

SAS®: Data Manipulation Tools

Audrey Yeo, Athene USA, West Des Moines, Iowa

Abstract

Let's face it, the data provided to us is typically never easy to work with. Missing values, dates that aren't populated the way we need (e.g. Year, Month, Day vs. Month, Day, Year), leading zeros that need to be removed from a string, leading zeros that need to be added to a string, to name a few. We must work through and massage the data before we can start creating reports with it. In this paper, we present a collection of basic data manipulation tools that should help make the data cleaning easier.

1. PROC STDIZE – Can be used to replace missing values

It is easy to change missing values to a zero from a . (dot) in a DATA step, as shown below

```
IF MISSING(VAR) THEN VAR = 0;
```

However, PROC STDIZE is a much simpler and effective tool to do that.





Data set (Input):

	Make	Model	Origin	MSRP
1	Acura	MDX	Asia	36945
2	Acura	TL 4dr	Asia	
3	Acura	NSX coupe 2dr..	Asia	89765
4	Audi	A4 3.0 4dr	Europe	
5	Audi	A6 3.0 4dr	Europe	36640
6	Audi	S4 Avant Quattro	Europe	49090
7	BMW	325xi 4dr	Europe	
8	BMW	530i 4dr	Europe	44995
9	BMW	325xi Sport	Europe	32485
10	Buick	Regal LS 4dr	USA	
11	Buick	Park Avenue Ultr..	USA	40720
12	Cadillac	SRX V8	USA	
13	Cadillac	Tahoe LT	USA	41465
14	Chevrolet	Tahoe LT	USA	
15	Chevrolet	Malibu 4dr	USA	18995
16	Chrysler	PT Cruiser 4dr	USA	17985
17	Chrysler	Pacifica	USA	
18	Dodge	Caravan SE	USA	
19	Dodge	Ram 1500 Regul..	USA	20215
20	Ford	Escape XLS	USA	
21	Ford	Freestar SE	USA	26930
22	GMC	Safari SLE	USA	
23	Honda	Pilot LX	Asia	27560

Code:

```
PROC STDIZE DATA=CARS REPNLY MISSING=0 OUT=CARS2;
    VAR MSRP;
RUN;
```

Data set (Output):

	 Make	 Model	 Origin	 MSRP
1	Acura	MDX	Asia	36945
2	Acura	TL 4dr	Asia	0
3	Acura	NSX coupe 2dr...	Asia	89765
4	Audi	A4 3.0 4dr	Europe	0
5	Audi	A6 3.0 4dr	Europe	36640
6	Audi	S4 Avant Quattro	Europe	49090
7	BMW	325xi 4dr	Europe	0
8	BMW	530i 4dr	Europe	44995
9	BMW	325xi Sport	Europe	32485
10	Buick	Regal LS 4dr	USA	0
11	Buick	Park Avenue Ultr...	USA	40720
12	Cadillac	SRX V8	USA	0
13	Cadillac	Tahoe LT	USA	41465
14	Chevrolet	Tahoe LT	USA	0
15	Chevrolet	Malibu 4dr	USA	18995
16	Chrysler	PT Cruiser 4dr	USA	17985
17	Chrysler	Pacifica	USA	0
18	Dodge	Caravan SE	USA	0
19	Dodge	Ram 1500 Regul...	USA	20215
20	Ford	Escape XLS	USA	0
21	Ford	Freestar SE	USA	26930
22	GMC	Safari SLE	USA	0

Let's say we have the input data set above, where there are some missing values in the MSRP column, and we want to replace all missing values in that column with the value 0.

The REPNLY option in the code above tells SAS to replace only missing data and the MISSING=0 option tells SAS that we want the missing value to be replaced with a value 0.

Let's say we have multiple columns with missing numeric values (as shown below) and we want to substitute all the numeric missing values with 0s.

Data set (Input):

	Make	Model	Origin	MSRP	Invoice	Sale
1	Acura	MDX	Asia	36945	.	37000
2	Acura	TL 4dr	Asia	.	21671	22000
3	Acura	NSX coupe 2dr..	Asia	89765	79978	.
4	Audi	A4 3.0 4dr	Europe	.	28846	29500
5	Audi	A6 3.0 4dr	Europe	36640	33129	35555
6	Audi	S4 Avant Quattro	Europe	49090	.	50000
7	BMW	325xi 4dr	Europe	.	.	.
8	BMW	530i 4dr	Europe	44995	41170	.
9	BMW	325xi Sport	Europe	32485	30110	31050
10	Buick	Regal LS 4dr	USA	.	22835	23050
11	Buick	Park Avenue Ultr..	USA	40720	36927	.
12	Cadillac	SRX V8	USA	.	43523	44000
13	Cadillac	Tahoe LT	USA	41465	.	.
14	Chevrolet	Tahoe LT	USA	.	36287	35000
15	Chevrolet	Malibu 4dr	USA	18995	.	.
16	Chrysler	PT Cruiser 4dr	USA	17985	16919	.
17	Chrysler	Pacifica	USA	.	28725	28000
18	Dodge	Caravan SE	USA	.	20508	.
19	Dodge	Ram 1500 Regul..	USA	20215	.	20000
20	Ford	Escape XLS	USA	.	.	15670
21	Ford	Freestar SE	USA	26930	24498	.
22	GMC	Safari SLE	USA	.	23215	.
23	Honda	Pilot LX	Asia	27560	.	27000

Code:

```
PROC STDIZE DATA=CARS3 REPNLY MISSING=0 OUT=CARS4;
    VAR _NUMERIC_;
RUN;
```

Data set (Output):

	Make	Model	Origin	MSRP	Invoice	Sale
1	Acura	MDX	Asia	36945	0	37000
2	Acura	TL 4dr	Asia	0	21671	22000
3	Acura	NSX coupe 2dr..	Asia	89765	79978	0
4	Audi	A4 3.0 4dr	Europe	0	28846	29500
5	Audi	A6 3.0 4dr	Europe	36640	33129	35555
6	Audi	S4 Avant Quattro	Europe	49090	0	50000
7	BMW	325xi 4dr	Europe	0	0	0
8	BMW	530i 4dr	Europe	44995	41170	0
9	BMW	325xi Sport	Europe	32485	30110	31050
10	Buick	Regal LS 4dr	USA	0	22835	23050
11	Buick	Park Avenue Ultr..	USA	40720	36927	0
12	Cadillac	SRX V8	USA	0	43523	44000
13	Cadillac	Tahoe LT	USA	41465	0	0
14	Chevrolet	Tahoe LT	USA	0	36287	35000
15	Chevrolet	Malibu 4dr	USA	18995	0	0
16	Chrysler	PT Cruiser 4dr	USA	17985	16919	0
17	Chrysler	Pacifica	USA	0	28725	28000
18	Dodge	Caravan SE	USA	0	20508	0
19	Dodge	Ram 1500 Regul..	USA	20215	0	20000
20	Ford	Escape XLS	USA	0	0	15670
21	Ford	Freestar SE	USA	26930	24498	0
22	GMC	Safari SLE	USA	0	23215	0
23	Honda	Pilot LX	Asia	27560	0	27000

We can do that by changing the specific variable name from MSRP to `_NUMERIC_`, this tells SAS that we want all numeric columns with missing values be substituted with the value 0.

We can also tell SAS to change the missing values to values other than 0s.

Code:

```
PROC STDIZE DATA=CARS3 REONLY MISSING=-1 OUT=CARS5 ;
    VAR _NUMERIC_ ;
RUN ;
```

Data set (Output):

	Make	Model	Origin	MSRP	Invoice	Sale
1	Acura	MDX	Asia	36945	-1	37000
2	Acura	TL 4dr	Asia	-1	21671	22000
3	Acura	NSX coupe 2dr...	Asia	89765	79978	-1
4	Audi	A4 3.0 4dr	Europe	-1	28846	29500
5	Audi	A6 3.0 4dr	Europe	36640	33129	35555
6	Audi	S4 Avant	Europe	49090	-1	50000
7	BMW	325xi 4dr	Europe	-1	-1	-1
8	BMW	530i 4dr	Europe	44995	41170	-1
9	BMW	325xi Sport	Europe	32485	30110	31050
10	Buick	Regal LS 4dr	USA	-1	22835	23050
11	Buick	Park Avenue Ultr...	USA	40720	36927	-1
12	Cadillac	SRX V8	USA	-1	43523	44000
13	Cadillac	Tahoe LT	USA	41465	-1	-1
14	Chevrolet	Tahoe LT	USA	-1	36287	35000
15	Chevrolet	Malibu 4dr	USA	18995	-1	-1
16	Chrysler	PT Cruiser 4dr	USA	17985	16919	-1
17	Chrysler	Pacifica	USA	-1	28725	28000
18	Dodge	Caravan SE	USA	-1	20508	-1
19	Dodge	Ram 1500 Regul...	USA	20215	-1	20000
20	Ford	Escape XLS	USA	-1	-1	15670
21	Ford	Freestar SE	USA	26930	24498	-1
22	GMC	Safari SLE	USA	-1	23215	-1
23	Honda	Pilot LX	Asia	27560	-1	27000

All we need to do is to change the `MISSING=0` option to the value that we want. In the example above, we changed all the missing values to a value of -1.

2. INPUT function – Use to convert character variables into numeric variables

For this example, we have information in the form of character which we want to be numeric information. This can be done using the `INPUT` function with an informat.

Code:

```
DATA NUMERIC ;
    NUM_STRING1 = '12345' ;
    NUM_STRING2 = '123' ;

    NUM1 = INPUT(NUM_STRING1, 8.);
    NUM2 = INPUT(NUM_STRING2, 8.);
RUN ;
```

Data set (Output):

	num_string1	num_string2	num1	num2
1	12345	123	12345	123

In the example above, we have two character strings, num_string1 and num_string2. We use the INPUT function with an 8. informat to tell SAS to “read” in num_string1 and convert it to a numeric string and assign that value to a new variable num1. We did the same thing to convert num_String2 to num2. The numeric informat of 8 tells SAS how to read in the form of the values that needs to be converted.

In addition to numeric string, we’ve also included an example on using the INPUT function to convert a character string of numbers to a date.

Code:

```
DATA DATES ;
    DATE_STRING1 = '20140508' ;
    DATE_STRING2 = '03082014' ;
    DATE_STRING3 = '21082014' ;

    DATE_NUMERIC1 = INPUT(DATE_STRING1 , YYMMDD10. ) ;
    DATE1 = INPUT(DATE_STRING1 , YYMMDD10. ) ;
    FORMAT DATE1 YYMMDD10. ;

    DATE_NUMERIC2 = INPUT(DATE_STRING2 , MMDDYY10. ) ;
    DATE2 = INPUT(DATE_STRING2 , MMDDYY10. ) ;
    FORMAT DATE2 MMDDYY10. ;

    DATE_NUMERIC3 = INPUT(DATE_STRING3 , DDMMYY10. ) ;
    DATE3 = INPUT(DATE_STRING3 , DDMMYY10. ) ;
    FORMAT DATE3 DDMMYY10. ;
```

RUN ;

Data set (Output):

date_String1	date_String2	date_String3	date_numeric1	date1	date_numeric2	date2	date_numeric3	date3
20140508	03082014	21082014	19851	2014-05-08	19790	03/08/2014	19956	21/08/2014

We have three different date strings in the data set above, date_string1, date_string2, and date_string3 respectively. Date_string1 is in the year, month, day informat; date_string2 is in the month, day, year informat while date_string3 is in the day, month, year informat.

Next, we used the INPUT function with the respective date informats to change the character information into numeric values, as shown in the output data set above, date_numeric1, date_numeric2, and date_numeric3, respectively. To make it easier to read the dates, we add a date format with the FORMAT statement and results can be seen in date1, date2, and date3.

3. PUT function – Use to convert numeric variables into character variables

Now that we know how to change character variables into numeric variables, it only makes sense to demonstrate how to convert character variables into numeric variables. In the example below, the variable “code” is in numeric format, and we wish to add leading zeros to it if the variable has less than five characters.

Code:

```
DATA NUMBERS ;
    INPUT CODE 5. ;
    DATALINES ;
34639
34234
93473
 5495
23748
 4857
  548
 4589
  45
 459
  345
43555
 3458
;
RUN ;

DATA NUMBERS2 ;
    SET NUMBERS ;
    CODE2 = PUT(CODE, z5.);
RUN ;
```

Data set (Output):

	CODE	CODE2
1	34639	34639
2	34234	34234
3	93473	93473
4	5495	05495
5	23748	23748
6	4857	04857
7	548	00548
8	4589	04589
9	45	00045
10	459	00459
11	345	00345
12	43555	43555
13	3458	03458

We have a list of code in numeric format in the input data set above that we would like to add leading zeros in front of it if the value has less than five characters. This can be done using the PUT function with a format. The format (2nd argument of the PUT function) tells SAS how we want the output to be presented.

As can be seen in the data set output above, we have the original "code" column in numeric format, after using the PUT function with a z5. format, we're able to convert the numeric column into a character string with leading zeros for variables that has less than five characters.

In addition, we can also use PUT function to convert a numerical value to a date character.

Code:

```
DATA DATES_C;  
  
    DATE1 = 19758;  
    DATE2 = 1;  
  
    CHAR_DATE1 = PUT (DATE1,DATE9.);  
    CHAR_DATE2 = PUT (DATE2,DATE9.);  
RUN;
```

Data set (Output):

	DATE1	DATE2	CHAR_DATE1	CHAR_DATE2
1	19758	1	04FEB2014	02JAN1960

In the code above, we have DATE1 and DATE2 in numeric format and we want to convert it into a date character string. Using a DATE9. format as the 2nd argument of the PUT function tells SAS the format that we want DATE1 and DATE2 to be formatted and assign them to CHAR_DATE1 and CHAR_DATE2 respectively.




4. LAG function – Can be used to access a previous numeric value

We have a list of S&P Indexes collected daily and we want to compare today's index value to the previous day's index value. We can use the LAG function to do this.

Code:

```
DATA SP500MAP1;  
    INPUT INDEXDATE DDMYY10. SP500;  
    DATALINES;  
01/01/2008      1468.4  
02/01/2008      1447.2  
03/01/2008      1447.2  
04/01/2008      1411.6  
05/01/2008      1411.6  
06/01/2008      1411.6  
07/01/2008      1416.2  
08/01/2008      1390.2  
09/01/2008      1409.1  
10/01/2008      1420.3  
11/01/2008      1401.0  
;  
RUN;  
  
DATA SP500MAP2;  
    SET SP500MAP1;  
    SP500_PREVIOUS = LAG(SP500);  
RUN;
```

Data set (Output):

	 indexdate	 sp500	 sp500_previous
1	01/01/2008	1468.4	.
2	01/02/2008	1447.2	1468.4
3	01/03/2008	1447.2	1447.2
4	01/04/2008	1411.6	1447.2
5	01/05/2008	1411.6	1411.6
6	01/06/2008	1411.6	1411.6
7	01/07/2008	1416.2	1411.6
8	01/08/2008	1390.2	1416.2
9	01/09/2008	1409.1	1390.2
10	01/10/2008	1420.3	1409.1
11	01/11/2008	1401	1420.3




In order to compare today's index values to the previous day's index value, we created a new column named sp500_previous, using the LAGn function. The LAG function in the example code can also be written as LAG1, which tells SAS to return one missing value and the values of sp500 lagged once.

Let's say we want to compare the index values five days apart, or even ten days apart, we can do this by changing LAG1 to LAG5 (for five days apart) or LAG10 (for ten days apart), as shown in the examples below.

Code:

```
DATA SP500MAP3;  
    SET SP500MAP1;  
    SP500_PREVIOUS = LAG5(SP500);  
RUN;
```

Data set (Output):

	 indexdate	 sp500	 sp500_previous
1	01/01/2008	1468.4	.
2	01/02/2008	1447.2	.
3	01/03/2008	1447.2	.
4	01/04/2008	1411.6	.
5	01/05/2008	1411.6	.
6	01/06/2008	1411.6	1468.4
7	01/07/2008	1416.2	1447.2
8	01/08/2008	1390.2	1447.2
9	01/09/2008	1409.1	1411.6
10	01/10/2008	1420.3	1411.6
11	01/11/2008	1401	1411.6

LAG5 in the example above tells SAS to return five missing values and the values of sp500 lagged five times.

Code:

```
DATA SP500MAP4;  
    SET SP500MAP1;  
    FORMAT INDEXDATE MMDDYY10.;  
    SP500_PREVIOUS = LAG10(SP500);  
RUN;
```


Data set (Output):

	indexdate	sp500	sp500_previous
1	01/01/2008	1468.4	.
2	01/02/2008	1447.2	.
3	01/03/2008	1447.2	.
4	01/04/2008	1411.6	.
5	01/05/2008	1411.6	.
6	01/06/2008	1411.6	.
7	01/07/2008	1416.2	.
8	01/08/2008	1390.2	.
9	01/09/2008	1409.1	.
10	01/10/2008	1420.3	.
11	01/11/2008	1401	1468.4

LAG10 in the example above tells SAS to return ten missing values and the values of sp500 lagged ten times.

5. SUBSTR function – Use to parse a text string

Let's say we have an ID column that can be split into two different columns. We can use the SUBSTR function to do that.

Code:

```
DATA TEXT_SPLIT;
    INPUT ID $9.;
    DATALINES;
234367ABC
314631XYZ
347834PDQ
348324ABC
343267PDQ
346327PDQ
234673AIL
734627XYZ
482137XYZ
346328XYZ
;
RUN;

DATA TEXT_SPLIT2;
    SET TEXT_SPLIT;
    ID_NO = SUBSTR(ID,1,6);
    ID_CODE = SUBSTR(ID,7,3);
RUN;
```

Data set (Output):

	△ ID	△ ID_NO	△ ID_CODE
1	234367ABC	234367	ABC
2	314631XYZ	314631	XYZ
3	347834PDQ	347834	PDQ
4	348324ABC	348324	ABC
5	343267PDQ	343267	PDQ
6	346327PDQ	346327	PDQ
7	234673AIL	234673	AIL
8	734627XYZ	734627	XYZ
9	482137XYZ	482137	XYZ
10	346328XYZ	346328	XYZ

In this example, we have a column called ID that can be split into two different columns, ID_no and ID_code. The first 6 characters of the ID column is the ID_no and the remaining 3 characters make up the ID_code. Here, we use two SUBSTR functions to split the ID up.

In the first SUBSTR function, we named the new column as ID_no, and tell SAS to pull the first character from the ID column and we want up to the sixth character of the ID column.

In the second SUBSTR function, we tell SAS to pull the seventh character from the ID column and up to three characters and assign it to the ID_Code column.

6. SCAN function – Searches a string for a defined substring of text

The SUBSTR function is a very useful function if we want to parse a text string. The SCAN function is another useful function for parsing text, especially if we have delimiters in it.

Code:

```
DATA ALL;
    INPUT VALGROUP $35.;
    DATALINES;
ABC 10-High
ABC 10-Low
ABC 6-High
ABC 6-Low
ABC Bonus Pro-High
ABC Bonus Pro-Low
ABC Bonus-High
ABC Bonus-Low
ABC Ultra-High
ABC Ultra-Low
XYZ 10-High
XYZ 10-Low
XYZ 5-High
XYZ 5-Low
XYZ 7-High
XYZ 7-Low
;
RUN;
```

```

DATA PROD_BAND;
  SET ALL;
  PRODUCT = SCAN(VALGROUP, 1, "-");
  BAND = SCAN(VALGROUP, -1, "-");
run;

```

Data set (Output):

	▲ VALGROUP	▲ PRODUCT	▲ BAND
1	ABC 10-High	ABC 10	High
2	ABC 10-Low	ABC 10	Low
3	ABC 6-High	ABC 6	High
4	ABC 6-Low	ABC 6	Low
5	ABC Bonus Pro-High	ABC Bonus Pro	High
6	ABC Bonus Pro-Low	ABC Bonus Pro	Low
7	ABC Bonus-High	ABC Bonus	High
8	ABC Bonus-Low	ABC Bonus	Low
9	ABC Ultra-High	ABC Ultra	High
10	ABC Ultra-Low	ABC Ultra	Low
11	XYZ 10-High	XYZ 10	High
12	XYZ 10-Low	XYZ 10	Low
13	XYZ 5-High	XYZ 5	High
14	XYZ 5-Low	XYZ 5	Low
15	XYZ 7-High	XYZ 7	High
16	XYZ 7-Low	XYZ 7	Low

In this example above, we have a column called valgroup which consists of the product name and whether it is a high band product or a low band product. Using the SCAN function, we are able to parse the information into two different types of information.

In the first SCAN function, we are asking SAS to extract the first word, as indicated by the 2nd argument of the function, from the variable valgroup until we see a '-' and assign that word to the column product.

Using a negative value in the 2nd argument of the function tells SAS to proceed to scan from right to left and extract the first word until we see a '-' and assign that word to the column band.

7. UPCASE, LOWCASE, PROPCASE function – Changes the case of the text variables into uppercase, lowercase and proper case respectively

Sometimes we get information that has inconsistent cases, especially if it is inputted manually. We can always standardize the variable by changing it to all uppercase, lower case, or proper case.

Code:

```
DATA CASE ;  
    INPUT STATE $15. ;  
    UPPER = UPCASE (STATE) ;  
    LOWER = LOWCASE (STATE) ;  
    PROPER = PROPCASE (STATE) ;  
    DATALINES ;  
NeW York  
cAlifornIA  
CHicago  
SEATTLE  
;  
RUN ;
```

Data set (Output):

	State	Upper	Lower	Proper
1	NeW York	NEW YORK	new york	New York
2	cAlifornIA	CALIFORNIA	california	California
3	CHicago	CHICAGO	chicago	Chicago
4	SEATTLE	SEATTLE	seattle	Seattle

The ability to standardize the case of a variable is very useful. It is difficult to subset or filter a variable if they differ by case. Using the UPCASE function, we are able to change all the text into upper case text; LOWCASE function changes the text into lower case text; and PROPCASE function changes the text into their appropriate cases.

Summary:

Data cleaning can be time consuming and frustrating. Having a few tricks up your sleeve could really expedite the process and make things easier.

Contact Information

Name: Audrey Yeo
Enterprise: Athene USA
Address: 7700 Mills Civic Parkway
City, State, ZIP: West Des Moines, IA 50316
Work Phone: 515-342-3759
E-mail: AYeo@Athene.com

Trademark Citations

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. Other brand and product names are trademarks of their respective companies.