Paper 263-29

Mrs. Clean Tackles Dirty Data

Janet E. Stuelpner, New Canaan, CT

ABSTRACT

What is clean data and how do you clean it? This is an age-old question that has some pretty easy answers. There are some techniques that can be used to find invalid data values in both numeric and character data, missing values of any type and duplicate observations. Simple procedures and DATA steps can be used to ferret out inappropriate observations. This paper will illustrate some of the techniques used after data entry is complete and the data has to be reviewed and cleansed.

INTRODUCTION

Data is collected and stored in a database. Before the analysis begins, the data must be checked for accuracy. Each variable must have valid values and any duplicates must be removed. There are some general methods that can be used to check the data and make sure that the data is as accurate as possible. The first task is to identify the invalid values, missing values and the duplicate data. Once the identification takes place, the data must be corrected. Our job here is to identify what is wrong. The methods below point out some of the ways in which the data cleansing can occur. These methods can be used with any type of data. Of course, although date values are considered numeric data, they are handled in a special way. Many of the techniques mentioned here can apply to dates, but a true analysis of date values is outside the scope of this paper. The purpose of this paper is to get you started on the process of checking your data.

METHODS

There are several methods that can be used to clean your data. The method you chose is dependent on the kind of information you need to locate the problems in your data and fix them. You can start to clean the data in the most general way by finding the invalid values. Knowing what and how much data is bad is just the start. Once these are found, you can dig deeper into the data to find out exactly which records contain the invalid data. Below is a step by step methodology of reviewing the data to determine if there are invalid values and then to find out where they are.

METHOD 1: PROC FREQ

The frequency procedure (PROC FREQ) creates a frequency table. A frequency table is used to show the distribution of the values of a variable. Using this procedure, you can find out what the unique values that are populating a variable. This works particularly well for character variables as the data have categorical or discrete values. A PROC FREQ can be used to find out what the invalid or bad values are, if any values are missing or if there are any duplicate values. For example, the variable GENDER should contain only the values ‘M’ or ‘F’. When the frequency procedure is run there might be other values found in this field. The variable might contain lowercase values (‘m’ or ‘f’) as well as uppercase values. There might be some stray characters included as well. A statement about how many values are missing is placed at the bottom of the data also. The PROC FREQ will point out all of this information. An example of the frequency procedure:

```
proc freq data=testdata;
 tables gender;
run;
```

The frequency procedure can be a valuable tool in this case. However, there is one problem with this method. In order to correct the errors that are made during data entry, we must know on which record these invalid values appear. This method fails to give us this needed information. So, let’s move on to the next method.

METHOD 2: PROC PRINT

The next method of determining what the invalid values are is to use a printing procedure, PROC PRINT. This simple procedure, when used in combination with a WHERE statement will point out the bad records that are contained in our data. The nice thing about the WHERE statement or clause is that it can be very simple or complicated as the situation warrants. Using our GENDER variable as an example, the WHERE statement below will cause our print procedure to print a record where the values are invalid.

```
proc print data=testdata;
 where gender not in('M','F');
run;
```

We can choose to print out the information that will uniquely identify the record. If we leave the observation numbers associated with each record, then we know exactly which record has the offending data.

A more complex WHERE statement will print out the invalid values for multiple variables at the same time. Again, we would print the variable that uniquely identifies the record so that we can correct the invalid data.

The one issue that arises with this method is that the procedure is limited to one WHERE statement. As stated previously, the complexity of the statement is dependent on the data being queried. In many cases, this will be enough. However, there are methods that can provide more information. Let's explore one.

METHOD 3: DATA _NULL_

Of course, if you want a method that has even greater flexibility, the DATA step is the way to go. With a DATA step, the information that is printed out can be formatted in a very specific way. Also, there can be many specific IF statements to tailor the request for invalid data. Multiple situations can be checked at the same time. For example, you can check that a start value takes place before an end value. Or you can check to see that if GENDER is 'M' then the diagnosis cannot be 'Uterine Cancer'. This is easy to do with the IF-THEN-ELSE structure rather than the WHERE. You can use multiple IF statements in a DATA step. The fields that are being tested do not have to be connected in any way. The variables can be mutually exclusive.

Another technique is to use functions. Functions such as VERIFY (This returns the position of the first character in the source that is not present in any excerpt. If VERIFY finds every character in source in at least one excerpt, it returns a 0.) and TRANSLATE (This copies a character value, substituting individual characters you specify for other individual characters already specified, returning the altered character value.) can be used to find invalid values and convert multiple errors in one easy step. Of course, there are many other functions that can be used in the DATA step to review any problems.

Using PUT statements, the results from the DATA step can be printed and give valuable information in the data cleansing process. As with the PROC PRINT, the specific records with the errors can be identified and corrected. If you use a FILE PRINT with a DATA _NULL_ the offending records will be printed to the log. If you want the information to be stored in a file, you will need to change the FILE statement to point to the file in which you want the data to be stored.

Why use a DATA _NULL_ and not a regular DATA step. This type of DATA step is use when you are trying to get information and don't need to copy and save the data. The DATA _NULL_ allows you to read the data, manipulate the data, print out the results of the data without saving the data. In the data cleansing process, it is not necessary to create a new data set with dirty data. The purpose here is identification and correction of the original data. If you create a new data set for each cleansing activity, you will waste a great deal of disk space, need to go back to delete all of the unneeded data sets and waste valuable time.

METHOD 4: FORMATS

Another method to check invalid values is to use user-defined formats and informats. There are many possibilities of how this technique can clean the data. The quickest method is to create a format that will lump together all valid values and then highlight any invalid values. The format below will show you how:

```
proc format:
        value $gender   'F','M'   = 'Valid'
                        ' '       = 'Missing'
                        other     = 'Miscode'
```

;
This method allows you to keep track of the missing values and those that were invalid.

Of course, .let's not forget about informats. The user-written informat can test the data as it is being read into a SAS data set. Remember that formats are used to specify what the output will look like. Informats modify the values of a variable as it is read in from the raw data. Error checking can be done as the data is read from the raw data source and the invalid values can be translated into valid values or they can be flagged for other types of investigation and correction.

METHOD 5: COMBINATION

The best thing that a programmer can do is to use a combination of procedures and DATA steps to identify the invalid values. If a PROC FORMAT is used to identify those values that are missing and those values that are incorrect, that can be first step. Then a PROC FREQ can be used to specify how many records fall into each category. The next step would be to find the individual records that contain the problems. Tracking a specific record can be done in several ways. Some of the possibilities are to use a printing procedure such as PROC PRINT or PROC REPORT or a DATA step such as a DATA _NULL_ where you don't create another storage area for the data, but can query the data and get the needed information. Once the records are identified, the information can be queried for correction. Of course, functions can be used to identify and correct the invalid data as well.

METHOD 6: NUMERIC VARIABLES

Numeric variables provide a unique challenge in the data cleansing process. Because most numeric variables contain values that are continuous rather than categorical, we need to approach the problem of identification and correction in a different way. There are many procedures that can be used to check numeric variables. Some of those procedures are mentioned below. Remember that the options that you chose to incorporate into your procedure will determine how much extra information you can gather while hunting for invalid data.

PROC MEANS

A PROC MEANS (or PROC SUMMARY) can be used to identify minimum and maximum values. An example of how the proceudre can be use:

```
proc means data=patients n nmiss min max maxdec=3;
  var hr sbp dbp;
run;
```

PROC UNIVARIATE

PROC UNIVARIATE is a good choice to see if there are any outliers in the data. One feature of this procedure is to find the 5 highest and 5 lowest values of a variable. An example of how the procedure can be used:

```
proc univariate data=patients plot;
  title "Using Proc Univariate to Look for Outliers";
  id patno;
  var hr;
run;
```

PROC TABULATE

A PROC TABULATE can be used to display descriptive data. An example of how the procedure can be used:

```
proc tabulate data=patients format=7.3;
  title "Statistics for Numeric Variables";
  var hr sbp dbp;
  table hr sbp dbp,n*f=7.0 nmiss*f=7.0 mean min max /rtspase=18;
  keylabeln = 'Number'
              nmiss = 'Missing'
              mean  = 'Mean'
              min   = 'Lowest'
              max   = 'Highest';
run;
```

PROC RANK

PROC RANK will help you look for the highest and lowest values by percentage. An example of how the procedure can be used:

```
proc rank data=patients out=ranks groups=5;
  var hr;
  ranks range;
run;
```

Of course, the techniques that were discussed for the character variables will work as well. We can check for valid ranges and see if the values of the variable fall outside the range, or check for the individual values.

METHOD 5: MACROS

The repetitiveness of the search for invalid data can be made easy by using macros. This will allow you to automate the process and allow the programmer to create one program that can be varied as needed. A macro will allow you to change the variable to be reviewed without re-writing the program each time. Macros can be very simple or complex depending on skill level and the amount of detail that is needed in the data cleansing process. It is something that can be very helpful when the task at hand is large and contains a great deal of repetition.

CONCLUSION

Give 10 programmers the same task and they will produce 10 different ways to do it. This paper has shown the beginning of the thought process to guide you through data cleansing. The purpose is to make you start thinking of the many ways to approach the task. It gives you a hint of the methods that are commonly used to find the invalid data. A combination of DATA steps and procedures will help to identify the invalid values, missing values and duplicate data. The complexity of the method you use is only restricted by your ability to program and your imagination.

REFERENCES

Cody, Ron, *Cody's Data Cleaning Techniques Using* SAS® *Software*, Cary, NC, SAS Institute Inc. 1999. 226 pp.

ACKNOWLEDGMENTS

I would like to take this opportunity to acknowledge all of the help and support given to me by my husband, Robert Stuelpner. He diligently read and reread this paper in an effort to correct the obvious errors and keep me on track. His criticisms were constructive and his support never ending.

Author Contact:

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

Janet E. Stuelpner
326 Old Norwalk Road
New Canaan, CT 06840
Voice: (203) 966-7520
Fax: (203) 966-8027
Email: jstuelpner@usa.net