Paper 56-26

Data Cleaning and Base SAS Functions

Caroline Bahler, Meridian Software Inc

Introduction

Functions are small programming subroutines and can be defined as the "work horses" of any data cleansing operation. "Dirty data", unfortunately, is the norm especially within demographic data where input errors are common. In addition, often there is the necessity of converting a variable within a data source from one data type into another (for example from a character date to SAS® date) in order to conform to pre-existing data. This paper is not an exhaustive study of all functions available within SAS® to cleanse data. Instead the objective of this paper is to discuss the most commonly used base functions within the following categories: data type conversion (input/put), character, date/time, and "geographic".

General Comments on Data Cleansing

Data cleaning, cleansing, or scrubbing all are synonymous terms for the same process – the removal of data values that are incorrect from a data source[5]. Dirty data refers to data that contains incorrect/ erroneous data values.

Data cleansing is an art not a science. Each set of data that needs to be cleaned has its own set of headaches and cleansing solutions. Therefore, the following functions allow the "cleanser" to tackle many types of problem in the basic cleansing line instead of being specific solutions for a defined situation.

Data cleansing requires the following information:

- Is there a pre-existing data source, either a database table or data set that the new data will be added to?
- Are there any business rules that need to be used during cleansing? Often one of the cleansing chores is to convert a field into another using a set of criteria?
- What are the cleansing problems in the new data? Before any cleansing effort can begin a inventory of all of the obvious flaws in the data needs to be compiled.

Finally, some general rules of data cleansing:

- The data is ALWAYS dirtier than you thought it was.
- New problems will always come to light once the obvious ones have been solved.
- Data cleansing is an on-going process that never stops.

Overview of Functions

Data Type Conversion Functions

Frequently a variable value needs to be converted from one format to another. For example, data within a new mailing list contains the zip code as a numeric value but your permanent customer data set has zip code as a character variable. The zip code can be converted from numeric to character using the PUT function:

```
data newlist;
 set newdata.maillist;

 zipcode = PUT(zip,z5.);
run;
```

In the previous example, a new character variable called zip code was created utilizing the PUT function. Conversely, if the zip code in the new mail list is character but it needs to be numeric then the INPUT function can be used[2]. For example,

```
data newlist;
 set newdata.maillist;

 zipcode = INPUT(zip,8.);
run;
```

In addition, to character / numeric conversions the PUT and INPUT functions can be used in the conversion of data/time values into character variables and vice versa.

Character Functions

Frequently it is necessary to change the form of a character variable or use only a portion of that value. For example, you might need to uppercase all letters within the variable value. In this case, a new variable does not need to be defined for the function to be used.

The following is a list of character functions that are extremely useful in data cleansing.

Function	Use
Compress	Removes specified characters from a variable. One use is to remove unnecessary spaces from a variable.
Index, indexc, indexw	These functions return the starting position for a character, character string, or word, and are extremely useful in determining where to start

Function	Use
	or stop when sub stringing a variable.
Left	Left justifies the variable value.
Length	Returns the number of characters with a character variable value.
Lowcase	Lower cases all letters within a variable value.
Right	Right justifies the variable value.
Scan	Returns a portion of the variable value as defined by a delimiter. For example, the delimiter could be a space, comma, semi-colon etc.
Substr	Returns a portion of the variable value based on a starting position and number of characters.
Translate	Replaces a specific character with characters that are specified.
Tranwrd	Replaces a portion of the character string (word) with another character string or word. For example, a delimiter was supposed to be a comma but data in some cases contains a colon. This function could be used to replace the comma with a colon.
Trim	Removes the trailing blanks from the right-hand side of a variable value.
Upcase	Upper cases all letters within a variable value.

If you need to use one of these functions on a numeric variable then it is preferable to first convert the numeric value into a character value (see previous section). By default, conversion from numeric to character will occur when using these functions within the DATA step with a warning placed at the end of the DATA step.

For example –

A new mailing list contains a date value that is a character and it needs to be converted into a SAS date value. An additional challenge is that the character value does not match any date informats.

Date character value format - Mon dd, yyyy

The solution to this conversion has two (2) steps –

1. Need to re-arrange the date character value so that the date is in the following format – ddmonyyyy, i.e. date9. informat.
2. Convert the new character value to a date value.

```
data newlist;
 set newdata.maillist;

/* Extract month, day and year */
/* from the date character var[a] */

   m = scan(date,1,' ');
   d = scan(date,2,' ');
   y = scan(year,2,',');
   dd = compress(d||m||y,' ,');

/* Convert mon, day, year into */
/* new date variable[b]        */
   newdate = input(dd,date9.);
 run;
```

a) In this case the SCAN function was used, but the SUBSTR function could also have been used to extract the month, day, and year from the original character date variable. The SCAN function was used because the data values contained a space or comma delimiter. Note that the comma was used to delimit the year and the text portion was the second and NOT the third. The reason for this is the text string has only two pieces, month and day, before the comma and year after the comma, when the comma is used as the only delimiter. The SUBSTR function would have been the only choice if a delimiter had not been available.

b) Conversion of the resulting mon, day and year variables into a new variable was accomplished by utilizing the COMPRESS function and INPUT functions. The COMPRESS function was used to remove any spaces present within the three (3) concatenated variables and to remove the comma within the day variable value. Note – by choosing to use the scan function for extracting the day value from the original date variable, the comma was left with the day value since there was no space between the day and comma. Finally, the use of the INPUT function creates a new variable with a SAS date value.

Parsing along –

In many data cleansing scenarios, a single data variable contains multiple pieces of data that need to be split into separate variables. If there is no delimiter between them, then the variable must be divided using the SUBSTR (substring) function.

The SUBSTR function requires a starting point and the number of characters to be kept in the new variable. In some cases however, the starting point may not be constant. In those cases then several

other character functions can be useful in determining where to start sub stringing.

Some examples -

- The last three characters of a variable are an id that requires a new variable. An additional hurdle is that the variable length is not constant.

  To extract the last 3 characters in this case the LENGTH function is used to define the starting position.

```
Data cleandata;
  Set dirtydata;
 a = substr(oldid,length(oldid)-3);
 put a;
run;
```

Oldid	New Id
A123B24	B24
AS1456B35	B35

- A character or a specific set of characters occur where the character string starts. Using the data from the last example, the last 3 characters can be extracted using INDEX to define the starting position.

```
data cleandata;
  set dirtydata;
 oldidx = upcase(oldid);
 a = substr(oldid,index(oldidx,'B'),3);
 put a;
run;
```

  In this case, the length of the character string to be extracted was specified. Note – case is important here so the following variation removes any case problems without affecting the case of the extracted string.

Date/Time Functions

Date/Time functions are a set of functions that return portions of date time, date, or time values or convert numeric values for month, day and year or hour, minute and seconds into SAS date or time values. These functions are especially useful for extracting the date and time from a date time value or converting separate month, day and year values into a SAS date value.

The following is a list of date/time functions that are extremely useful in data cleansing.

Function	Use
Month	Returns the month from a date value
Day	Returns the day from a date value
Year	Returns the year from a date value
Hour	Returns the hour from a time value
Minute	Returns the minute from a time value
Second	Returns the second from a time value
Datepart	Returns the date only from a date time value
Timepart	Returns the time only from a date time value
MDY	Returns a date value from the numeric values for month, day and year into a date value
HMS	Returns a time value from the numeric values for hour, minutes and seconds
Today()	Returns the current date value.
Date()	Returns the current date value.
Datetime()	Returns the current datetime value.

For example –

The new mailing list has in one case separate variables for month, day, and year for one date. The problem is that this data needs to be added to a pre-existing data set that contains this information as a single SAS date. If the data is numeric, then the use of the MDY function converts the separate variables into a single date value variable. However, if the data is character then the conversion to numeric should occur first and then the conversion to the date value.

The following codes shows how this two(2) part process can occur within one (1) statement.

```
data newlist;
  set newdata.maillist;

  newdate=mdy(input(mon,2.),
              input(day,2.),
              input(yr,4.);
run;
```

Note: as noted before if the character variables are not converted to numeric before the use in the MDY function, SAS will automatically convert these values to numeric and issue a warning at the end of the DATA step. However, good programming practices prefer the conversion of character variables into numeric before their use in a function like MDY.

"Geographic" Functions

"Geographic" functions consist of a set of state and zip code functions that can be used to verify state name spelling and state abbreviations (to a point). All useful when data cleansing, especially the conversion of the zip code to the abbreviation. This

is a function that can be used to verify that the abbreviation for the state is correct. However, this conversion has another use in identifying the zip codes that are potentially incorrect.

The following is a list of date/time functions that are extremely useful in data cleansing.

Function	Use
Stname	Returns state name in all upper case from state abbreviation.
Stnamel	Returns state name in mixed case from state abbreviation.
Zipname	Return state name in upper case from zip code.
Zipnamel	Returns state name in mixed case from zip code.
Zipstate	Returns state abbreviation from zip code.

For example –

```
data newlist;
  set newdata.maillist;

  if state ne zipstate(zip) then
     stateflag=1;
   else
     stateflag=0;
Run;
```

In the example, above the value returned by the ZIPSTATE function is compared to the variable containing the state abbreviation. If the two state abbreviations do not match, then a flag is set.

Putting it all together

Appendix 1 is an example of using all of the function types to cleanse a set of data that is going to be added to a pre-existing data table in a data warehouse. Table 1 lists the data in its "raw" form. All variables within the "raw" data set are character variables.

The following changes need to be made:

- Change moddate to datetime value
- Upper case all state abbreviations
- Ensure all phone numbers use only a dash as divider.
- Add identifier – the data needs a character variable that uniquely identifies each row. The identifier needs to start with 1000.
- Determine if state abbreviations match zip code determined abbreviations

Table 2 lists the data after cleansing and table 3 is a listing that identifies the case where the state abbreviations do not match. Note – the mismatch could be caused by either a data entry problem with the state abbreviation or a data entry problem with the zip code. In this case, our program has not identified the actual problem. Instead the program has identified only that there is a problem.

Conclusion

This paper was not an exhaustive study of all functions available within SAS® to cleanse data. Instead it discussed the most common base functions used to perform:

- data type conversions
- parse or change the justification or case of character variables
- parse and create date/time values
- determine state names from state abbreviations and zip codes

References

1. Functions and Call Routines, Base SAS Software. SAS On-line Documentation version 8. SAS Institute, Inc. Cary, NC.
2. Delwiche, Lora D. and Slaughter, Susan J. 1998. The Little SAS Book, Second Edition, SAS Institute, Inc. Cary NC. pp 204-205
3. Zip codes for basic example – **www.usps.com**
4. Howard, Neil. 1999. Introduction to SAS Functions. Proceeding of the Twenty-fourth Annual SAS User's Group International Conference. SAS Institute, Inc. Cary NC. pp 393-399.
5. Karp, Andrew. 1999. Working with SAS Date and Time Functions Proceeding of the Twenty-fourth Annual SAS User's Group International Conference. SAS Institute, Inc. Cary NC. pp 400-406.
6. Cody, Ron. 2000. Cody's Data Cleaning Techniques Using SAS Software. SAS Institute, Inc. Cary NC.

Trademarks

Contact Information

Caroline Bahler
Meridian Software, Inc.
12204 Old Creedmoor Road
Raleigh, NC 27613
(919) 518-1070
merccb@meridian-software.com

Appendix 1 - Basic Example

Cleansing Program.

```
data clean(drop = date time i moddate);
 set newdata.maillist;
 format datetime datetime21.;

 retain i 1000;

 /* identifier */
 i = i + 1;
 id = put(i,4.);

 /* conversion to datetime */
 date = compress(scan(moddate,2,' '),',')||
            scan(moddate,1,' ')||
            scan(moddate,3,' ');
 time = scan(moddate,4,' ');
 datetime = input(compress(date||":"||time),datetime21.);

 /* upper case state */
 st = upcase(st);

/* ensuring dash is divider in phone */
 phone = tranwrd(phone,'/','-');

 /* zip check on state abbrev */
 stabbrv = zipstate(zip);
 if stabbrv ne st then flag = "*";

run;
```

Table 1. Original Data

Id	First	Last	Address	City	State	zip	Area	Phone	Moddate
1001	Brenda	Jones	101 1st St	Omaha	NE	68101	123	147-2457	Jan 17, 2001 20:07:49
1002	Jim	Smith	5 Keyland Lane	Portland	ME	04103	213	125-4596	Jan 17, 2001 20:07:49
1003	John	Handford	3269 Graceland Ave	Memphis	MI	37501	111	235-9875	Jan 17, 2001 20:07:49
1004	Jane	Kew	5684 Jonesboro Rd	Blowing Rock	NC	28605	102	286-5468	Jan 17, 2001 20:07:49
1005	Mary	Roderick	201 Garland Dr	Atlanta	ga	30344	412	965/5692	Jan 17, 2001 20:07:49

Table 2. Cleaned Data

Id	First	Last	Address	City	State	zip	Area	Phone	Moddate
1001	Brenda	Jones	101 1st St	Omaha	NE	68101	123	147-2457	17JAN2001:20:07:49
1002	Jim	Smith	5 Keyland Lane	Portland	ME	04103	213	125-4596	17JAN2001:20:07:49
1003	John	Handford	3269 Graceland Ave	Memphis	MI	37501	111	235-9875	17JAN2001:20:07:49
1004	Jane	Kew	5684 Jonesboro Rd	Blowing Rock	NC	28605	102	286-5468	17JAN2001:20:07:49
1005	Mary	Roderick	201 Garland Dr	Atlanta	GA	30344	412	965-5692	17JAN2001:20:07:49

Table 3. State abbreviation check

Id	Orig State	State	Flag
1001	NE	NE	
1002	ME	ME	
1003	MI	TN	*
1004	NC	NC	
1005	**ga**	**GA**	