

Letting SAS® Find the Data Errors When you Have Multiple Observations and/or Multiple Datasets

Deb Cassidy, Computer Horizons Corporation, Indianapolis, IN

ABSTRACT

Many companies have data entry systems that do not permit invalid values to be entered or they have programs that check for invalid values. Unfortunately, this checking isn't always sufficient. How do you identify records that were not even entered but should have been? For example, subjects may have to complete a daily diary. The dataset should have one observation per day but how do you identify missing days? Maybe your data is stored in multiple datasets. How do you know the subjects are in all the datasets? Many database management systems let you check 2 modules at a time. I recently checked all combinations of 34 datasets in a few minutes. This presentation will a variety of techniques to check data when there are multiple observations or multiple datasets.

Please note that the results of the code are found at the end of the paper. The TITLE statement corresponds to the title for the results. Some of the PROC PRINTs are only included for illustration

DEFINE YOUR CHECKS

The part that is often the most difficult is the part you need to do even if you are checking records by hand. You must define what you can check. Although it may not take long to write some of the SAS code, it may not be worth the time to validate the data. In the diary example each subject was to record 7 days. Was it acceptable to complete only some days? If a partial diary is acceptable, do the days have to be consecutive or can days be skipped? That is, if 4 days are OK do they have to be days 1,2,3,4 or are days 2,4,5,7 acceptable? In a study where any combination of 7 days is acceptable it would not be worth your time to check for missing days. However, another study might require that all 7 days be completed and a statistician wants values imputed for missing values. (To impute missing values means to substitute a non-missing value. This might be the mean of non-missing values from similar subjects or the last non-missing value for that subject.) You would definitely want the fewest

possible imputed values so detailed checking would be worth the time.

CHECKING FOR DUPLICATES

Your data entry system may not permit you to enter duplicate observations. If you just want to verify that you do not have any duplicates you can use:

```
PROC SORT DATA=DIARY OUT=NEWDIARY NODUP;  
  BY SUBJECT;  
PROC PRINT DATA=NEWDIARY;  
  TITLE  
  'NEWDIARY FROM DIARY USING NODUP OPTION';
```

The NODUP option says that if any duplicate records are found in DIARY only the first one will be written out to NEWDIARY. You will also receive a message in the log which tells you how many observations were deleted. The NODUP options checks across all variables. If a subject had two observations with Day 7 but another variable had a different value then NODUP would not consider these as duplicates. If you only want to check some variables you would use:

```
PROC SORT DATA=DIARY OUT=NEWDIARY NODUPKEY;  
  BY SUBJECT DAY;  
PROC PRINT DATA=NEWDIARY;  
  TITLE  
  'NEWDIARY FROM DIARY USING NODUPKEY OPTION';
```

The NODUPKEY option only checks the variables in the BY statement. These are the "key" variables. Two Day 7 observations for the same subject would be considered duplicates.

Sometimes this check is sufficient. But what if you want a list of duplicate records?

The following code would create a list.

```
DATA DUPREC;  
  SET DIARY;  
  BY SUBJECT;  
  IF FIRST.SUBJECT AND LAST.SUBJECT THEN  
    DELETE;  
PROC PRINT DATA=DUPREC;  
  TITLE  
  'DUPREC FROM DIARY AFTER DELETING  
  DUPLICATE SUBJECTS';
```

Your data must already be sorted by the variables in the BY statement. When you use a BY statement, SAS creates temporary variables that can be used for that data step. You can assign the values of the temporary variables to regular variables so a PROC PRINT will show what happens.

```
DATA CHECK;
  SET DIARY;
  BY SUBJECT;
  FIRSTSUB=FIRST.SUBJECT;
  LASTSUB =LAST.SUBJECT;

PROC PRINT DATA=CHECK;
  VAR SUBJECT FIRSTSUB LASTSUB;
  TITLE
  'EXAMPLE OF FIRST. AND LAST. PROCESSING';
```

The first time a value appears for a variable the FIRST variable will be assigned a 1. Otherwise FIRST will have a 0. The last time a value appears for a variable the LAST variable will have a 1. Otherwise LAST will have a 0. SUBJECTS 10 and 12 are only listed once so both FIRST and LAST have a value of 1. The code in the DUPREC data step says to delete records where FIRST and LAST are both 1. If you don't specify an "=" value then "=1" is assumed.

What happens if you have multiple days for each subject?

```
DATA CHECKDAY;
  SET DIARY;
  BY SUBJECT DAY;
  FIRSTSUB=FIRST.SUBJECT;
  LASTSUB=LAST.SUBJECT;
  FIRSTDAY=FIRST.DAY;
  LASTDAY=LAST.DAY;

PROC PRINT DATA=CHECKDAY;
  VAR SUBJECT DAY FIRSTSUB LASTSUB FIRSTDAY
  LASTDAY;
  TITLE
  'EXAMPLE 2 OF FIRST. AND LAST. PROCESSING';
```

This is the part which often confuses people. How can the observation for SUBJECT 11, DAY 1 have FIRSTDAY and LASTDAY set to 1 if DAY 1 already appeared? The observation is the first and last occurrence of DAY 1 WITHIN SUBJECT 11. Whenever there are several variables in the BY statement, FIRST and LAST are assigned for a particular variable within each value for any preceding variables in the BY statement. I think of the data as though I had several smaller datasets. If I had a dataset which only had SUBJECT 11 observations and used a BY DAY statement then I would have the values for FIRSTDAY and LASTDAY as shown above.

To identify duplicate observations when there are several BY variables, you will use the first variable in the BY list that should be unique. Make sure your variables are in the right logical order.

COMPARING DATASETS

If you have several datasets, how do you know you have the same subjects in all of them? You can't tell by number of observations because some datasets might be like the diary dataset above and have several observations per subject. The code above is part of the answer. You need to create new datasets which have unique records for each subject.

```
PROC SORT DATA=DIARY OUT=DIARY1
  (KEEP=SUBJECT) NODUPKEY;
  BY SUBJECT;
PROC SORT DATA=DEMO OUT=DEMO1 (KEEP=SUBJECT)
  NODUPKEY;
  BY SUBJECT;
```

If your data is already sorted you can use:

```
DATA DIARY1;
  SET DIARY (KEEP=SUBJECT);
  BY SUBJECT;
  IF FIRST.SUBJECT;
DATA DEMO1;
  SET DEMO (KEEP=SUBJECT);
  BY SUBJECT;
  IF FIRST.SUBJECT;
```

The KEEP options are used to be more efficient and to eliminate the possibility of overwriting any other variables. Make sure you use a different dataset name or you will overwrite all your data.

After all you datasets have only one observation per SUBJECT, you will MERGE the datasets using the IN= option. This option will also create a SAS temporary variable. If the observation is found in the dataset, the variable will have a 1. Otherwise it is a 0. The following code shows what happens except the IF statements will make 0's be missing for easier readability.

```
DATA CHK_ALL;
  MERGE DIARY1 (IN=A) DEMO1 (IN=B);
  BY SUBJECT;
  IF A THEN INDIARY=1;
  IF B THEN INDEMO=1;
PROC PRINT DATA=CHK_ALL;
  VAR SUBJECT INDIARY INDEMO;
  TITLE 'COMPARING DATASET SUBJECTS';
```

SUBJECT 12 is in DIARY but not DEMO while SUBJECT 14 is in DEMO but not DIARY. There is

a good chance that there was a typographical error and 14 really should be 12.

I like to use the single character variable names such as A,B,C with the IN= option so I know they are SAS temporary variables that are not available outside that data step. The only other time I use single character names is in DO LOOPS or as ARRAY subscripts and I drop these variables at the end of a data step. This way I don't confuse these variables with my real data. I use names like INxxx when I want to keep them for another data step or PROC. Other people always name their variables in the IN=option as INxxx. You should pick a style for naming such variables and be consistent.

DATA CHK_ALL isn't very useful as it stands now. I sort the observations so missing values will appear at the top.

```
PROC SORT DATA=CHK_ALL;
  BY INDEMO INDIARY;

PROC PRINT DATA=CHK_ALL;
  VAR SUBJECT INDEMO INDIARY;
TITLE
'COMPARING DATASET SUBJECTS AFTER SORTING';
```

This way I can easily tell which ones have missing values. I might choose to print only those subjects that do not appear in all the datasets. I could have added the following line in the data step which would delete the subjects in all the datasets.

```
IF SUM(A,B)=2 THEN DELETE;
```

For some studies I will enhance this list by merging another dataset that contains variables which might help me know if there was a legitimate reason for a subject to be missing from a dataset. For example, I have a dataset that contains information collected on Day 7. If the subject dropped out early, then this dataset should not contain information. I also have a summary dataset which includes a status variable and one observation per subject.

```
DATA CHK_ALL2;
  MERGE DIARY1(IN=A) DEMO1(IN=B)
  DAY7INFO(IN=C) SUMMARY(KEEP=SUBJECT STATUS);
  BY SUBJECT;
  INDIARY=A;
  INDEMO=B;
  INDAY7=C;

PROC PRINT DATA=CHK_ALL2;
  VAR SUBJECT INDIARY INDEMO INDAY7 STATUS;
TITLE
'COMPARING DATASETS USING EXTRA INFO';
```

I didn't need an IN= option on the SUMMARY dataset because my STATUS variable will tell me if it is in the dataset. I could also use this as a check that only those subjects with a "complete" status have a value in DAY7INFO.

This code will also work if you have multiple datasets that had multiple key variables. For example, if the study was conducted at different sites, you might need to do your MERGE by SITE SUBJECT.

FINDING MISSING RECORDS

You may be thinking how could I possibly find something that does not exist. You create a "perfect" dataset and compare that against the actual dataset. For example, if every subject is supposed to have 7 days of records then you need to create a dataset which does have Days 1-7 for each subject. You need to start with a dataset which has one observation per subject and only contains the variable SUBJECT. You can use either the PROC SORT or FIRST. processing described above.

```
DATA PERFECT;
  SET DIARY1;
  DO DAY=1 TO 7;
    OUTPUT;
  END;

PROC PRINT DATA=PERFECT;
  VAR SUBJECT DAY;
TITLE 'PERFECT DATASET';

DATA MISSREC;
  MERGE DIARY(IN=A) PERFECT(IN=B);
  BY SUBJECT DAY;
  IF B AND NOT A;

PROC PRINT DATA=MISSREC;
  VAR SUBJECT DAY;
TITLE 'DAYS NOT IN DIARY DATASET';
```

DATA PERFECT will have an observation created for each day for each subject. The IF statement will select those observations that were found in PERFECT but not in DIARY.

What if you wanted to print out all the observations for the subjects which had one or more missing days? You would create a dataset with only the subjects that had missing values and merge that with the original dataset. Since some subjects might be missing multiple days, you must first eliminate any duplicate subjects from the list.

```
PROC SORT DATA=MISSREC
OUT=SUB_LIST(KEEP=SUBJECT) NODUPKEY;
```

```

BY SUBJECT;

DATA DIARYERR;
MERGE DIARY SUB_LIST(IN=A);
BY SUBJECT;
IF A;

PROC PRINT DATA=DIARYERR;
TITLE
'ALL OBSERVATIONS FOR SUBJECTS WITH
MISSING DAYS';

```

CHECKING FOR RANGE OF OBSERVATION COUNTS

What if each subject can have somewhere between 4 and 7 days? You can't create the "perfect" dataset. If days do not have to be consecutive you can't even check to make sure Days 1-4 are there. The only check in this case is the number of records per subject. Once you get a list of subjects that do not have a valid number of days you can do other processing such as merging the list with the original dataset to see what days they do have. The following code would find the subjects and print their diary along with a STATUS field from the SUMMARY dataset.

```

PROC FREQ DATA=DIARY;
TABLES SUBJECT/NOPRINT OUT=SUB_CNT;

PROC PRINT DATA=SUB_CNT;
TITLE 'DATASET SUB_CNT FROM PROC FREQ';

DATA ERROR;
MERGE DIARY SUMMARY
SUB_CNT(IN=A WHERE=(COUNT<4 OR
COUNT>7));
BY SUBJECT;
IF A;

PROC PRINT DATA=ERROR;
TITLE
'SUBJECTS WITH INVALID NUMBER OF
OBSERVATIONS';

```

PROC FREQ will count the number of times each subject appears in the dataset. I used the NOPRINT option because I don't want to actually see the frequencies. The results from PROC FREQ are saved in a dataset SUB_CNT. SAS automatically stores the frequency count in a variable named COUNT. Even though SAS named this variable, do not confuse it with the temporary variables used earlier. Since you only want to see the subjects with invalid counts, use a WHERE= option on the dataset when you merge it. By putting the WHERE= as an option on the dataset, the other subjects are not merged. A WHERE= as a separate statement could have been used instead of the IF A. An IF COUNT<4 OR COUNT>7 could have been used instead of the WHERE=/IF A combination. (Any

volunteers to determine which is most efficient? You can present the results next year.) One thing you may notice is that STATUS and COUNT will be repeated on all the observations for that subject which is why you have several choices here.

CONTENTS CHECK

So far it has been assumed the datasets were all created consistently. However, if one dataset has SUBJECT as a numeric variable and another as a character, you couldn't do the checks. In many cases you can just do PROC CONTENTS on each dataset and compare the results by hand. But what if you had several datasets and several variables that are common to the datasets? This could become a tedious task. The simple solution is to save the results of the PROC CONTENTS and look at it. If all your datasets are in the same catalog you can use:

```

PROC CONTENTS DATA=IN._ALL_ NOPRINT
OUT=CONTLIST(KEEP=NAME MEMNAME TYPE);

PROC SORT DATA=CONTLIST;
BY NAME TYPE MEMNAME;

PROC PRINT DATA=CONTLIST;
BY NAME;
ID NAME;
VAR TYPE MEMNAME;
TITLE 'PROC CONTENTS FOR ALL DATASETS';

```

The NOPRINT option was used because you don't want to see all the normal information. The output dataset will contain many variables. For this example, we are only interested in NAME, MEMNAME, and TYPE so a KEEP= is used. NAME contains the variable name, MEMNAME contains the dataset the variable is from (member name in catalog), and TYPE indicates if it is character or numeric. For TYPE, 1 is numeric and 2 is character. You might also want to check labels, lengths, formats or other information. There will be one observation for each variable for each dataset that it appears in. The BY and ID statements in the PROC PRINT will group the observations for easier review. You can now look of the results quickly to see any inconsistencies. You can also check spelling with this and it is a handy reference to know where a variable can be found.

If the datasets are stored in different catalogs or you want only some datasets then you would need to run PROC CONTENTS for each dataset. Don't forget to save the output to a different dataset each time. You can then SET all the output datasets together.

MACROS

The macros are not included here because this was designed for Beginning Tutorials. If you would like to streamline some of this code, please contact the author for a copy of the macros. You will need to modify them to fit your situations.

CONCLUSION

The real key to checking data is to define what checks you can do regardless of whether you do checking with SAS or by hand. The steps shown here are just some of the ways to check your data. SAS has many other statements or options that can help. The examples in the manuals often provide the help you need to write your code.

SAS is a registered trademarks of SAS Institute Inc. in the USA and other countries.

Contact:

(Home)

Deb Cassidy

3809 Secretariat Court

Columbus, OH 43221

(614) 771-7861 (answering machine)

(Work)

Computer Horizons Corp.

1980 E. 116th Street, Suite 325

Carmel, IN 46032

(317) 574-5600

RESULTS

DIARY

OBS	SUBJECT	DAY	RESULT
1	10	1	3
2	11	1	3
3	11	7	4
4	11	7	4
5	12	1	7
6	13	7	7
7	13	7	8

DEMO

OBS	SUBJECT	AGE
1	10	23
2	11	35
3	13	39
4	14	32

DAY7INFO

OBS	SUBJECT	FINLSCOR
1	10	82
2	11	95
3	13	32

SUMMARY

OBS	STATUS	SUBJECT
1	COMPLETE	10
2	COMPLETE	11
3	STOPPED DAY 5	12
4	STOPPED DAY 7	13

RESPONSE

OBS	SUBJECT	RESPONSE
1	10	Y
2	11	N
3	12	Y
4	13	N

NEWDIARY FROM DIARY USING NODUP OPTION

OBS	SUBJECT	DAY	RESULT
1	10	1	3
2	11	1	3
3	11	7	4
4	12	1	7
5	13	7	7
6	13	7	8

NEWDIARY FROM DIARY USING NODUPKEY OPTION

OBS	SUBJECT	DAY	RESULT
1	10	1	3
2	11	1	3
3	11	7	4
4	12	1	7
5	13	7	7

DUPREC FROM DIARY AFTER DELETING DUPLICATE SUBJECTS

OBS	SUBJECT	DAY	RESULT
1	11	1	3
2	11	7	4
3	11	7	4
4	13	7	7
5	13	7	8

EXAMPLE OF FIRST. AND LAST. PROCESSING

OBS	SUBJECT	FIRSTSUB	LASTSUB
1	10	1	1
2	11	1	0
3	11	0	0
4	11	0	1
5	12	1	1
6	13	1	0
7	13	0	1

EXAMPLE 2 OF FIRST. AND LAST. PROCESSING

OBS	SUBJECT	DAY	FIRSTSUB	LASTSUB	FIRSTDAY	LASTDAY
1	10	1	1	1	1	1
2	11	1	1	0	1	1
3	11	7	0	0	1	0
4	11	7	0	1	0	1
5	12	1	1	1	1	1
6	13	7	1	0	1	0
7	13	7	0	1	0	1

COMPARING DATASET SUBJECTS

OBS	SUBJECT	INDIARY	INDEMO
1	10	1	1
2	11	1	1
3	12	1	.
4	13	1	1
5	14	.	1

COMPARING DATASET SUBJECTS AFTER SORTING

OBS	SUBJECT	INDEMO	INDIARY
1	12	.	1
2	14	1	.
3	10	1	1
4	11	1	1
5	13	1	1

COMPARING DATASETS USING EXTRA INFO

OBS	SUBJECT	INDIARY	INDEMO	INDAY7	STATUS
1	10	1	1	1	COMPLETE
2	11	1	1	1	COMPLETE
3	12	1	0	0	STOPPED DAY 5
4	13	1	1	1	STOPPED DAY 7
5	14	0	1	0	

PERFECT DATASET

OBS	SUBJECT	DAY
1	10	1
2	10	2
3	10	3
4	10	4
5	10	5
6	10	6
7	10	7
8	11	1
9	11	2
10	11	3
11	11	4
12	11	5
13	11	6
14	11	7
15	12	1
16	12	2
17	12	3
18	12	4
19	12	5
20	12	6
21	12	7
22	13	1
23	13	2
24	13	3
25	13	4
26	13	5
27	13	6
28	13	7

DAYS NOT IN DIARY DATASET

OBS	SUBJECT	DAY
1	10	2
2	10	3
3	10	4
4	10	5
5	10	6
6	10	7
7	11	2
8	11	3
9	11	4
10	11	5
11	11	6
12	12	2
13	12	3
14	12	4
15	12	5
16	12	6
17	12	7
18	13	1
19	13	2
20	13	3
21	13	4
22	13	5
23	13	6

ALL OBSERVATIONS FOR SUBJECTS WITH MISSING DAYS

OBS	SUBJECT	DAY	RESULT
1	10	1	3
2	11	1	3
3	11	7	4
4	11	7	4
5	12	1	7
6	13	7	7
7	13	7	8

DATASET SUB_CNT FROM PROC FREQ

OBS	SUBJECT	COUNT	PERCENT
1	10	1	14.2857
2	11	3	42.8571
3	12	1	14.2857
4	13	2	28.5714

SUBJECTS WITH INVALID NUMBER OF OBSERVATIONS

OBS	SUBJECT	DAY	RESULT	STATUS	COUNT	PERCENT
1	10	1	3	COMPLETE	1	14.2857
2	11	1	3	COMPLETE	3	42.8571
3	11	7	4	COMPLETE	3	42.8571
4	11	7	4	COMPLETE	3	42.8571
5	12	1	7	STOPPED DAY 5	1	14.2857
6	13	7	7	STOPPED DAY 7	2	28.5714
7	13	7	8	STOPPED DAY 7	2	28.5714

PROC CONTENTS FOR ALL DATASETS

NAME	TYPE	MEMNAME
AGE	1	DEMO
DAY	1	DIARY
FINLSCOR	1	DAY7INFO
RESPONSE	2	RESPONSE
RESULT	1	DIARY
STATUS	2	SUMMARY
SUBJECT	1	DAY7INFO
	1	DEMO
	1	DIARY
	1	SUMMARY
	2	RESPONSE