Paper 2824-2018

Tools Designed to Aid in Quality Control of Administrative Data Research

Matthew Keller, MS, Washington University in St. Louis, School of Medicine; Katelin B. Nickel, MPH, Washington University in St. Louis, School of Medicine

ABSTRACT

Using administrative healthcare data for research, including health services, outcomes and comparative effectiveness research, is challenging for researchers because the data sources are not designed for research purposes. These data take different forms (e.g., hospital discharge data from individual hospitals, insurance claims data from health insurers) and are often reformatted by a secondary entity (e.g., individual state, government or commercial data warehouse) before being available for research. Together, these factors create substantial variation in the structure and content of the datasets. Therefore, when using administrative health databases for research purposes, the need for quality control becomes an important focus for data scientists. This paper will discuss examples of quality control measures utilized by programmers at our center. Specifically, we will present two macros used at the start of a project by the lead programmer and two macros used by a second quality control programmer after the analytic dataset has been created.

INTRODUCTION

The data source discussed in this paper is the State Inpatient Databases (SID) [1], which is supported by the Agency for Healthcare Research and Quality (AHRQ) and maintained by the Healthcare Cost and Utilization Project (HCUP). Additional HCUP databases include the State Ambulatory Surgery (SASD) [3] and State Emergency Department (SEDD) [4] Databases that will be mentioned but not discussed in detail. The SID database contains discharge records from participating states over many years. There is variation in the data elements by state and by year, which is exacerbated by the unique specifications of each project we extract using the SID. Most research projects have a standard process of pulling the requested records, organizing them into patient-level data (when available) and creating required variables. However, the individual details require a thorough review of each SAS® program to make sure the SID variables are handled properly.

The SID contains de-identified, all-payer hospital discharge data from community hospitals. HCUP provides standardized variables, for example the variable race, coded with values one thru six with three different missing values (., .A, .B) [2] even though each state reports race using different values. Many states have restrictions regarding what information is available in the data; for example, Nebraska does not provide race. Another example is some states provide a unique patient identifier to distinguish an individual person to allow for long-term follow up. This identifier does not span states and does not always span years within a state. It would be possible for a new user of HCUP data to not fully account for this fluctuation and erroneously follow a person across states because the identifier value is the same, when in reality two different people are represented in the data from the two states. A simple concatenation of the state abbreviation with the identifier solves this issue. With over a thousand variables in consideration, simple solutions are not always available.

Each project at our center begins with a work order from an investigator, usually a physician, who has a hypothesis about outcomes associated with a particular medical diagnosis or procedure. The investigator outlines the question and a general framework of the project including the International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) diagnosis and procedure codes and the Current Procedural Terminology (CPT) codes needed for the programmer to identify and organize millions of records into an analyzable dataset. The final dataset should have all variables properly labeled and ready for analysis. Traditional methods for quality control are used before, during and after the project – frequency tables, crosstabs, measures of central tendency, identification of outliers, re-coding missing values and checking for impossible values. This process, while very efficient at finding data errors, is incomplete. Due to the complexity of both the database and the specific projects, errors are still possible. The following macros will demonstrate an efficient process for identifying potential errors before and after programming has started. The code for the macros is provided in Appendix A.

THE SETUP: HCUP VARIABLES

Given the large number of variables in the HCUP database, it would be a daunting task to learn the intricacies of every variable. It is common for a single project to use multiple states across multiple years which compounds the variation. The first step in the authors' process was to capture this variation by identifying all variables by year and state across each database type (i.e., SID [1], SASD [3], SEDD [4]). The databases are stored in their own library and not all libraries have all or the same datasets. Having a single permanent metadata table that contained a description of each variable from all datasets laid the groundwork for building several quality control (QC) tests that can be used before, during and after a project is completed. The metadata table is only updated when new HCUP data are added to the libraries.

The variables in the metadata table include the state abbreviation, year of data, the database that contains the table, the library, table and variable name, the type, length and label for the variable, any associated formatting and finally the index associated with the variable name for those variables that can be iterated. These last three values (DX_Index, PR_Index and CPT_Index) are necessary for determining how many ICD-9-CM diagnosis, procedure or CPT codes are included in the table. An example of the usefulness of these variables will be given in a later section.

Figure 1 is a screen shot of the metadata table used for each of the macros presented below.

Hospst	Year	Source	Libname	Memname	Name	Type	Length	Label	Format	DX_Index	PR_Index	CPT_Index
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	CPT8	char	5	CPT/HCPCS procedure code 8		.	.	8
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	CPT9	char	5	CPT/HCPCS procedure code 9		.	.	9
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DIED	num	3	Died during hospitalization		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DISPUB92	num	3	Disposition of patient (UB-92 standard coding)		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DISPUNIFORM	num	3	Disposition of patient (uniform)		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DISP_X	char	2	Disposition of patient (as received from source)		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DMONTH	num	3	Discharge month		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DQTR	num	3	Discharge quarter		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DSHOSPID	char	13	Data source hospital identifier		.	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DX1	char	5	Principal diagnosis		1	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DX10	char	5	Diagnosis 10		10	.	.
CA	2005	SASD	HCSASDCA	CA_SASDC_2005_CORE	DX11	char	5	Diagnosis 11		11	.	.

Figure 1. Screen Shot of Metadata Table

MACRO 1: CHECKING AVAILABILITY OF A VARIABLE

The first macro identifies the availability of a variable during the study period. When discussing a new project, this macro is suitable in determining if a certain variable will be useful in the project. It allows the programmer to pass a variable name and the database type (SID [1], SASD [3], SEDD [4]) and the return output is a listing of states and years for which the variable is available. When starting a new project, if a programmer is not familiar with the availability of a particular variable, then he or she can quickly determine if the variable is usable for the states and years needed for the project.

Figure 2 is a screen shot of the output produced from %Variable_Available(VAR=hcup_surgery_narrow). As you can see from the output, this variable is only available in 2011.

Variable is Available in Source=SASD for These States and Years

Hospst	Year	Libname	Memname	Name	Type	Length	Label	Format	DX_Index	PR_Index	CPT_Index
CA	2011	HCSASDCA	CA_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
CO	2011	HCSASDCO	CO_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
FL	2011	HCSASDFL	FL_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
MD	2011	HCSASDMD	MD_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
NE	2011	HCSASDNE	NE_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
NJ	2011	HCSASDNJ	NJ_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
NV	2011	HCSASDNV	NV_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
NY	2011	HCSASDNY	NY_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.
VT	2011	HCSASDVT	VT_SASDC_2011_CORE	HCUP_SURGERY_NARROW	num	3	Revised HCUP_AS (PCLASSn=4, or narrow definition)		.	.	.

Figure 2. Output of Variable_Available Macro

MACRO 2: PROJECT METADATA

Built upon this concept of checking variables before the start of a project, this next macro is much more involved but provides substantial benefit before programming is started. When calling the macro, a dataset containing the states and years for the project are passed as a parameter. The macro will create

the PrjMetaData table with pre-defined columns. Next, the macro will use the database (SID [1], SASD [3], SEDD [4]) given in the macro call and construct the important parameters of the project – the states being used, the years of data requested, the years of data available and most importantly the number of diagnosis and procedure codes available across those states and years.

This last part is crucial because a few states provide over 50 diagnosis codes while other states provide as few as nine (in the SID data available to the authors). When many states and years are requested in a project, bias can be introduced when pulling data based on varying numbers of diagnosis and/or procedure code fields. This is a quick way of reviewing the parameters before pulling any data.

Figure 3 is an example of the PrjMetaData_SID table produced from %Give_Database_Details(TABLE, DATABASE):

hospst	Source	PrjBeginYear	PrjEndYear	DBSMinYearAvail	DBSMaxYear Avail	MinDX	MaxDX	MinPR	MaxPR
AR	SID	2009	2014	2008	2014	18	18	8	8
AZ	SID	2004	2007	2000	2014	9	9	6	6
CA	SID	2005	2011	2003	2011	25	25	21	21
FL	SID	2006	2015	2000	2015	31	31	31	31
IA	SID	2010	2012	2000	2014	57	66	26	50
MA	SID	2011	2014	2000	2014	15	71	15	66
NE	SID	2006	2014	2001	2015	9	9	6	6
NY	SID	2006	2014	2000	2014	15	25	14	15
WA	SID	2004	2006	2000	2015	9	9	6	6
WI	SID	2014	2015	2000	2015	66	67	50	50

Figure 3. Screen Shot of PrjMetaData_SID Table

MACRO 3: NEWLY CREATED VARIABLES

The next macro is used during QC by another programmer to extract all programmer-created variables from the final dataset in order to focus review. The dataset produced is the name, length and label for each variable that was newly created and is very useful for quickly determining if the variable has a label which is a requirement for final datasets at our center.

Figure 4 displays the output from %CreatedVars(LIBRARY, TABLE):

Column Name	Column Length	Column Label
APRNMPER1000	8	ADVANCED PRACTICE NURSES MIDWIVES PER 1000 CENSUS POPULATION
APRNPER1000	8	ADVANCED PRACTICE NURSES PER 1000 CENSUS POPULATION

Figure 4. Screen Shot of Output for CreatedVars Macro

MACRO 4: CHANGES IN LENGTH AND LABELS

The next macro is more involved and addresses the potential issue of original HCUP variables becoming manipulated during the programming of a project. Two issues can be created: the variable's length can be changed leading to truncation of the values or the variable label can be replaced or lost. The macro starts by identifying the states and years used in the final dataset. The original name, length and label for the HCUP variables are pulled from the metadata table and matched to the HCUP variables in the final dataset. The table that is created contains the HCUP variables that were manipulated during the program and lost their original length or label. The next step is to check to see which variables are being provided that do not span the entire study period. Finally, the macro prints warnings to the log to notify the QC-programmer that there may be an issue.

Figure 5 is the Changed_Vars table that is produced from %Changed_and_Yearly_Vars(LIBRARY, TABLE):

name	HCUPNAME	length	HCUPLENGTH	label	HCUPLABEL
DX1	DX1	4	5	Diagnosis 1	Diagnosis 1
HCUP_ED	HCUP_ED	3	3	ER Indicator	HCUP Emergency Department service indicator
PAY2_X	PAY2_X	2	1	Secondary expected payer (as received from source)	Secondary expected payer (as received from source)
PAY3_X	PAY3_X	2	1	Tertiary expected payer (as received from source)	Tertiary expected payer (as received from source)

Figure 5. Screen Shot of the Changed_Vars Table Produced by the Changed_and_Yearly_Vars Macro

Figure 6 is the Yearly_Vars table that is produced from %Changed_and_Yearly_Vars(LIBRARY, TABLE):

Hospst	name	HCUPNAME	length	HCUPLENGTH	label	HCUPLABEL	HCUPYear	MinDataYear	MaxDataYear
AR	dx_admitting	DX_Admitting	8	5		Admitting Diagnosis Code	2012	2009	2014
AR	dx_admitting	DX_Admitting	8	5		Admitting Diagnosis Code	2013	2009	2014
AR	dx_admitting	DX_Admitting	8	7		Admitting Diagnosis Code	2014	2009	2014
FL	dx_admitting	DX_Admitting	8	5		Admitting Diagnosis Code	2012	2006	2015
AR	serviceline	SERVICELINE	8	3		Hospital Service Line	2014	2009	2014
FL	serviceline	SERVICELINE	8	3		Hospital Service Line	2014	2006	2015
FL	serviceline	SERVICELINE	8	3		Hospital Service Line	2015	2006	2015
MA	serviceline	SERVICELINE	8	3		Hospital Service Line	2014	2011	2014

Figure 6. Screen Shot of the Yearly_Vars Table Produced by the Changed_and_Yearly_Vars Macro

Figure 7 displays the warnings placed in the log when changes to the standardized HCUP variables are identified during execution of %Changed_and_Yearly_Vars.

```
WARNING: THERE ARE CHANGES TO HCUP VARIABLES EITHER LENGTH OR LABEL
WARNING: NOT ALL VARIABLES AVAILABLE OVER THE YEARS
```

Figure 7. Screen Shot of the Log after Running Changed_and_Yearly_Vars Macro

CONCLUSION

The macros described above provide a quick process for checking data availability and reviewing other programmers' projects to identify potential data integrity problems. The output from Macros 1 and 2 help the programmer prevent issues before the project begins. Macros 3 and 4 focus quality control of the final dataset on the variables that were created and changed. After identifying the variables that were created, the variables can be saved in macro variables and run through other SAS® procedures, such as PROC FREQS, PROC MEANS and PROC UNIVARIATE. The next step in developing additional macros for quality control is to identify some common data errors that can be discovered using the log and output to further streamline the quality control process.

REFERENCES

1. HCUP State Inpatient Databases (SID). Healthcare Cost and Utilization Project (HCUP). 2005-2009. Agency for Healthcare Research and Quality, Rockville, MD. http://www.hcup-us.ahrq.gov/sidoverview.jsp.
2. HCUPnet. Healthcare Cost and Utilization Project (HCUP). 2006-2009. Agency for Healthcare Research and Quality, Rockville, MD. http://hcupnet.ahrq.gov/ Accessed October 9, 2017.
3. HCUP State Ambulatory Surgery and Services Databases (SASD). Healthcare Cost and Utilization Project (HCUP). 2007-2009. Agency for Healthcare Research and Quality, Rockville, MD. http://www.hcup-us.ahrq.gov/sasdoverview.jsp.
4. HCUP State Emergency Department Databases (SEDD). Healthcare Cost and Utilization Project (HCUP). 2009. Agency for Healthcare Research and Quality, Rockville, MD. http://www.hcup-us.ahrq.gov/seddoverview.jsp.

ACKNOWLEDGMENTS

Thank you to Scott Bass for his review and comments on this project. We would also like to thank Margaret A. Olsen PhD, MPH for her comments on the project as well as her continued support at Washington University in St. Louis, School of Medicine. The Center for Administrative Data Research is supported in part by the Washington University Institute of Clinical and Translational Sciences Grant UL1 TR002345 from the National Center for Advancing Translational Sciences (NCATS) of the National Institute of Health (NIH), Grant Number R24 HS19455 through the Agency for Healthcare Research and Quality (AHRQ).

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Matthew Keller
kellermr@wustl.edu

APPENDIX A:

```
/*
PROGRAM PURPOSE: Macros for QA and QC on CADR Projects

CREATED BY: MATTHEW KELLER, KATELIN B NICKEL

NOTES - SAMPLE CODE

*/

*************************************************************************
*************************************************************************
MACRO #1 - CHECKING AVAILABILITY OF A VARIABLE
*************************************************************************
*************************************************************************;
*REVIEW VARIABLES FOR STUDY - LIKE A QUICK SEARCH;
   %macro variable_available(VAR=);

      *SORT DATASET FOR BY STATEMENT IN PROC PRINT;
      proc sort data=u.hcupvars out=_test_;
         by source;
      run;

      *PRINT INFO BROKEN OUT BY SOURCE;
      options nobyline;
      proc print data=_test_ noobs;
         where lowcase(name) = "%lowcase(&VAR)";
         by source;
        title1 'Variable is Available in #byline for These States and Years';
      run;

      *CLEAN UP;
      proc delete data=_test_;run;
      title;

   %mend variable_available;

/*
*INPUTS - VARIABLE TO SEARCH FOR;
%Variable_Available(VAR=hcup_surgery_narrow);
*/

*************************************************************************
*************************************************************************
MACRO #2 - CREATE PROJECT METADATA TABLE
*************************************************************************
*************************************************************************;
*COMPILE A TABLE OF RELEVANT INFORMATION WHEN STARTING A PROJECT;
   %macro give_database_details(TABLE=, DATABASE=);
```

```
*COLLECT MIN/MAX DX, PR FOR EACH STATE BETWEEN YEARS OF INTEREST;
*FIRST SORT METADATA TABLE;
proc sort data=u.hcupvars out=_test_;
   by hospst year;
run;

*RUN MEANS TO GET MAX VALUES INTO A DATASET;
proc means data=_test_ max noprint;
   where source = "%upcase(&DATABASE)";
   by hospst year;
   var dx_index pr_index;
   output out=_test2_;
run;

*JOIN THE OUTPUT FROM MEANS TO THE INFO PULLED FROM INPUT
DATASET;
proc sql;
   create table prjMetaData_&DATABASE as
   select distinct a.hospst,
     "%upcase(&DATABASE)" as Source,
     a.PrjBeginYear,
     a.PrjEndYear,
     min(b.minYr) as DBSMinYearAvail,
     max(b.maxYr) as DBSMaxYearAvail,
     min(c.DX_Index) as MinDX,
     max(c.DX_Index) as MaxDX,
     min(c.PR_Index) as MinPR,
     max(c.PR_Index) as MaxPR
   from (select distinct hospst,
                min(year) as PrjBeginYear,
                max(year) as PrjEndYear
         from &TABLE
         group by hospst
        ) as a
   left join (select distinct hospst,
                min(year) as minyr,
                max(year) as maxyr
              from u.hcupvars
              group by hospst)
     as b
     on a.hospst = b.hospst
   left join _test2_(where=(_stat_ = 'MAX')) as c
     on a.hospst = c.hospst
     and c.year between a.PrjBeginYear
     and a.PrjEndYear
   group by a.hospst
;quit;

*DELETE CREATED DATASET;
proc delete data=_test_;run;
proc delete data=_test2_;run;
```

```
   %mend give_database_details;

/*
*INPUTS - DATASET THAT HOLDS THE STATES AND YEARS FOR STUDY,
DATABASE THAT IS BEING USED FOR THE STUDY;
%Give_Database_Details(TABLE=test, DATABASE=SID);
*/

*******************************************************************
*******************************************************************
MACRO #3 - NEWLY CREATED VARIABLES
*******************************************************************
*******************************************************************;
*RETURNS THE CREATED VARIABLE NAMES FROM A USER-CREATED DATASET THAT
ARE NOT IN THE STANDARD VARIABLE LIST FOR HCUP DATA;
   %macro CreatedVars(LIBRARY, TABLE);

      *SET MISSING LIBRARY TO WORK;
      %if (&LIBRARY = ) %then %do;
        %let LIBRARY = WORK;
      %end;

     *ANOTHER QUICK SEARCH QUERY;
     proc sql;
       select distinct name, length, label
       from dictionary.columns
       where upcase(libname) = upcase("&LIBRARY")
         and upcase(memname) = upcase("&TABLE")
         and upcase(name) not in (select upcase(name)
                                  from u.hcupvars
                                 )
     ;quit;

   %mend CreatedVars;

/*
*SIMPLY PASS THE DATASET OF INTEREST TO THE MACRO;
%CreatedVars(LIBRARY=work, TABLE=test);
*/

*******************************************************************
*******************************************************************
MACRO #4 - CHANGES IN LENGTH AND LABELS FOR HCUP ORIGINAL VARIABLES
*******************************************************************
*******************************************************************
*CHECK TO DETERMINE IF AN EXISTING HCUP VARIABLE WAS MANIPULATED;
   %macro Changed_and_Yearly_Vars(LIBRARY, TABLE);

      *SET MISSING LIBRARY TO WORK;
      %if (&LIBRARY = ) %then %do;
        %let LIBRARY = WORK;
      %end;
```

```
*PULL IN STATES AND YEARS FOR STUDY DATA;
proc sql;
  create table temp as
  select distinct hospst,
         min(year) as minyr,
         max(year) as maxyr
  from &LIBRARY..&TABLE
  group by hospst
;quit;

*CREATE A TABLE OF HCUP VARIABLES THAT CHANGED LENGTH/LABEL;
proc sql;
  create table CHANGED_VARS as
  select distinct a.name, b.name as HCUPNAME,
        a.length, b.length as HCUPLENGTH,
        a.label, b.label as HCUPLABEL
  from (select name, length, label
       from dictionary.columns
       where upcase(libname) = upcase("&LIBRARY")
       and upcase(memname) = upcase("&TABLE")
      ) as a
  inner join (select distinct a.name, a.length, a.label
              from u.hcupvars as a
              inner join temp as b on a.hospst = b.hospst
              and a.year between b.minyr and b.maxyr
             )
    as b on lowcase(a.name) = lowcase(b.name)
    and (a.length ne b.length
         or lowcase(a.label) ne lowcase(b.label)
        )
;quit;

*HCUP VARIABLES THAT DO NOT SPAN THE ENTIRE STUDY PERIOD;
proc sql;
  create table YEARLY_VARS as
  select distinct b.hospst, a.name, b.name as HCUPNAME,
                  a.length, b.length as HCUPLENGTH,
                  a.label, b.label as HCUPLABEL,
                  b.year as HCUPYear,
                  b.minyr as MinDataYear,
                  b.maxyr as MaxDataYear
  from (select name, length, label
        from dictionary.columns
        where upcase(libname) = upcase("&LIBRARY")
        and upcase(memname) = upcase("&TABLE")
      ) as a
  inner join (select distinct a.hospst,
                              a.name,
                              a.length,
                              a.label,
                              a.year,
```

```
                                b.minyr,
                                b.maxyr
                 from u.hcupvars as a
                 inner join temp as b
                 on a.hospst = b.hospst
                 and a.year between b.minyr and b.maxyr
                )
        as b on lowcase(a.name) = lowcase(b.name)
    group by b.name
    having count(unique b.year) < (b.maxyr - b.minyr)
  ;quit;

  *COUNT IF TABLES HAVE VALUES;
  proc sql noprint;
    select distinct count(unique name) into :ctERROR
    from changed_vars
  ;
    select distinct count(unique name) into :ctERRORS
    from yearly_vars
  ;quit;

  *IF VALUES THEN PRINT WARNING MESSAGES;
  %if (&ctERROR ne) %then %do;
    %put WARNING: THERE ARE CHANGES TO HCUP VARIABLES EITHER
                  LENGTH OR LABEL;
  %end;

  %if (&ctERRORS ne) %then %do;
    %put WARNING: NOT ALL VARIABLES AVAILABLE ACROSS ALL YEARS;
  %end;

  *CLEAN UP TEMPORARY TABLE;
  proc delete data=temp;run;

%mend Changed_and_Yearly_Vars;

/*
*SIMPLY PASS THE DATASET OF INTEREST TO THE MACRO;
%Changed_and_Yearly_Vars(LIBRARY=work, TABLE=test);
*/
```