Data Hygiene Routines for Administrative Healthcare Data

Craig Dickstein
Tamarack Professional Services, LLC

Boston Area SAS Users Group
September 2012

The Business Case

- Programmer/analysts need to have lookup capability in industry standard tables to support data processing.
  - Data hygiene routines
  - Claim selection criteria
  - Displaying formats for report production
- Business rules are needed to drive processes.
  - Rules need to reside outside of problem programs.

2

The Technical Challenge

- Must have intimate knowledge of code architectures
- Need machine readable 'date-stamped' code sets with nomenclature

DRG Code	Nomenclature	Beg_Eff_DT	End_Eff_DT
009	Bone Marrow Transplant	2010-01-01	2011-12-31
010	Pancreas Transplant	2009-07-01	2099-12-31

3

ICD-9 Diagnosis Codes – Architecture

- ~16,000 codes for signs, symptoms, injuries, diseases, and conditions
- Three to five characters long **XXX.XX**
- Alphanumeric – beginning with 0-9, E, or V
- Codes range from 001. through V91.99
- E codes – accident, injury, or poison-related
  - External Cause of Injury (ECI)
  - *Exception:* ***Ennn.n***
- V codes – patient care not related to a particular disease or injury
  - Typically not used as a principal diagnosis

4

continued...

ICD-9 Diagnosis Codes – Architecture

- Invalid code if less than three characters long
- Category **X X X . _ _**
- Etiology, anatomic site, manifestation **_ _ _ . X X**

403.	Hypertensive chronic kidney disease
403.0	Hypertensive chronic kidney disease, malignant
403.00	Hypertensive chronic kidney disease, malignant, with chronic kidney disease stage I through stage IV
403.01	Hypertensive chronic kidney disease, malignant, with chronic kidney disease stage V or ESRD

5

CPT® Category I – Architecture

- Current Procedural Terminology
  - American Medical Association
- Billable services provided by a person, not a facility
- Five digits, ranging from 00100 to 99607 **X X X X X**
  - Character type data, ***not*** numeric
  - No decimal point, no alpha characters
  - Cannot be less than five characters
- ~9,000 codes and descriptors

99201	Office/OP New Patient - 10 minutes
99202	Office/OP New Patient - 20 minutes
99203	Office/OP New Patient - 30 minutes
99204	Office/OP New Patient - 45 minutes
99205	Office/OP New Patient - 60 minutes

6

CPT Category II – Architecture

- Performance Measurement codes
  - non-billable
  - will become important under new and improved reimbursement mechanisms (healthcare reform)
- Four numeric characters are followed by an 'F'

x x x x F

2000F	Blood pressure measured
2001F	Weight recorded
2002F	Clinical signs of volume overload (excess) assessed
2004F	Initial examination of the involved joint(s)
2010F	Vital signs (temperature, pulse, respiratory rate, and blood pressure) documented and reviewed
2014F	Mental status assessed

7

CPT Category III – Architecture

- Emerging Technology
  - non-billable
  - may be accepted/promoted to Category I code within 5 years
- Four numeric characters followed by a 'T'

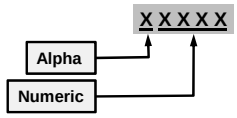
x x x x T

0059T	Cryopreservation; oocyte(s)
0143T	Laparoscopy, surgical, pancreatic islet cell transplantation through portal vein
0207T	Evacuation of meibomian glands, automated, using heat and intermittent pressure, unilateral

8

HCPCS Codes – Architecture

- Healthcare Common Procedure Coding System
  - Centers for Medicare & Medicaid (CMS)
  - billable events for Ancillary providers and Facilities
- Over 5,000 codes
- Five characters
  - Cannot be shorter



- Grouped by the type of service or equipment received

9

Code Architecture

Codes Set	Minimum Length	Maximum Length	Character Composition
ICD-9 dx	3	5	0 – 9, V, E
ICD-10 dx	3	7	alphanumeric
ICD-9 px	3	4	numeric
ICD-10 px	7	7	alphanumeric
CPT - Category I	5	5	numeric
CPT - Category II	5	5	alphanumeric
CPT - Category III	5	5	alphanumeric
HCPCS	5	5	alphanumeric
Revenue	4	4	alphanumeric
Type of Bill	4	4	alphanumeric
NDC	11	11	numeric, hyphen
LOINC	3	7	numeric, hyphen
DRG	3	3	numeric

10

The FORMAT Procedure – a review

PROC FORMAT can do the following:

- Create informats or formats
- Store descriptions of informats or formats in a data set
- Print a listing of informats or formats
- Use a data set to create informats or formats
  - ***The focus of this discussion***

11

PROC FORMAT – a review

```
PROC FORMAT                          [options];
    VALUE [$]name                    [(options)]
        value-range-set(s)
    ;
    PICTURE name                     [(options)]
        value-range-set-1
        [...value-range-set-n]
    ;
    INVALUE [$]name                  [(options)]
        value-range-set(s)
    ;
    …
```

12

Procedure Options of Interest

- LIBRARY=
  - SAS library to store the created informats and formats
  - The default is the 'work' library
- CNTLIN=
  - SAS data set from which to build informats or formats
- FMTLIB
  - Prints information about informats and formats

13

VALUE Statement

VALUE [$]*fmtname*
start = "*label*"
start - end = "*label*"
LOW - end = "*label*"
start - HIGH = "*label*"
OTHER = "*label*"
;

- Key components of the VALUE statement
- CNTLIN= procedure option
  - A data set with these variables
    - **`Type  Fmtname`**
    - **`Start  End  Label  HLO`**

14

CNTLIN= Option

The CNTLIN= option has the following features:

- Specifies an existing SAS data set.
- Has no need for the VALUE statement
- Requires these variables:
  - FMTNAME
  - START
  - LABEL
- Has optional variables
  - TYPE (or use a $ with a character format name)
  - END

15

VALUE Statement – An Example

```
VALUE $hcpcs
  "A0433"="ADVANCED LIFE SUPPORT LEVEL 2"
  "A0434"="SPECIALTY CARE TRANSPORT"
  "A0435"="FIX WING AIR MILEAGE"
  "A0436"="ROTARY WING AIR MILEAGE"
        ...
   OTHER = "** UNKNOWN HCPCS **"
;
```

type, fmtname

start

label

other

Most medical code sets do not have a value range, low or high.

16

Building Data Sets and Displaying Formats

This demonstration illustrates the following:

- Building a data set from imported tables
- Building format(s) from a data set
- Using the format(s) as lookup tables

17

Presenter Contact Information

Craig Dickstein

Tamarack Professional Services, LLC
P.O. Box 128, Caratunk, ME 04925-0128
FAX: by request – Voice: (207) 672-4989

tamarack.llc@myfairpoint.net

18

```
/* ********************************************************************************************** *\
|          SET UP JOB PARAMETERS                                                                   |
\* ********************************************************************************************** */

%let rootdir = c:\documents and settings\craig dickstein\my documents;
%let datadir = &rootdir\SAS Institute\BHCDW\Data;
%let codedir = &rootdir\SAS Institute\BHCDW\Source Code;

%let work    = work;

libname _all_ clear;
libname bhcdw "&datadir" ACCESS=readonly;

options  mautosource mrecall
         fmtsearch=(work bhcdw )             /* order in which formats are searched for and used  */
         sasautos =("&codedir" sasautos)     /* order in which macros are searched for and used   */
;
run;

*** ------------------------------------------------------------------------------------------- ***;
*** Import the data resource                                                                     ;
*** ------------------------------------------------------------------------------------------- ***;

proc import out=&work..procedure_codes                    /* ----import billable procedure codes */
            datafile= "&datadir\code sets.xlsx"
            dbms=excel replace;
     sheet='px$';        /* specify the worksheet within the workbook - see RANGE= to limit       */
     getnames=yes;       /* generate SAS variable names from the data values in the first row     */
     mixed=yes;          /* converts numeric data to character values when mixed data types found */
     scantext=yes;       /* uses the longest string found as the character variable length        */
     textsize=2000;      /* maximum variable length allowed while importing data                  */
     usedate=yes;        /* assign a DATE format while importing a date column                    */
     scantime=yes;       /* scan time data while importing data from a time column                */
run;

*** ------------------------------------------------------------------------------------------- ***;
*** Create procedure code lookup tables with Proc Format                                         ;
*** ------------------------------------------------------------------------------------------- ***;

data &work..px_cntlin(drop=short_nomenclature);                    /* ------build CNTLIN data set */
    length fmtname $8;                              /* initiate length prior to sourcing the var */
    set &work..procedure_codes(rename=(code=start)) end=last;
    label = propcase(short_nomenclature);
    select;
           when(anyalpha(start) =  1 ) fmtname = "$hcpcs";
           when(anyalpha(start) =  0 ) fmtname = "$cptI" ;
           when(substr(start,5) = "F") fmtname = "$cptII" ;
           when(substr(start,5) = "T") fmtname = "$cptIII" ;
       end;

/* Code WHEN conditions from most frequently occurring to least.
   Processing is sequential until a WHEN condition is met.         */

    output;
    fmtname = "$allpx";   output;
    if last then do;
                   hlo='O';
                   fmtname = "$hcpcs" ;  label='** UNKNOWN HCPCS **'  ;  output;
                   fmtname = "$cptI"  ;  label='** UNKNOWN CPT I **'  ;  output;
                   fmtname = "$cptII" ;  label='** UNKNOWN CPT II **' ;  output;
                   fmtname = "$cptIII";  label='** UNKNOWN CPT III **';  output;
                   fmtname = "$allpx" ;  label='** UNKNOWN Px Code **';  output;
              end;
run;

proc sort data=&work..px_cntlin;              /* necessary for proper functioning of Proc Format */
    by fmtname;

proc freq data=&work..px_cntlin;              /* reasonability check                             */
    tables fmtname;
run;

/* >>>>>>>>>>>>>   VIEW TABLE   <<<<<<<<<<<<<<<<<<< */
```

```
proc format cntlin=&work..px_cntlin;* fmtlib ;            /* ---build multiple procedure formats */

/* >>>>>>>>>>>>>   fmtlib - show alternate method with SAS EXPLORER */
run;

*** ------------------------------------------------------------------------------------------ ***;
*** Use procedure code lookup tables to indentify bad values                                    ;
*** ------------------------------------------------------------------------------------------ ***;

data _NULL_;                                          /* ------ look at professional detail lines
                                                                put notes to the LOG              */
    set bhcdw.pro_claim_detail(keep=claim_id px );
    if Px ne " " and put(Px,$allpx.) =: "**"
      then put "Invalid procedure code for claim #" Claim_ID +2 "-- Code is " Px;
run;

data &work..bad_fac_proccodes;                       /* ------- look at facility detail lines
                                                                create a dataset of bad obs       */
    set bhcdw.fac_claim_detail;
    if not (Px = " " or anyalpha(Px,2) = 2)               /* blank and HIPPS code ignored */
       and put(Px,$allpx.) =: "**"
      then output;
run;

proc freq data=&work..bad_fac_proccodes order=freq;              /* ------------ display results */
    tables Px;
run;

*** ------------------------------------------------------------------------------------------ ***;
*** Allocate the data resource - an alternate method of bringing in external data               ;
*** ------------------------------------------------------------------------------------------ ***;

libname readxls EXCEL "&datadir\code sets.xlsx"
                      MIXED=yes                /* converts numeric data to character values when
                                                                          mixed data types found */
                      ACCESS=readonly
      ;

proc contents data=readxls._all_ directory nods;
run;

/* >>>>>>>>>>>>>   SAS EXPLORER   <<<<<<<<<<<<<<<<<< */

*** ------------------------------------------------------------------------------------------ ***;
*** Create ICD9 code lookup tables with Libname and Proc Format                                 ;
*** ------------------------------------------------------------------------------------------ ***;
run;
data dx_cntlin(drop=short_nomenclature)
     icdpx_cntlin(drop=short_nomenclature);
    length fmtname $8;
    set readxls."icd9$"N(rename=(code=start)) end=last;       /* note the need for a name literal */
    label = propcase(short_nomenclature);

         if (   index(start,'.')  = 4                              /* a Dx code                 */
             or length(start)     = 3                              /* min length for Dx code    */
             or substr(start,1,1) = 'E'                            /* exception to first rule   */
            )
           then do;
                  fmtname = "$dx";
                  start   = compress(start,'.');                   /* lose the decimal point    */
                  output dx_cntlin;
                end;
    else if (   substr(start,3,1) = '.'                            /* a Surg Px code            */
             or length(start)     = 2
            )
           then do;
                  fmtname = "$surgpx" ;
                  start   = compress(start,'.');                   /* lose the decimal point    */
                  output icdpx_cntlin;
                end;
    if last then do;
                   hlo='O';
                   fmtname = "$dx"     ;  label='** UNKNOWN DX CODE **'  ; output dx_cntlin;
                   fmtname = "$surgpx" ;  label='** UNKNOWN SURG CODE **'; output icdpx_cntlin;
                 end;
run;
```

```
proc format cntlin=dx_cntlin;
proc format cntlin=icdpx_cntlin;
run;

*** ------------------------------------------------------------------------------------------ ***;
*** Use these tables to find bad facility codes                                               ;
*** ------------------------------------------------------------------------------------------ ***;
run;
data _NULL_;                                                    /* putting data errors to the log */
    set bhcdw.fac_claim_header;
    array dx (*) Pdx Dx:;                     /* this is colwise data requiring array processing */
    do i = 1 to dim(dx);
                        if dx(i) = " " then leave;          /* exit loop when first empty element */
                        if put(dx(i),$dx.) =: "**"
                          then put "Invalid diagnosis " i "for claim #" Claim_ID +2
                                   " -- Code is " Dx(i);
    end;
run;

data &work..bad_pro_dxcodes;                               /* putting data errors to a work dataset */
    keep claim_id dx: baddx;
    set bhcdw.pro_claim_header;
    array dx (*) Dx:;                         /* implicit array, importance of naming convention */
    do i = 1 to dim(dx)
      while(dx(i) > ' ');                                   /* different form of DO WHILE        */
           if put(dx(i),$dx.) =: "**"
             then do;
                    baddx = catx('.',substr(Dx(i),1,3),substr(Dx(i),4));     /* put decimal back */
                    output;
                end;
    end;
run;
```