handling: when comparing variables with special missing values such as “.y” or “.z”, where it is important to know the relative assignment of these values. The following example illustrates this point.

Data one;

X = .y;

Y = .;

If x < y then Z =1;

Else If y < x then Z =2;

If . z < x <1 then A =1;

if .<x <1 then B=1;

Run;

The output will be as follows:

X	Y	Z	A	B
.y	.	2	.	1

Since “.” is less than “.y” and “.y” is less than “.z” then it is true to say that “.z” is the largest missing value.

Conclusion

The knowledge of the type of missing observations is crucial in resolving the issues arising from them. This paper has demonstrated a number of problems emerging from this kinds of observations and has shown how to deal with them in each case. It is hoped that the beginner SAS programmers will find the materials contained in this paper useful.

Reference

SAS Institute Inc., SAS Language Reference: Concepts, Cary, NC , 1999.

Contact Information

Tahereh Nourbakhsh
Quintiles, Inc.
10245 Hickman Mills Drive
Kansas City, MO 64137
Work Phone: 816 – 767 – 6654
Fax: 816 – 767 – 7336
Email: tahereh.nourbakhsh@quintiles.com