SAS®
GLOBAL FORUM
2020
MARCH 29 - APRIL 1
WASHINGTON, DC
Abstract:

In the DATA step, merging data sets with common variables that are not included as BY variables can yield undesirable results. Specifically, the value of a common variable can be overwritten with an incorrect value. To prevent this from happening, you must ensure that the variable is read from only one "master" data set, by either dropping or renaming the variable in the other data sets. When working with data sets with just a few variables, you can quickly check which variables appear in more than one data set. However, as the number of data sets and variables increases, the chance of missing a common variable also increases. The SAS® macro CHECK_VAR_EXIST was written to identify variables that exist in more than one data set more efficiently and accurately. The macro prints all common variables, which data sets they appear in, and other pertinent information. You can then use the list to drop or rename variables where they are not relevant, thereby reducing the chance of unintentionally overwriting a large number of variables.
Introduction:

- SAS programmers are commonly taught that when you merge datasets in the DATA step, variables in the dataset listed later on the MERGE statement replace the values of variables that also exist in a previously listed dataset.
- This may be true for one-to-one merging, but not for one-to-many merging, because of how the Program Data Vector works.
- As such, you need to be careful when combining multiple datasets that have variables in common, and not all of them are included as BY variables.
- The best way to avoid seeing unexpected results is to drop or rename common variables so that they only show up in one dataset.
- Figuring out the common variables can be done easily if you’re working with just a couple of datasets with few variables. However, it gets more cumbersome the more datasets and variables are involved.
- The SAS® macro `CHECK_VAR_EXIST`, which will be described in the next slides, provides an automated way of identifying common variables.
DON'T OVERWRITE ME! A SAS® MACRO TO IDENTIFY VARIABLES THAT EXIST IN MORE THAN ONE DATA SET

SAS® Macro

**CHECK_VAR_EXIST:**
- Identifies variables that exist in more than one dataset.
- Ideal to use before merging 2+ datasets as a check to prevent incorrect variables from overwriting correct ones with the same name.
- Input parameters: DTA is a list of datasets to check (preceded by a libref if stored as a permanent dataset), LINK_VAR is a list of variables that should be excluded from the checking (usually the ones used as BY variables in the MERGE statement).
- Output: list of variables that appear in more than one dataset, with additional info like length & type, in the Results Window.

```sas
%macro check_var_exist(dta=,link_var=);
  data _null_;  
  /*remove excess blank characters from list of datasets*/
  _var="&dta";
  dta_list=tranwrd(compbl(strip(_var))," ",".");
  call symputx("dta_list",dta_list);

  /*count how many datasets to check for overlapping variables*/
  cnt_dta=count(strip(dta_list)) + 1;
  call symputx("cnt_dta",cnt_dta);

  /*list of variables to exclude from checking*/
  list_var=lowcase("'|| tranwrd(compbl(strip("&link_var"))," ",","','")||'";
  call symputx("list_var",list_var);
  run;

  %put &dta_list &cnt_dta &list_var;

  /*output variables that exist in more than 1 dataset*/
  proc sql;
    select *
    from (select distinct upcase(name) as name label="Column Name",
                    type,length,libname,memname
                    from sashelp.vcolumn
                    %if %sysfunc(find(%scan(%scan(lowcase(&dta_list)),1,' '),1,'.'))>0 %then %do;
                      where ( (lowcase(libname)=%scan(%sysfunc(lowcase(&dta_list)),1,' ',1,'.'))
                          and lowcase(memname)=%scan(%sysfunc(lowcase(&dta_list)),1,' ',2,'.'))
                      %end;
                    %else %do;
                      or (lowcase(libname)=work and lowcase(memname)=%scan(%sysfunc(lowcase(&dta_list)),1,' '))
                      %end;
                  %do i=2 %to %sysfunc(find(%sysfunc(lowcase(&dta_list)),41,' ','.'))>0 %then %do;
                    %if %sysfunc(find(%scan(%sysfunc(lowcase(&dta_list)),41,' ','.'))>0 %then %do;
                      or (lowcase(libname)=%scan(%sysfunc(lowcase(&dta_list)),41,' ',1,'.'))
                      %end;
                    %else %do;
                      or (lowcase(libname)=work and lowcase(memname)=%scan(%sysfunc(lowcase(&dta_list)),41,' '))
                      %end;
                  %end;
                %and lowcase(name) not in (&list_var)
                )
              group by name
              having count(*)>1
              order by name,libname,memname
          ;
  quit;
%mend check_var_exist;
```
Results:

- To illustrate how the macro can be used, we downloaded a few CSV files from Data.Medicare.gov and imported into SAS.
- **Data.Medicare.gov** is a website where consumers can freely download official healthcare-related data produced by the Centers for Medicare & Medicaid Services (CMS).
- We checked 5 datasets, 3 of which are temporary and 2 are permanent datasets, for common variables. As we’re interested in merging all 5 datasets by the variable, **Provider_ID**, we exclude this from the check.

```sas
%check_var_exist(dta = Hospital_general_information 
                  Fy_2019_ipps_fr_impact_file 
                  sasgf.Complications_and_deaths___hosp 
                  Healthcare_associated_infections 
                  sasgf.Patient_survey__hcahps___hospit, 
                  link_var = Provider_ID)
```

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Type</th>
<th>Column Length</th>
<th>Library Name</th>
<th>Member Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>char</td>
<td>51</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>char</td>
<td>50</td>
<td>SASGF</td>
<td>PATIENT_SURVEY_HCAHPS___HOSPIT</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>char</td>
<td>50</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>HOSPITAL_NAME</td>
<td>char</td>
<td>71</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>HOSPITAL_NAME</td>
<td>char</td>
<td>71</td>
<td>SASGF</td>
<td>PATIENT_SURVEY_HCAHPS___HOSPIT</td>
</tr>
<tr>
<td>HOSPITAL_NAME</td>
<td>char</td>
<td>50</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>LOCATION</td>
<td>char</td>
<td>88</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>LOCATION</td>
<td>char</td>
<td>88</td>
<td>SASGF</td>
<td>PATIENT_SURVEY_HCAHPS___HOSPIT</td>
</tr>
<tr>
<td>LOCATION</td>
<td>char</td>
<td>86</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>MEASURE_ID</td>
<td>char</td>
<td>25</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>MEASURE_ID</td>
<td>char</td>
<td>15</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>MEASURE_NAME</td>
<td>char</td>
<td>72</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>MEASURE_NAME</td>
<td>char</td>
<td>98</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>STATE</td>
<td>char</td>
<td>2</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>STATE</td>
<td>char</td>
<td>2</td>
<td>SASGF</td>
<td>PATIENT_SURVEY_HCAHPS___HOSPIT</td>
</tr>
<tr>
<td>STATE</td>
<td>char</td>
<td>2</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>ZIP_CODE</td>
<td>num</td>
<td>8</td>
<td>SASGF</td>
<td>COMPLICATIONS_AND_DEATHS___HOSPIT</td>
</tr>
<tr>
<td>ZIP_CODE</td>
<td>num</td>
<td>8</td>
<td>SASGF</td>
<td>PATIENT_SURVEY_HCAHPS___HOSPIT</td>
</tr>
<tr>
<td>ZIP_CODE</td>
<td>num</td>
<td>8</td>
<td>WORK</td>
<td>HEALTHCARE_ASSOCIATED_INFECTIONS</td>
</tr>
<tr>
<td>ZIP_CODE</td>
<td>num</td>
<td>8</td>
<td>WORK</td>
<td>HOSPITAL_GENERAL_INFORMATION</td>
</tr>
</tbody>
</table>
Discussions:

- When variables exist in multiple datasets involved in a merge, and they’re not listed as BY variables, you need to ensure they are read from a single “most correct” source, or there’s a risk the incorrect value is saved.

- The SAS macro CHECK_VAR_EXIST was written to aid programmers in identifying more efficiently which variables could be wrongly overwritten even before the merging is done.

- The output of the macro is used to determine where to include a DROP or KEEP statement. It can also be used to determine the maximum length for each common variable, which could be handy when concatenating datasets using the SET statement, to prevent the truncation of the variable. Another use is to determine if any of the common variables have different types (character vs numeric).

- A simpler but less efficient way to check for common variables is by using OPTIONS MSGLEVEL=I. Setting MSGLEVEL to I will make the log display additional notes pertaining to the merge processing. However, this requires you to run the DATA step merging first and then check the log after.