

Automating Preliminary Data Cleaning in SAS®

Alec Zhixiao Lin, Loan Depot, Foothill Ranch, CA

ABSTRACT

Preliminary data cleaning or scrubbing tries to delete the following types of variables considered to be of little or no use: 1) variables with missing values or a uniform value across all records; 2) variables with very low coverages; 3) character variables for name, addresses and IPs. These variables are very commonly seen in big data. This paper introduces a SAS process that will automatically and efficiently detect these variables with a minimal manual handling from users. The output also helps users to identify those character variables that need to be converted to numeric values for downstream analytics.

INTRODUCTION

Variables of the following types are considered to be of little or no use in analytics:

- A numeric or character variable with missing values across all records
- A numeric or character variable with a uniform value across all records
- A numeric or character variable with a very low coverage (for example, < 0.5%)
- A character variable with too many categories. Name, address and email are common examples.

Cleaning the variables of the above types will reduce the file size, improve data quality and expedite downstream analyses. Users often manually run multiple procedures in SAS and review output summaries to find potential variables for deletion. When one works with large data with numerous variables or intends to build multiple models for different segments, this exercise can become quite repetitive and onerous. This paper introduces SAS process that will effectively save users from much of the burden in preliminary data cleaning or scrubbing.

THE LOGIC FOR PRELIMINARY CLEANING

Let's strike a note of realism first. Preliminary data cleaning robotically examines the value expressions of all fields in the raw SAS file to detect possible variables of the aforementioned types that have little or no use. The machine does not – neither is it able to – interpret the business meanings of variables before making recommendations for deletion. Therefore, users still need to review the output list to decide which variables to exclude from further analyses.

For numeric variables, users can run the following PROC MEANS procedure and use the summary to detect variables of little or no use preliminarily:

```
proc means data=sasdataset MIN MAX N; run;
```

The following is how one can detect numeric or data-and-time-related variables of little or no use:

- N=0 suggests all records contain null (or missing) values.
- MIN=MAX suggests a uniform value across all records. Please bear in mind that this logic does not extend to a variable with missing values for some records and a non-missing uniform value for other records. For example, if a variable called "approved" has 45% of records showing missing values while 55% of other records showing a value of 1 for "approved", it will not be recommended for deletion because the missing value could mean 0 for "unapproved". The process introduced in the paper has also considered this case.
- When N/total_number_of records is extremely small, it suggests a very low coverage by the variable. Users can decide whether they would like to retain or drop these variables.

PROC MEANS does not apply to character variables, but one can manually run the following code and apply a similar set of logic for examining the data:

```
proc sql;
create table count_all_variables as
select count(X1) as X1_count, count(distinct X1) as distinct_X1_count,
```

```

count(X2) as X2_count, count(distinct X2) as distinct_X2_count,
.....
count(Xn) as Xn_count, count(distinct Xn) as distinct_Xn_count
from sasdataset; quit;

```

For character variables, the following is how one can detect those of little or no use:

- When `Xi_count=0`, it suggests all observations contain null (or missing) values;
- When `Xi_count=total_number_of_records` and `distinct_Xi_count=1`, it means the variable has uniform value across all records. Also, bear in mind that this logic does not extend to a character variable with invalid values for some records and a valid uniform value for other records. For example, if a variable called `having_score_or_not` has 25% of records showing null values while 75% of other records showing a flag "YES", it will not be recommended for deletion because the null value could mean "NO". We have considered this case in designing the process.
- When `Xi_count/total_number_of_records` is extremely small, it suggests a very low coverage by the variable. Users can decide later whether they would like to retain or drop these variables.
- A very high `distinct_Xi_count` suggests numerous flags for a variable, such as name and address. Numeric values or date-and-time strings expressed in character should be retained and converted accordingly.

For a very limited number of variables, the above measures can be applied manually with a relative ease. When a data set contains hundreds or thousands of variables, one usually does not have time to screen through each variable. We would like to find a more efficient way rather than typing, copying and pasting these lines of codes.

THE SAS PROGRAM

In this paper, we introduce a SAS process that will automatically do the following in a sweep:

- Identify numeric variables and character variables with minimal manual handling from the user. Date-and-time-related variables are treated the same as numeric variables..
- Examine numeric and character variables by applying the aforementioned logic. The process is equally applicable to a data set containing numeric variables only or character variables only.
- At the end of the program, SAS generates a list of variables recommended for deletion, along with variable type (NUM, DATETIME or CHAR) and reason for recommendation for each variable.

Users only need to make minimal changes to several macro values at the beginning of the SAS program in order to use it.

```

libname my_lib "/opt/sas/Tablespace/alin/ARS/data";      /* your libname */
%let sasdata=subsamp;                                  /* SAS data set */
%let cvthresh=0.005;                                   /* threshold for defining a low coverage. */
%let dsthresh=0.3;                                     /* for examining variables with too many flags. */
%let nodelete=csp_acct_id customer_id;                 /* merging keys and IDs to keep */
** no need to change any code below;

```

(See the rest of the program in the appendix.)

The following are explanations for some macro variables:

`cvthresh`: the threshold for defining a low coverage. For example, 0.005 means that variables with coverage less than 0.5% will be recommended for deleting. If one would like to retain all variables regardless of their coverages, set `cvthresh=1`.

`dsthresh`: number of distinct values or flags for a character variables. Typical examples are names, addresses, IP addresses, etc.

`nodelete`: variables you definitely would like to retain such as:

- Target variable (y) for analysis or for modeling
- record ID, account number and other merging keys

If you cannot determine which variables to retain, leave it blank as nodelete= now and decide later after reviewing the output.

The SAS program lists the following text strings as common expressions for invalid values:

' ', '#DIV/0!', '#DIV/0', '#N/A', '#NAME?', '#NULL!', '#NUM!', '#REF!', '#VALUE!', 'NAN', 'INF', 'NULL', 'NA', 'N/A'

User can list more expressions for additional exclusions.

THE SAS OUTPUT

At the end of the program, a list of numeric and character variables will be assembled for review. The following is an example:

Obs	variable	variable_type	reason_for_cleaning
1	ACQFICO201311	NUM	coverage too low (0.001319503)
2	AUTH_SEG	NUM	uniform value
3	BALANCE201311	NUM	coverage too low (0.001319503)
4	CRLINE201311	NUM	coverage too low (0.001319503)
5	ObsNum	NUM	uniform value
6	STMT_DT201311	DATETIME	all values null
7	approved	NUM	uniform value
8	extractdate	NUM	all values null
9	have_score	NUM	uniform value
10	sum_rstn_All_cnt_6mth	NUM	coverage too low (0.009877946)
11	CBNOHIT	CHAR	uniform value
12	ORG_ID201311	CHAR	coverage too low (0.001319503)
13	PAYHIST201311	CHAR	all values null
14	datasource	CHAR	uniform value
15	transaction_id	CHAR	too many flags (54566)
16	FICO_txt	CHAR	too many flags (213)

A fraction within brackets suggests the low percentage of coverage by a variable. An integer within brackets is the number of distinct flags or indicators a character variable has.

One should not rush to put the list of variables recommended for deletion into a DROP statement in a DATA step. SAS has identified these “suspects” merely based on their expressions rather than on their actual business meanings. I suggest the following practices for rescuing useful fields:

- Variables that are definitely worth retaining even though its expressions or values cannot be scaled or quantified. For example, transaction_id will be used for merging with other files and should be retained.
- Some character variables flagged as “too many flags” could be numeric values or date-and-time strings expressed in text. For example, the variable FICO_txt suggests 213 distinct values for FICO score across all records. This variable should be retained and converted to a numeric one.

After carefully reviewing all variables suggested by the summary list, one can apply a DROP statement in a DATA step to delete those unneeded ones. With a more compact-looking data set at hand, users can go on to apply various statistical methods to examine retained variables.

CONCLUSION

This paper introduces a process in SAS that can efficiently and effectively detect some variables of little or no use for preliminary data cleaning. We hope the suggested practice will expedite data examination for analysis and for modeling.

ACKNOWLEDGEMENTS

I would like to thank the team in Loan Depot led by Shawn Benner for their continuous support for this endeavor.

CONTACT INFORMATION

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

Alec Zhixiao Lin

VP of Modeling

Loan Depot

26642 Towne Center Drive

Foothill Ranch, CA 92610

Email: alecindc@gmail.com

Web: www.linkedin.com/pub/alec-zhixiao-lin/25/708/261/

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 registered trademarks or trademarks of their respective companies.

APPENDIX

```
libname my_lib "/opt/sas/Tablespace/alin/ARS/data";          /* your libname */
%let sasdata=subsamp;                                       /* SAS data set */
%let cvthresh=0.001;                                       /* threshold for defining a low coverage. */
%let dsthresh=0.3;                                         /* for examining variables with too many flags. */
%let nodelete=csp_acct_id customer_id; /* merging keys and IDs to keep */
** no need to change any code below;

PROC CONTENTS DATA=my_lib.&sasdata(drop=&nodelete) MEMTYPE=data OUT=varlist NOPRINT;
data subchar subnum;
length name $ 32.;
set varlist(keep=name type format);
if upcase(format)="DATETIME" then variable_type="DATETIME";
else variable_type="NUM";
if type=2 then output subchar;
else output subnum; run;

proc sql noprint; select count(type) into :subnumct from subnum; quit;
proc sql noprint; select count(type) into :subtxtct from subchar; quit;

%macro charnum;
%if &subnumct=0 %then %do;
proc delete data=subnum; run; %end;
%else %if &subtxtct=0 %then %do;
proc delete data=subchar; run; %end;
%mend;
%charnum;

%macro chknum;
%IF %SYSFUNC(EXIST(subnum))=1 %THEN %DO;
proc sql noprint; select count(*) into :totalcnt from my_lib.&sasdata; quit;
proc sort data=subnum; by name; run;

proc means data=my_lib.&sasdata min max n nway noprint;
output out=check_mean(drop=_type_ _freq_); run;

proc transpose data=check_mean out=trans_means; run;
data num_delete(keep=_name_ reason_for_cleaning);
length _name_ $ 32.;
length reason_for_cleaning $ 32.;
set trans_means;
if col1 in (0, .) then do; reason_for_cleaning="all values null"; clean=1; end;
else if col2=col3 and col1=&totalcnt then do; reason_for_cleaning="uniform value";
clean=1; end;
else if col1/&totalcnt le &cvthresh then do; cvg=col1/&totalcnt;
format cvg percent5.4;
informat cvg percent5.4;
reason_for_cleaning=("coverage too low ("||cvg||")"); clean=1; end;
if clean=1; run;

proc sort data=num_delete; by _name_; run;
```

```

proc sort data=subnum; by name; run;

data num_delete2(rename=(name=variable));
merge num_delete(in=t) subnum(rename=(name=name_));
by name_; if t; run; %end;
%else %do; data num_delete; set _null_;
name_=' '; reason_for_cleaning=' '; %end;
%mend;
%chknum;

%macro chkchar;
%IF %SYSFUNC(EXIST(subchar))=1 %THEN %DO;
data subchar2; set subchar; seqnum+1; varid=compress("x"||seqnum); run;
proc sql noprint;
create table varcount as select count(*) as vcnt from subchar2; quit;
data _null_; set varcount; call symputx('varcount', vcnt); run;
proc sql; select name into :x1-x&varcount from subchar; quit;

%macro countvar;
proc sql noprint; select count(*) into :totalcnt from my_lib.&sasdata; quit;
%do i=1 %to &varcount;
proc sql;
create table varcount&i as
select "&x&i" as variable,
count(&x&i) as obscnt,
count(&x&i)/&totalcnt as obscntpct,
count(distinct &x&i) as distinct_obsCnt,
count(distinct &x&i)/&totalcnt as distinctpct
from my_lib.&sasdata
where upcase(&x&i) not in (' ', '#DIV/0!', '#DIV/0', '#N/A', '#NAME?', '#NULL!',
'#NUM!', '#REF!', '#VALUE!', 'NAN', 'INF', 'NULL', 'NA', 'N/A'); quit;
%end;
%mend;
%countvar;

data char_delete;
length variable $ 32.;
length reason_for_cleaning $ 32.;
set varcount1-varcount&varcount;
if obscnt in (0, .) then reason_for_cleaning="all values null";
else if distinct_obsCnt=1 and obscntpct=1
then reason_for_cleaning="uniform value";
else if obscntpct < &cvthresh
then reason_for_cleaning="coverage too low ("||obsCntpct||)";
else if distinctpct > &dsthresh
then reason_for_cleaning="too many flags ("||distinct_obsCnt||)";
if reason_for_cleaning ne ' '; run;
%end; %else %do;
data char_delete; set _null_; variable=' '; reason_for_cleaning=' '; run;
%end;
%mend;
%chkchar;

data variables_to_clean;
retain variable variable_type reason_for_cleaning;
set num_delete2(in=s keep=variable variable_type reason_for_cleaning)
char_delete(in=t keep=variable reason_for_cleaning);
if t then variable_type='CHAR';
if variable ne ' '; run;
proc print data=variables_to_clean; run;

```