Paper SA-06-2015

LET SAS® CLEANSE YOUR DIRTY DATA

Kaushal Raj Chaudhary[1], Deanna Naomi Schreiber-Gregory[2]

[1]Sanford Research, Sioux Falls, SD, [2] National University, La Jolla, CA

ABSTRACT

In an ideal world, every data set is complete, clean, and properly formatted. However, in real world situations, the data available to us is very rarely presented in this form. They may contain any number of problematic events such as outliers, duplicate observations, missing values, invalid character and numeric data values, as well as many other issues. Given the necessity that the data being examined is as complete and clean as possible, it is very important that these issues are addressed prior to any analysis. In this paper we describe various cases of dirty data and techniques to clean them. These techniques are explored within the context of SAS® 9.4 and presented in a way that would benefit beginning and moderate level SAS users.

INTRODUCTION

Clean data are essential for any analysis and SAS is a great tool for getting data cleaned. A wealth of functions and procedures are available for inspecting and cleaning data. This paper describes some of those techniques.

METHODS

In this section we will use DATA step, SAS procedures and /or SAS SQL to address the following scenarios.

- Variable Type Conversion
- Invalid Character Values
- Invalid Numeric Values and Outliers
- Duplicate Observations
- Missing Values

VARIABLE TYPE CONVERSION

IMPLICIT AND EXPLICIT CONVERSION

Sometimes numeric variables are stored as character variables. SAS automatically converts the character variable to numeric when used in numeric context (arithmetic operations) and prints the note in the log.

```
data have;
 input ID  Weight $ Height;
 cards;
 1 120   5.3
 2 130   5.5
 3 145   5.6
 ;
run;
data want;
  set have;
  Weight_num=Weight*1;
run;
```

```
NOTE: Character values have been converted to numeric
      values at the places given by: (Line):(Column).
      31:14
```

Converting the variable explicitly is better programming practice. This can be done using the INPUT and PUT functions. Automatic conversion can cause errors, and should be avoided to have a better, cleaner log. The log tells us when errors and problems occur. If we know about a problem and use a proper conversion, the log will not show us these known warnings, and unexpected warnings and errors will be easier to see.

Input function to convert character to numeric:
Num_var=input (oldvar, informat)

Put function to convert numeric to character:
Char_var=put (oldvar, format)

Character variable “Weight” is explicitly converted to numeric using the INPUT function with BEST3. informat. If “Weight” contains any non-numeric data SAS prints note (Invalid argument) and corresponding data in the log. To suppress both note and data from the log “??” modifier can be used in INPUT function. “Height” is converted to character using the PUT function and displayed in one decimal point.

```
data want;
  set have;
  Weight_num=input(Weight, Best3.);
  Height_char=put(Height, 3.1);
run;
data want;
  set have;
  Weight_num=input(Weight,?? Best3.);
  Height_char=put(Height, 3.1);
run;
```

INVALID CHARACTER VALUES

A data set might contain character values other than permitted values, such as other values than “M” and “F” for Sex. To detect invalid character values, we will use SAS procedures and a DATA step technique in this section.

SAS PROCEDURES

We illustrate PROC FREQ, PROC FORMAT, and PROC PRINT in finding invalid character values in a data set.

PROC FREQ is very handy data exploratory tool especially for categorical variables. The FREQ procedures below return the number of unique values of Sex and Race, all character variables and all variables of data, respectively. MISSING option in the table statement puts the frequency of missing in the table. We used an asterisk (*) sign to denote missing value in frequency table.

```
options missing='*';
proc freq data=have;
  tables  Sex Race /missing;
run;
options missing='*';
proc freq data=have;
  tables _character_/missing;
run;
proc freq data=have;
  tables _all_/missing;
run;
```

PROC FORMAT is one of the powerful tools in data manipulation. It uses VALUE and INVALUE statement to create user defined formats and informats respectively. In the example code we are creating character format $Sex and character informat $Sex_ to output and read the data in specified format, respectively. “F” and “M” are regarded as valid values for Gender. Informat reads “F” and “M” as they are using _SAME_ keyword and set to missing other values using OTHER keyword. PROC PRINT prints any values other than “M”,”F”, and missing.

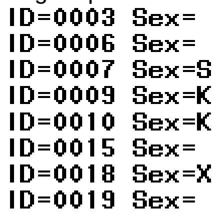
```
options missing='*';
proc format;
  value $Sex 'F','M' = 'Valid'
              ' '     = 'Missing'
             other = 'Not Valid';
proc freq data=have;
  format Sex $Sex.;
  tables Sex/ missing nopercent nocum;
run;
proc format;
  invalue $Sex_ 'F','M' = _same_
                   other = ' ';
data have;
input ID $ Sex $Sex_. Race $ Age Weight Height Smoking;
cards;
   0001 M WH 24 170 176.3 1
   0002 M WH 27 178 170.2 0
   ........................
   ........................
   ;
run;
proc print data=have;
  where Gender not in ('M' 'F' ' ');
  id ID;
  var Sex;
run;
```

DATA STEP

DATA _NULL_ can be used to return invalid character values of variable(s) specified. _NULL_ keyword tells SAS not to create a data set. PUT statement prints the invalid values in the log.

```
data _null_;
  set have;
  if Sex not in ("F", "M") then put ID= Sex=;
run;
```

Log output:



```
ID=0003 Sex=
ID=0006 Sex=
ID=0007 Sex=S
ID=0009 Sex=K
ID=0010 Sex=K
ID=0015 Sex=
ID=0018 Sex=X
ID=0019 Sex=
```

SAS has a number of character functions to clean and manipulate character data. The table below shows some commonly used character functions.

Table 1 Character functions

Function	Example	Output	Description

Length	`data _null_;`
	`len=length("MWSUG2015");`
	`  put len=;`
	`run;`
		len=9	returns the length of character string.

Upcase	`data _null_;`
	`var=upcase("mwsug2015");`
	`  put var=;`
	`run;`
		var=MWSUG2015	converts letters of character string to uppercase.

Substr	`data _null_;`
	`var=substr("MWSUG2015",1,5);`
	`   put var=;`
	`run;`
		var=MWSUG	extracts part of a character string.

Scan	`data _null_;`
	`  var="MWSUG2014,MWSUG2015";`
	`  newvar=scan(var,2,',');`
	`  put newvar=;`
	`run;`
		newvar=MWSUG2015	returns nth word from a character string.

Compress	`data _null_;`
	`  var=compress("M W S U G");`
	`  put var=;`
	`run;`
		var=MWSUG	returns a character string with specified characters removed from the original string

Compbl	`data _null_;`
	`  var=compbl("MWSUG  2015");`
	`  put var=;`
	`run;`
		var=MWSUG 2015	removes multiple blanks from a character string.

Tranwrd	`data _null_;`
	`var=tranwrd("MWSUG2014","2014","2015");`
	`  put var=;`
	`run;`
		var=MWSUG2015	replaces all occurrences of a substring in a character string

Translate	`data _null_;`
	`var=translate("MWSUG2014","2015","2014");`
	`  put var=;`
	`run;`
		var=MWSUG2015	replaces specific characters in a character string.

Index	`data _null_;`
	`  pos=index("MWSUG2015","SUG");`
	`  put pos;`
	`run;`
		3	searches a character expression for a string of characters, and returns the position of the string's first character for the first occurrence of the string.

Verify	`data _null_;`
	`  check=verify("ABCD","ABCD");`
	`  put check=;`
	`run;`
		check=0	returns the position of the first character in a string that is not in any of several other strings.

Anyalpha	`data _null_;`
	`  var=anyalpha("MWSUG2015");`
	`  put var=;`
	`run;`
		var=1	searches a character string for an alphabetic character, and returns the first position at which the character is found.

Anyalnum	`data _null_;`
	`  var=anyalnum("MWSUG2015");`
	`  put var=;`
	`run;`
		var=1	searches a character string for an alphanumeric character, and returns the first position at which the character is found.

Anydigit	`data _null_;`
	`  var=anydigit("MWSUG2015");`
	`  put var=;`
	`run;`
		var=6	searches a character string for a digit, and returns the first position at which the digit is found.

Notalpha	`data _null_;`
	`  var=notalpha("MWSUG2015");`
	`  put var=;`
	`run;`
		var=6	searches a character string for a nonalphabetic character, and returns the first position at which the character is found.

Notdigit	`data _null_;`
	`  var=notdigit("MWSUG2015");`
	`  put var=;`
	`run;`
		var=1	Searches a character string for any character that is not a digit, and returns the first position at which that character is found.

Anypunct	`data _null_;`
	`  var=anypunct("MWSUG2015!");`
	`  put var=;`
	`run;`
		var=10	Searches a character string for a punctuation character, and returns the first position at which that character is found.

INVALID NUMERIC VALUES AND OUTLIERS

A data set might contain numeric values out of permitted ranges and outliers. Hawkins (1980) defines an outlier as “an observation that deviates so much from other observations as to arouse the suspicion that it was generated by a different mechanism. In general, any numeric values out of lower quartile (Q1) -1.5 IQR (Interquartile range) and upper quartile (Q3) +1.5 IQR or out of 5% and 95 % percentiles are considered as outliers. These invalid numeric values or outliers will give incorrect result during analysis. SAS procedures and DATA step techniques are presented to capture those values.

SAS PROCEDURES

PROC MEANS and PROC UNIVARIATE can be used to check invalid numeric values and outliers.
PROC MEANS produces summary statistics for numeric variables. The default summary statistics are N, MEAN, STD, MIN, and MAX. To get summary statistics other than those, such as first quantile (q1) or third quantile (q3), it has to be specified in PROC MEANS statement. MAXDEC option limits the decimal points in summary statistics values.

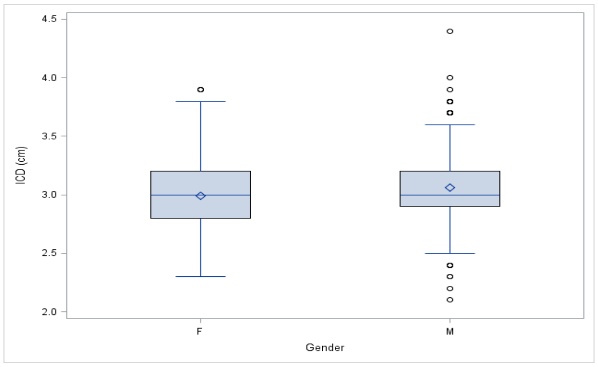
PROC UNIVARIATE yields various information about numeric variables. It also produces graphs, such as histogram, box plot, and normal probability plot if PLOTS option is included. ODS SELECT statement outputs only the desired table among several other tables. In our example we select only extreme observations table. The default quantiles in PROC UNIVARIATE are 1st, 5th, 10th, 25th, 50th, 75th, 90th, 95th, and 99th. To get specific quantiles we can use PCTLPTS in OUTPUT statement. In example below we obtain every tenth percentile from 10 to 100 of “TRIGLYCERIDE”.

```
proc means data=trig n min max mean median q1 q3 maxdec=2;
  var triglyceride;
run;
ods select extremeobs;
proc univariate data=trig;
  var triglyceride;
run;
Proc univariate data=trig;
  Var triglyceride;
  Output out=P pctlpre=P_ pctlpts= 10 to 100 by 10;
Run;
```

UNIVARIATE BOX PLOT

Box plot provides graphic summary of numeric variables. It displays median, first quartile, third quartile, inter quartile range and any outliers if exist. PROC UNIVARIATE, PROC GPLOT, PROC BOXPLOT, and PROC SGPLOT can produce box plot in SAS. I prefer SGPLOT procedure because of easier syntax. In our example “male” seems to have more outliers than “female” in “ICD” variable.

```
proc sgplot data=gad.final;
  vbox ICD/category=Gender;
run
```



PROC RANK

PROC RANK computes ranks for one or more numeric variables across the observations of a SAS data set and outputs the ranks to a new SAS data set. Groups option generates quantiles (groups=4), deciles (groups=10), and percentiles (groups=100). In example below new variable “ICD_Decile” with deciles values is added to output dataset “deciles”.

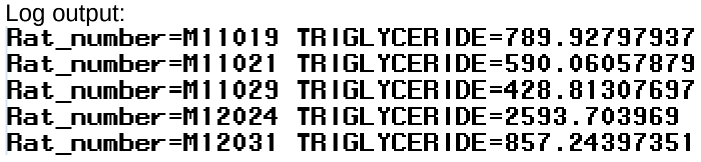
```
proc rank data=gad.final2 groups=10 out=deciles
           (keep=ID ICD ICD_Decile);
  var ICD;
  ranks ICD_Decile;
run;
```

DATA STEP

This technique is similar to detecting invalid character values. “TRIGLYCERIDE” values more than 400 have been printed in the log.

```
data _null_;
  set trig;
  if triglyceride > 400 then put Rat_Number=
triglyceride=;
run;
```

Log output:



```
Rat_number=M11019 TRIGLYCERIDE=789.92797937
Rat_number=M11021 TRIGLYCERIDE=590.06057879
Rat_number=M11029 TRIGLYCERIDE=428.81307697
Rat_number=M12024 TRIGLYCERIDE=2593.703969
Rat_number=M12031 TRIGLYCERIDE=857.24397351
```

SAS date and time are special type numeric values in SAS. SAS has several date and time functions to clean and manipulate date and time. The following table presents frequently used date and time functions.

Table 2 SAS Date and Time functions

Function	Example	Output	Description

date	`data _null_;`
	`  Today_date=date();`
	`  put Today_date mmddyy8.;`
	`run;`
		09/01/15	returns today's date as a SAS date value.

Today	`data _null_;`
	`  Today_date=Today();`
	`  Put Today_date mmddyy10.;`
	`  run;`
		09/01/2015	returns the current date as a SAS date value.

Datatime	`data _null_;`
	`  Date_time=DateTime();`
	`  put Date_time Datetime13.;`
	`run;`
		01SEP15:09:46	returns the current date and time of day as a SAS datetime value.

datepart	`data _null_;`
	`  Date_time='01SEP2015:9:30'dt;`
	`  Date_Part=Datepart(Date_time);`
	`  Put Date_part worddate18.;`
	`  run;`
		September 1, 2015	returns the date part of a SAS datetime value as a date value

timepart	`data _null_;`
	`  Date_time='01SEP2015:9:30'dt;`
	`  Time_Part=Timepart(Date_time);`
	`  put Time_part Time8.;`
	`run;`
		9:30:00	returns the time part of a SAS datetime value.

day	`data _null_;`
	` Day_=Day('01SEP2015'd);`
	` put Day_;`
	`run;`
		1	returns the day of the month from a SAS date value.

weekday	`data _null_;`
	` Weekday_=Weekday('01SEP2015'd);`
	`  put Weekday_;`
	`run;`
		3	returns the day of the week from a SAS date value

month	`data _null_;`
	` Month_=Month('01SEP2015'd);`
	` put Month_;`
	`run;`
		9	returns the numerical value for the month of the year from a SAS date value

Year	`data _null_;`
	`  Year_=Year('01SEP2015'd);`
	`  put Year_;`
	`run;`
		2015	returns the year from a SAS date value.

qtr	`data _null_;`
	`  qtr_=qtr('01SEP2015'd);`
	`  put qtr_;`
	`run;`
		3	returns the quarter of the year from a SAS date value.

mdy	`data _null_;`
	`  mdy_=mdy(9,1,2015);`
	`  put mdy_ worddate12.;`
	`run;`
		Sep 1, 2015	returns a SAS date value for month, day, and year values.

dhms	`data _null_;`
	`DHMS_=DHMS('01SEP2015'd,10,20,45);`
	`  put DHMS_ datetime.;`
	`run;`
		01SEP15:10:20:45	returns a SAS datetime value for date, hour, minute, and second values.

hms	`data _null_;`
	`  HMS_=HMS(10,20,45);`
	`  put HMS_ time.;`
	`run;`
		10:20:45	returns a SAS time value for hour, minute, and second values

Hour
Minute
Second
	`data _null_;`
	`  time_='11:20:30't;`
	`  hour_=hour(time_);`
	`  minute_=minute(time_);`
	`  Second_=second(time_);`
	`  put time_  time.  hour_ minute_`
	`second_;`
	`run;`
		11:20:30
		11
		20
		30
			returns the hour from a SAS datetime or time value.
			returns the minute from a SAS time or datetime value.
			returns the second from a SAS time or datetime value.

intck	`data _null_;`
	`  Birthday='01AUG1993'd;`
	`Age=intck('Year',Birthday,Today());`
	`  put Age;`
	`run;`
		22	returns the number of boundaries of intervals of the given kind that lie between the two date or datetime values.

intnx	`data _null_;`
	`  Birthday='01AUG1993'd;`
	`date_=intnx('Year',Birthday,22,'same');`
	`  put date_ date9.;`
	`run;`
		01AUG2015	returns the date or datetime value of the beginning of the interval that is *n* intervals from the interval that contains the given date or datetime value.

DUPLICATE OBSERVATIONS

Sometimes the observations are duplicated and we have to remove those duplicates.

SAS PROCEDURES

PROC SORT

NODUPKEY and NODUP or NODUPRECS options in PROC SORT can be used to find and remove the duplicates in the SAS data set. NODUP or NODUPRECS option removes all the exact observations matched by all variables. It compares the all variable values of current observation to values of previous observation. If it exists, then it is not written to output data set. However, if identical observations are not consecutive it fails to remove the duplicates. The only solution of this problem is to sort the input data set by all variables. NODUPKEY options exclude the duplicate observations matched by variable(s) in by statement. It compares observations by variable in by statement. DUPOUT option creates output dataset with duplicate observations.

```
proc sort data=have out=want nodup;
  by ID;
run;
proc sort data=have out=want nodupkey;
  by ID;
run;
proc sort data=have out=want noduprecs;
  by ID;
run;
proc sort data=have out=want dupout=duplicates nodup;
  by ID;
run;
```

PROC FREQ

PROC FREQ is another way of identifying duplicates in a data set. Here we are creating an output data set “want” in FREQ procedure and subsetting it by “count” variable to include more than 1 count. To get exact observations by all variables in a data set, they all have to be present in the table statement separated by an asterisk (*) sign. Unfortunately, _ALL_ key word cannot be used here to specify all variables in the data set.

```
proc freq data=dp;
  tables ID /out=want (where= (count > 1));
run;
```

PROC SQL

PROC SQL is yet another technique for getting duplicate observations form an input data set. In this example we are creating variable Count which contains number of identical ID in our data set. All columns of table have to be included in group by statement to identify the exact matches.

```
proc sql;
  select *, count(*) as Count
    from dp
    group by ID
    having count(*) > 1;
run;
```

DATA STEP

First.variable and last.variable in data step can be used to detect the duplicate observations in a data set. It consists of two steps -first sorting the input data set by key variable(s) and second creating output data set with duplicates. In this example we are outputting only those IDs where both first.ID and last.ID are not equal to 1 (ie. duplicate IDs).

```
proc sort data=dp out=sorted_dp;
  by ID;
run;
data want;
  set sorted_dp;
  by ID;
  if not(first.ID and last.ID) then output;
run;
```

MISSING VALUES

A data set might contain missing values. They can be of following types in SAS.

- . (Numeric)
- ‘ ‘ (Character)
- .letter (Special such as .Z) (Numeric)
- ._ (Special) (Numeric)

SAS sets variables to missing at the top of each implicit loop of the DATA step except for variables in the retain statement, variable created in the sum statement, _temporary_arrays, automatic variables, and variable from SAS data sets. An arithmetic operation on a missing value generates missing value. In the example below the first observation of variable “First_total” will be missing. Any further calculation on that result will be missing too. To prevent such propagation of the missing value, computation is performed using sample statistics functions (SUM function in example below).

```
data have;
input x y z;
cards;
2 . 4
4 3 2
;
data want;
  set have;
  First_total=x+y+z;
  Second_total=sum(of x--z);
run;
```

In this section we will explain various techniques in SAS to assess the missing values in a data set.

SAS PROCEDURES

The MISSING option in the PROC FREQ and PROC MEANS returns the number of missing values

```
options missing='*';
proc freq data=have;
  tables _character_/missing nopercent nocum;
run;
proc means data=have n nmiss;
  var _numeric_;
run;
proc mi data=have;
  ods select misspattern;
run;
```

The MISSING option in the table statement of PROC FREQ returns the frequency of missing values of variables in table statement. Similarly, the NMISS option in PROC MEANS gives the number of missing values for each variable included in var statement. We are using _NUMERIC_ keyword to get the frequency of missing of all numeric variables. PROC MI will output missing data patterns for the variables in the specified datasets.

DATA STEP FUNCTIONS

SAS has various functions to check the missing values in numeric and character variables.
IS NULL or IS MISSING function in SAS is ANSI Standard method of SQL for evaluating missing values. They work with character or numeric variables and can be used in WHERE clause of DATA statement or PROC SQL.

```
data want;
  set have;
  where Race is null;
run;
data want;
  set have;
  where Race is missing;
run;
data want;
  set have;
  where Sex is not missing;
run;
proc sql;
  create table test as
  select *
  from have
  where Sex is not missing;
quit;
```

MISSING function checks whether a character or numeric variable is missing and returns. It returns 1 if the variable is missing or 0 if it is not missing. CMISS and NMISS count the number of missing across observations. CMISS counts the missing observations for both character and numeric variable(s) whereas NMISS counts only for numeric variable(s). CALL MISSING assigns missing values to its argument (s) (character or numeric variables).

```
data want;
  set have;
  miss_n=cmiss(of Sex--Smoking);
  miss_num=nmiss(Age,Weight,Height);
  miss_y_n=missing(Age);
proc print data=want;
run;
data want;
  set have;
  call missing (of Sex--Smoking);
run;
```

CONCLUSION

This paper presented various techniques of inspecting and cleaning data.

REFERENCES

Philpot, L. B., and Cantu, G. (2012, November). *Dirty Data? Clean it up with SAS.* Paper presented at SCSUG 2012. Houston, TX.

Kincheloe, F. (2008, March). *Get Your Hands Dirty Cleaning Your Data with SAS Data Quality Server.* Presented at SAS Global Forum 2008. San Antonio, TX.

Cody, R. (2008). Cody's Data Cleaning Techniques Using SAS, Second Edition. Cary, NC: SAS® Institute Inc.

Field, A., & Miles, J. (2012). *Discovering Statistics Using SAS®* , Thousand Oaks, CA: Sage Publications.

SAS® Institute Inc. 2008. SAS /STAT® 9.2 *User's Guide*. Cary, NC: SAS® Institute Inc.

Horstman, J. & Muller, R. (2011). Dealing with Duplicates in Your Data. MWSUG 2011, Kansas City, KS.

CONTACT INFORMATION

Your comments, questions, and suggestions are valued and encouraged. Contact the authors at:

Kaushal Raj Chaudhary
Sanford Research
Sioux Falls, SD
Email: kaushal.chaudhary@SanfordHealth.org

Deanna Naomi Schreiber-Gregory
National University
La Jolla, CA / Moorhead, MN
E-mail: d.n.schreibergregory@gmail.com