SAS® Techniques for Incorporating Medical Code Updates into Longitudinal Health Care Data

Perry Watts, Independent Consultant
Alan C. Wilson, Ph.D., Robert Wood Johnson Medical School

Abstract

Processing longitudinal health care data presents a unique set of challenges to the SAS programmer. Typically, very large transaction data sets are involved with each record containing multiple medical codes describing a patient's hospitalization. The code definitions are updated annually, and those updates must also be tracked so that longitudinal study results are accurately reported. Because update patterns have not been fully identified, they have to be inferred from both the structure of the codes and an associated conversion table. Once identified, techniques then have to be developed for handling the updates. Two such techniques are described in the paper. Format concatenation is used to retrieve time-dependent ICD-9 Diagnostic (DX) code descriptions, and code mapping with the POINT option in a data step returns comparable codes needed for tallying cross-year frequencies. Sample data, source code listings, and generated output show how the techniques work and how they influence study results. Our ultimate goal in completing this exercise is to extend insights gained from working with multiple ICD-9 version updates to mapping ICD-10 codes to their ICD-9 predecessors. If successful, developers can continue to work fruitfully with software that is ICD-9 code compatible.

Problem Definition

Diagnostic codes in the health care industry are constantly changing. New diseases are classified and existing ones are re-classified. This means that all types of edits are possible: addition, modification and deletion. In addition, both code definitions and their associated descriptions can change over time. In 1998, for example, the description for DX code 041.04 was changed from STREPTOCOCCUS GROUP D to ENTEROCOCCUS GROUP D, and in 2000 the code, 372.8, was replaced by 372.89 for CONJUNCTIVA DISORDER NEC. In the first instance the code was held constant and the description changed, and in the second different codes were used for the same diagnosis.

A more general problem emerges when an attempt is made to define a complete rule-set that explains both a code's structure and how it is updated. In the ideal situation, the fourth and fifth digits in the coding scheme are reserved for increased refinement in diagnosis (Scerbo et. al., p. 11). Over time DX codes become longer, but predecessors with the same prefixes can easily be identified. Figure 1 shows how the set of diagnostic codes for salmonella configured as a hierarchical tree adheres to these guidelines. Obsolete parent nodes in the tree are grayed out meaning the referenced codes 003 and 003.2 in Figure 1 should not be found in a current transaction data set of patient hospitalizations. Progeny codes also adhere to the guidelines, because they contain exactly one digit more than their direct ancestors.

Unfortunately, however, the neat hierarchy in Figure 1 is frequently eclipsed by an arrangement similar to the one depicted in Figure 2 for pulmonary disease. Without a CDC conversion table it would be impossible to determine that 799.1 is directly related to 518.81, and that code 518.81 has generated siblings, 518.83 and 518.84, rather than more refined subordinates. Furthermore, two distinct trees are required for displaying all the related five-digit codes that range from 518.81 to 518.89, and codes 799.1 plus 518.81 continue to co-exist with their progeny.

Figure 1. A hierarchical tree shows how salmonella diagnostic codes are increasingly refined over time. Codes with a gray background are obsolete.

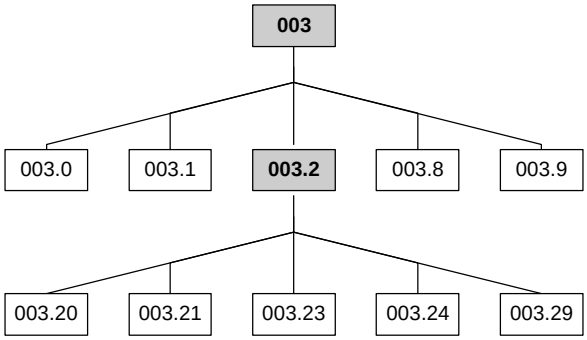
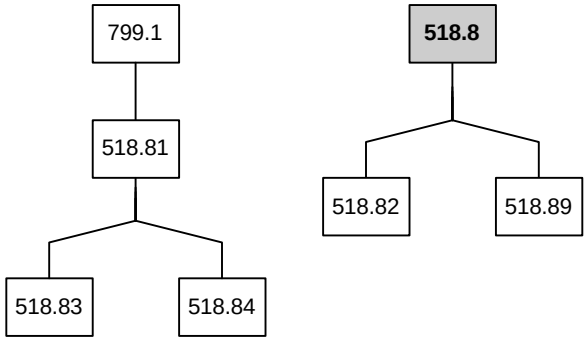


Figure 2. The nonhierarchical, concurrent arrangement among codes for pulmonary disease is displayed in two trees.



The presence of several hundred entries in the conversion table that resemble the schematic in Figure 2 shows that the lexicon for diagnostic code updates is complicated and that its derivation must arise from questions that are asked of the data. While books describing ICD-9 codes are available, they are generally geared towards providing detailed instructions for assigning correct codes to a patient hospitalization rather than in presenting a global picture about how diagnostic codes relate to each other and how they evolve over time. Therefore we rely on SAS programs to provide the information about update patterns described later in the paper. While the programs are not reviewed in the paper, they are available upon request.

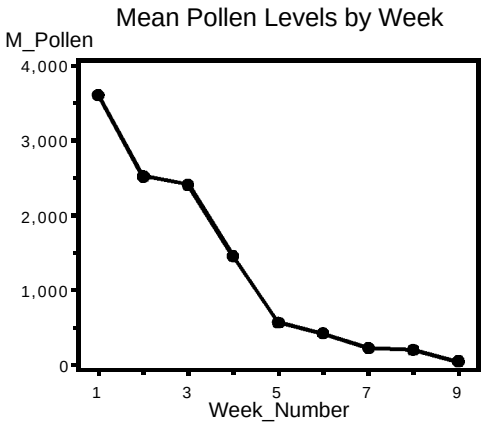
Longitudinal Data

The presence of changing code definitions adds another layer of complexity to the already difficult task of preparing data for longitudinal data analysis. Ron Cody alludes to this difficulty when he defines terms in the preface of his book, *Longitudinal Data and SASÒ: A Programmer's Guide:*

> In a SAS data set, performing calculations within an observation is relatively simple; performing calculations or making comparisons between observations is more difficult. In this book I use the term longitudinal data techniques to refer to operations involving two or more observations per subject (xvii).

With health care data, not only are there multiple observations per subject, but *what* is being observed also changes over time. This issue is addressed by example in Figures 3 and 4 below.

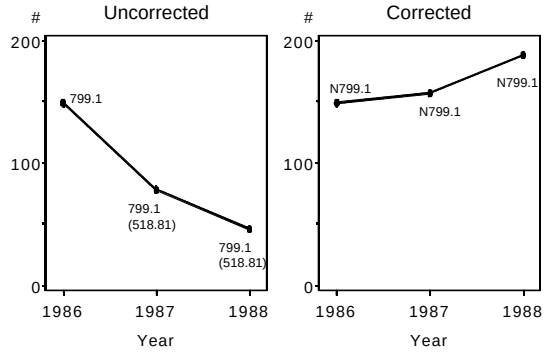
Figure 3. Average Pollen Counts for a short-term study from *Longitudinal Data and SASÒ: A Programmer's Guide*, p. 125. Since the same methods are used to collect data at weekly intervals, the downward trend is accurate.



In Figure 3 average pollen counts calculated at weekly intervals are displayed for a short-term study. It is safe to assume that the procedures used for obtaining the counts do not change over time. A more complicated situation is portrayed in Figure 4 where counts are tallied for pulmonary disease patients hospitalized at XYZ General from 1986 to 1988. Midway through the study, code 799.1 diagrammed in Figure 2 is mapped to 518.81. If the code mapping is not accommodated, undercounts are inevitable, but counts can also be inflated if a single patient is counted twice with the same diagnosis. The normalized code, N799.1, in the "corrected" plot redefines 518.81 as 799.1 and removes any duplicate counts that are a by-product of the redefinition. Totals change dramatically.

The use of the word "subject" in Cody's definition of longitudinal data may cause some confusion when hospital episodes are evaluated over time. Clearly, episodes are not repeated, nor are re-hospitalizations for a single patient even recorded. Instead the "subjects" in these data are defined later by a data analyst and may include such aggregate entities as geographical regions, particular hospitals (as in Figure 5), or particular departments within a hospital. In fact the DX codes themselves may play the role of "subject" in an analysis.

Figure 4. The definition of pulmonary disease changes in 1987.



Identifying Usable Code Update Patterns

Update Sources:

Information about DX code updates come from two data sources: DX code listings from the Centers for Medicare and Medicaid Services (CMS formerly HCFA) and the CDC conversion table from the National Center for Health Statistics (NCHS).

1) DX Code Listings with Descriptions:

Annual revisions of the 15,000+ DX codes are available as downloads from

`www.hcfa.gov/stats/pufiles.htm`

A sample from the most recent download, 9V19DIAG.TXT, is shown in Table 1. The codes are identical in format to the ones displayed in Figure 1 except that the decimal points have been removed.

Table 1. Diagnostic codes for SALMONELLA from HCFA download 9V19DIAG.TXT.

Diagnostic Code	Description
003	**OTH SALMONELLA INFECTION***
0030	SALMONELLA ENTERITIS
0031	SALMONELLA SEPTICEMIA
0032	**LOCAL SALMONELLA INFECT***
00320	LOCAL SALMONELLA INF NOS
00321	SALMONELLA MENINGITIS
00322	SALMONELLA PNEUMONIA
00323	SALMONELLA ARTHRITIS
00324	SALMONELLA OSTEOMYELITIS
00329	LOCAL SALMONELLA INF NEC

Obsolete codes with an asterisk appended to their labels identify *hierarchical* update types discussed in the next section.

The initial digit '9' in 9V19DIAG.TXT references ICD-9-CM or *International Classification of Diseases, Ninth Revision, Clinical Modification*, and V19 for Version 19 obliquely references fiscal year 2002.

Unfortunately CMS does not retain previous code releases on its HCFA website, but with some programming the more verbose CDC version of the codes can be translated into HCFA look-alikes. CDC DX code listings dating back to 1986 can be downloaded from the NCHS website at

www.cdc.gov/nchs/icd9.htm

2) The Conversion Table:
The CDC conversion table essential for identifying DX code update patterns provides the second and most important source of information about code updates. It can be found together with the most recent DX code listing at the NCHS website.

A sample listing of scattered entries from the conversion table appears in the ***Example*** region of Table 3. According to header comments in CNVTB02.RTF current new codes and their previously assigned equivalents are listed together with the date of conversion. Fiscal year ranges in the most recent download extend from October 1, 1986 to October 1, 2001. An entry date of October 1, 2001 means the code mapping cannot be altered until October 1, 2002.

An Exhaustive listing of Code Update Patterns

Nine distinct update patterns from the conversion table are listed in Table 2. Specific examples are also presented to explain what is going on.

Table 2. Identified DX code update patterns with examples.

Update Pattern	Example		
	Curr Code	Eff Oct1	Prev Code
❶1 PREV,1 CURR	007.5	2000	007.8
❷Code Del (*)	305.1	1994	305.10,305.11, 305.12,305.13*
❸Code Identity	041.04#	1997	041.04
❹Compound Maps	690.11	1995	691.8&704.8
❺M PREV,1 CURR	V45.79	2000	255.8,289.59, 388.8,569.49, 577.8,V45.89
❻Matched M-M	645.**1**0-645.**1**1, 645.**1**3	2000	645.**0**0-645.**0**1, 645.**0**3
❼1 PREV,M CURR	783.40-783.43	2000	783.4
❽Pure ADD	995.7	2000	None
❾Unmatched M-M	654.90-654.94		654.2,654.9

❶1 PREV -> 1 CURR: One-to-one cardinality is present with one previous code mapping to one current code. The update may or may not be hierarchical. PREV codes may or may not be obsolete.
❷CODE DELETION(*). PREV codes with a terminating asterisk in the conversion table are physically removed from the database. In this instance HCFA changed its ruling. Code 305.1 (TOBACCO USE DISORDER) formerly obsolete was reinstated as a current code in 1994. The subordinate codes were then removed.
❸CODE IDENTITY(#) indicates that descriptions are altered without codes being changed. Code 041.04 has already been described in the *Problem Definition* section.

Table 2 (contd)

❹For a COMPOUND MAPPING to occur, all PREV codes must be recorded in a single patient hospitalization.
❺MANY PREV->1 CURR. Multiple codes are mapped to a single successor. This process is just the opposite of the conventional trend towards increased refinement.
❻MATCHED MANY->MANY. Only one (internal) digit in the PREV codes is changed in this mapping. The code mapping structure remains in tact.
❼1 PREV->MANY CURR captures hierarchical mapping for increased refinement.
❽PURE ADD. No mapping takes place, because there are no PREV codes.
❾UNMATCHED MANY->MANY. Multiple codes map to other multiple codes. Tracing individual mappings is not possible.

Usable Update Code Patterns: a subset of the Exhaustive List

Update patterns amenable to code mapping are those that that point back to a single ancestor baseline code. An intervening PURE ADD with no baseline code is simply not counted. All code maps in Figures 1 and 2 are single ancestry maps.

In Table 3, *usable* update patterns are highlighted and reasons for rejecting the unusable patterns are also presented. While four out of nine update patterns are rejected, they only account for seven percent of the entries in the conversion table.

Table 3. Usable update patterns that translate into code mappings for a transaction data set are highlighted. They account for approximately 93% of the entries in the CDC conversion table.

Update Pattern	FREQ	%	Cum Freq	Cum %
1)1 PREV,1 CURR	345	**52.04**	345	52.04
❷Code Del(*)	2	0.30	347	52.34
3)Code Identity	5	**0.75**	352	53.09
❹Compound Maps	35	5.28	387	58.37
❺M PREV, 1 CURR	8	1.21	395	59.58
6)Matched M-M	10	**1.51**	405	61.09
7)1 PREV, M CURR	240	**36.20**	645	97.29
8)Pure ADD	15	**2.26**	660	99.55
❾Unmatched M-M	3	0.45	663	100.00

❷,❺,❾ There is no way to identify single ancestor codes with these update patterns.
❹Compound maps are rejected, because multiple passes are required to detect all required codes in a patient record.

Classification of the Usable Update Patterns: Hierarchical and Lateral Mapping Types

The remaining usable conversion table entries can be categorized as either *hierarchical* where predecessor codes are obsolete or *concurrent* where predecessor and successor codes exist side-by-side. The hierarchical type of entry is fully described in the setup for Figure 1 at the beginning of the paper, and it is also well documented in the literature (see Scerbo et. al., p. 11). Concurrency, on

the other hand, is not described anywhere even though 206 out of the 465 usable CDC code conversions are concurrent. For this reason we focus exclusively on the less tractable concurrent entries in this section. They typically lead to *lateral mapping*s among sibling codes without an obvious change in diagnostic refinement. The connection between the concurrent code 518.81 and its successors, 518.83 and 518.84, in Figure 2 is an example of lateral mapping.

Lateral Mapping for Concurrent Codes

Diagnoses don't always go from general to more specific. Even at the most general Diagnostic Category level the more precisely defined:

Category	Description
001-139	INFECTIOUS AND PARASITIC DISEASES
140-299	NEOPLASMS

coexists with:

Category	Description
790-796	NONSPECIFIC ABNORMAL FINDINGS
797-799	ILL-DEFINED AND UNKNOWN CAUSES OF MORBIDITY AND MORTALITY

Furthermore, the lack of specificity is repeated at lower code levels with such descriptors as NEC for "Not Elsewhere Classifiable", NOS for "Not Otherwise Specified", OTH for "other", UNS for "unspecified", and NSPCF for "nonspecific". NEC codes are included in the list, because they are used "only when the coder lacks the information necessary to code the term to a more specific category" (ICD-9CM Preface (FY01), 4). Needless to say these catchall codes wreak havoc on a hierarchical structure of increasingly refined diagnoses, but their flexibility is needed for generating a coding scheme that accommodates all degrees of specificity.

The catchall codes are also used to accommodate asynchronous updates where additional refinements in diagnoses are made at a later time. The following example is drawn from the CDC conversion table:

Current Code	Effective Date	Previous Code
V42.89	1997	**V42.8**
V42.81-V42.83	1997	V42.8
V42.84	**2000**	**V42.89**

While the 1997 updates are hierarchical in structure, code V42.89, five alphanumeric characters in length, is really a synonym for the shorter V42.8. When V42.84 is added later in 2000, it is generated from the coexistent, catchall code V42.89 not obsolete V42.8. Descriptors from the latest HCFA download explain what is going on:

V42.8 TRANSPLANT STATUS NEC*****
V42.81 TRNSPL STATUS-BNE MARROW
V42.82 TRSPL STS-PERIP STM CELL
V42.83 TRNSPL STATUS-PANCREAS
V42.84 TRNSPL STATUS-INTESTINES
V42.89 TRNSPL STATUS ORGAN **NEC**

While 139 out of the 206 coexistent previous codes in the conversion table contain character strings that put them into the catchall category, 67 do not. These 67 codes are difficult to track, because sometimes they mask label changes. Again from the conversion table:

Current Code	Eff Date	Previous Code
E967.2	1996	E967.0
BATTER BY MTH|STPM		BATTER BY FATH|STPF

From the mapping it would appear that female batterers evolved from their male counterparts. However, prior to 1996, E967.0 stood for

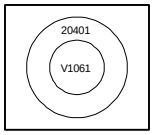
CHILD ABUSE BY PARENT

Since the term "parent" is less specific than "mother" or "father" this code update represents a second way to introduce increased specificity into the coding structure. However, now there is no way to specify a parent without having to identify that person as a "mother" or "father".

Finally, there are instances where increased refinement has to be assumed from the context of the update rather than by the presence of catchall phrases or masked labels. For example, the conversion table entry:

Current Code	Eff Date	Previous Code
20401	1991	V1061
ACT LYM LEUK W RMSION		HX OF LYMPHOID LEUKEMIA

shows that V1061 is subsumed by the more highly refined DX code, 20401. A Venn diagram captures the relationship better than a hierarchical tree structure.



Because the structure of the lateral updates just described is not based on a thorough examination of the relevant codes, we cannot be sure that all possible update scenarios have been described. Nevertheless, each of the scenarios identified so far is repeated multiple times in the conversion table. A coding specialist could probably broaden our understanding of the update process, but it would also be helpful to have a report that correctly labels previous codes in a conversion table. Such a report is constructed from an application of format concatenation the first technique described in the next section.

Techniques for Managing Code Updates

Format concatenation for time dependent code descriptions and *code mapping* for uniform code assignments across a pre-defined span in time are described in this section. The concatenated formats are constructed from the HCFA downloads whereas the CDC conversion table is used for creating a code map. The applications, however, use both update sources for input. Format concatenation is used for labeling usable conversion table entries, and raw codes in a patient hospitalization data set are labeled and validated with an application of format concatenation prior to code mapping.

Technique #1: Format Concatenation

•Definition of Terms

There are no universally accepted terms for describing a format that appears as an argument in a label of another format. Therefore, in this paper *format concatenation* is reserved for a format where another *related user-defined* format is assigned specifically to the OTHER range. The relationship among concatenated formats can always be depicted as a set of linked nodes shown in Figure 5. These formats are also known here as *appended* formats.

Appended formats are not the same as *nested* formats. The following example of a nested format is taken from the SAS Procedures Guide, Version 8:

```
proc format;
  value benefit
    low-'31DEC79'd = [worddate20.]
    '01JAN80'D-high = '** Not Eligible **';
run; (467)
```

WORDDATE is nested or contained within BENEFIT, because it uses starting *ranges* from BENEFIT. Appended formats, on the other hand, are separate entities containing both defined *ranges* and *values*. The more generic term *embedded* is reserved for both *nested* and *appended* formats.

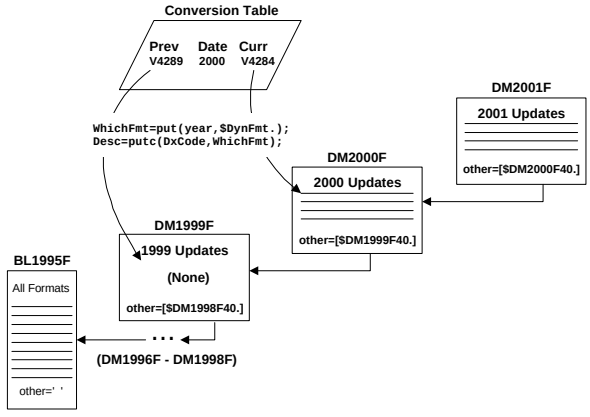
The following example of format concatenation comes from Shoemaker's paper cited in the Reference Section:

```
proc format;
  value $MYLOB
    'MD' = 'Medicaid' other = [$LOB12.];
  value $LOB
    'CO' = 'Commercial' 'MC' = 'Medicare'
    'SF' = 'Self-Funded' other = 'Unknown';
run; (105)
```

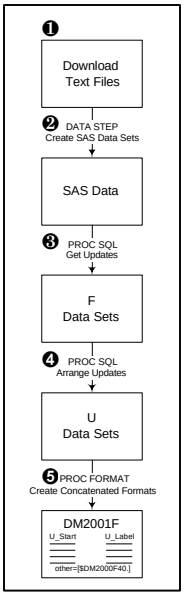
In the appended format, $MYLOB, 'MD' is the *range*, and 'Medicaid' is the *value* associated with the range. Shoemaker also states that overlapping range errors will not be generated when the same range is re-defined in a subsequent format. This feature makes it possible to reassign different labels to the same DX code over time. Note too that a *baseline* format is always part of format concatenation. It can be identified by the absence of an appended format assignment to the OTHER range. Finally, it should be noted that embedded formats are delimited by square brackets rather than quotation marks. Format lengths must also be specified within the brackets.

In Figure 5 year-stamped DX codes from the CDC conversion table are formatted by a link to the correct descriptive mapping (DM) format having a matching year value in its name. The linkage is handled by a *dynamic* format that works with the PUTC function at run-time to return the correct starting format for initiating a search. If a match is not found in the first linked format, the appended formats are traversed in order by moving backwards in time. Eventually, in the absence of an update, the baseline format, BL1995F, is applied, and if there is still no match, a missing value is returned. Figure 5 shows how the appended formats are traversed.

Figure 5. A schematic for format concatenation



•Constructing Appended Formats

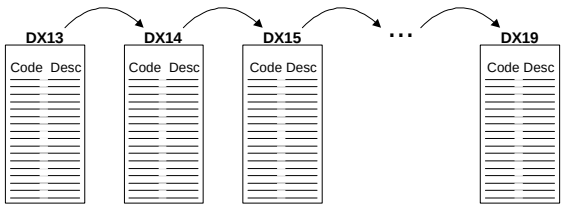


To create the series of appended formats, it is necessary to complete the tasks listed in the adjacent flowchart. This flowchart differs from an earlier version found in a related paper (Watts, 681). Currently, the starting point is the set of downloaded HCFA text files whereas the earlier version contained a double conversion from an initial set of SAS formats to control-out data sets and then back to a corresponding set of concatenated formats. Also an entire branch of instructions from the earlier flowchart can be eliminated, because the maximum length required for appended format definitions is obtained when the downloaded text files are processed. Before, label lengths were obtained in a separate process from the FORMATS catalog. What remains, therefore, is a simpler design with a single, linear path.

❶,❷ Input data sets are created from versions 13 to 19 HCFA downloads for years 1995 to 2001. The record layout is shown in Table 1 with a 5-character DX code followed by a 25-character descriptive label.

❸ Annual updates are obtained from two-by-two comparisons of the input data sets. The structure of the comparisons is depicted in Figure 6. Note here that the second data set in the initial two-by-two comparison becomes the first one in the next comparison. PROC SQL is used to process the data.

Figure 6. Obtaining yearly updates from HCFA downloads.



The SQL code below culls updates from the first two input data sets and stores the results in data set, F14:

```
create table F14 as
  select Early.DxCode as EStart,
         Early.DxDesc as ELabel,
         Late.DxCode  as LStart,
         Late.DxDesc  as LLabel
  from DX13 Early full join DX14 Late
  on Early.DxCode EQ Late.DxCode
  where Early.DxDesc NE Late.DxDesc
;
```

Data Set F14

Obs	ESTART	ELABEL
1	V428	TRANSPLANT STATUS NEC
2		
Obs	LSTART	LLABEL
1	V428	TRANSPLANT STATUS NEC*
2	V4289	TRNSPL STATUS ORGAN NEC

A full outer join in the SQL command retrieves both deleted and new codes whereas the selection criteria listed in the ON and WHERE clauses returns updated labels. A left join alone would return deleted codes whereas a right join would capture new codes. Repeating the SQL command from inside a macro generates all tables: F14 to F19. The numeric extension on the name references the latter of the two tables being compared - mirroring how the update process actually works. For example, F19 for 2001 captures the changes between the 2000 and 2001 code downloads. Note too that there is no table F13. The baseline format, BL1995F, with every code from the 1995 download is used instead. The baseline format, in other words, plays the role of *base case* in a recursive process.

❹ At this point the four variables in the F* data sets need to be compressed into variables START and LABEL used in the concatenated formats. A value for YEAR also has to be added, since the data sets will be appended to each other before they are changed into formats. The UNION set operator is used to differentiate processing for the three types of update:

```
create table U14 as /*U for update*/
/*additions*/
  select LStart as Start, LLabel as Label,
         "1996" as year
  from F14
  where LStart NE ' ' AND EStart EQ ' '

union
/*deletions*/
  select EStart as Start, "deleted" as Label,
         "1996" as year
  from F14
  where EStart NE ' ' AND LStart EQ ' '

union
/*modifications*/
  select LStart as Start, LLabel as Label,
         "1996" as year
  from F14
  where LStart EQ EStart AND LLabel NE ELabel
;
```

Data Set: U14

Obs	Start	Label	year
1	V428	TRANSPLANT STATUS NEC*	1996
2	V4289	TRNSPL STATUS ORGAN NEC	1996

Deletions must be labeled with a constant such as *deleted* so that the prior undeleted label for a code is not erroneously retrieved in a given transaction.

Again all the U* data sets are created in a macro with the value for YEAR being supplied by a macro variable. At the end of this step, U* data sets are appended to form ALLUPDATES, and ALLUPDATES is sorted by descending YEAR and within YEAR by ascending START (or DX code value).

❺ Two data sets are needed for the final assembly with CNTLIN. The first, MKFMT, comes directly from the appended data set, ALLUPDATES, and it contains all the updated codes with their time-delimited descriptions:

```
data MkFmt(keep=fmtname type start label);
  length label $&dxLen fmtname $8;
  retain type 'C';
  set AllUpdates;
    by descending year;
  fmtname = 'DM'||year||'F'; /*DM=Desc Map*/
run;
```

The second data set, MKHLO, uses a looping structure to generate one record for each year:

```
data MkHLO(keep=fmtname type start label hlo);
 length label $&dxLen fmtname $8;
 retain type 'C' start ' ';
 do year = 2001 to 1995 by -1;
  fmtname = 'DM'||put(year,4.)||'F';
  HLO = "OF";
  if year gt 1996 then /*1996fmt pts to BL*/
   label = 'DM'||put(year-1,4.)||"F&DxLen..";
  else
   label = "BL1995F F&DxLen..";  /*Baseln Fm*/
  output;
 end;
 stop;
run;
```

The operations here have to be divided between two data steps to accommodate years when no updates occur. In fact, codes were not updated in 1999, because HCFA anticipated Y2K problems. The format for that year, therefore, contains a single entry:

```
Start       End         Formatted Value
**OTHER**   **OTHER**   $DM1998F25.
```

In the MKHLO data set, HLO is assigned a two-character value "OF". O means that the range is OTHER and F references a FORMAT or INFORMAT. The argument for LABEL is a string, and the square brackets have been removed. Therefore, the format equivalent to the data step assignment is

```
OTHER=[$BL1995F25.]
```

The macro variable DXLEN in the two data sets above resolves to 25, the maximum length of the variable DESC in all the HCFA downloads. Finally, the data sets are combined to form DESCMAP, the control-in data set used as input to PROC FORMAT.

From Data Set DESCMAP

fmtname	type	HLO	Start	label
DM1997F	C		V428	TRANSPLANT STATUS NEC*
	C		V4289	TRNSPL STATUS ORGAN NEC
		OF		DM1996F25.

DESCMAP contains multiple format definitions grouped by values for FMTNAME. For additional comments about creating multiple formats from a single control-in data set, see Shoemaker, page 106.

•The Dynamic Format

The dynamic format $YRDYNF is created in a separate process with the application of a macro:

```
%macro runTime;
   proc format library=library;
     value $YrDynF
       "1995" = "$BL1995F"
     %do year = &minDMYr %to &maxDMYr;
       "&year" = "$DM&Year.F"
     %end;
       other="Time ERROR"
     ;
   run;
%mend runTime;
%runTime;
```

The $YRDYNF format and the PUTC function are both used to link the data-step value for YEAR with its associated starting format at run-time. Note that the label in a dynamic format is a text-string not an embedded format.

Late binding resolution at run-time is made possible by the PUTC (or PUTN) function and not by any special property of the format itself. Appended formats on the other hand are static. The links are fixed at compile-time.

•Application: Annotating the CDC conversion table

An annotated conversion table is necessary for understanding what is going on with the lateral mapping updates described earlier in the paper. Such a table eliminates the need for manually searching multiple HCFA text files to track code changes. Proper tracking involves knowing exactly how codes are defined over a period of time.

With hierarchical mapping, it is easy to figure out what a PREV code originally stood for. For example, V42.8 changed from TRANSPLANT STATUS NEC to TRANSPLANT STATUS NEC* in 1997. Only an asterisk for obsolescence was added to the definition. However, as mentioned previously, E9670 not only generated E9672 for BATTER BY MTH|STPM in 1996 but its definition changed from CHILD ABUSE BY PARENT to BATTER BY FATH|STPF as well.

Because of space constraints, no instructions are provided in this narrative about how to re-configure the subset of usable mappings in the code conversion table to the layout used for input in the data step below. All CURR code-mapping ranges in the input data set have been parsed so that single records are produced for every CURR and PREV code combination in the table.

```
data demo(keep=currYrPrev code dsc when);
  length curr prev code $5;
  length prvDscNow prvDscBef currDsc dsc $25;
  length year yrLess1 when $4;
  length nowFmt beforeFmt $8;
  length currYrPrev $16;
  array _code{3} $5 curr prev prev; ❶
  array _dsc{3} $25 currDsc prvDscNow
                    prvDscBef;
  array _when{3} $4 year year yrLess1;❶

  input curr year prev;
  nowFmt=put(year,$yrDynF.);❷
  if(index(nowFmt,'ERROR')) eq 0 then ❸
    do;
      prvDscNow=putc(prev,nowFmt);❷
      currDsc=putc(curr,nowFmt);❷
    end;
  yrLess1=put(input(year,4.)-1, 4.);
  beforeFmt=put(yrLess1,$yrDynF.);❹
  if(index(beforeFmt,'ERROR')) eq 0 then ❸
    prvDscBef=putc(prev,beforeFmt); ❹
  currYrPrev=trim(curr)||'-'||year||'-'||prev;
  do i=1 to 3;
    code=_code[i]; dsc=_dsc[i]; when=_when[i];❶
    output;
  end;
cards; ❺
  E9672 1996 E9670
  51883 1998 51881
  51884 1998 51881
  64510 2000 64500
run;
```

❶ One entry for CURR and two for PREV are posted. The first for PREV displays the first post-mapping description. The second earlier label shows what PREV stood for prior to update. Arrays are more manageable here than PROC TRANSPOSE, because they can accommodate repeated variable assignments in their definitions.

❷ The dynamic format and PUTC function work together to identify the correct starting format in the series of concatenated formats. Note, however, there is NO format traversal in this particular invocation of $YRDYNF. Updates in the starting appended format should correspond exactly to the mappings in the CDC conversion table.

❸ Error checking is built into the dynamic format to make sure that the data step value for YEAR is within range. Only code conversions from 1996 to 2001 are processed in this application. 1995 is defined as the baseline year so that former descriptions for 1996 PREV mappings can be retrieved.

❹ Formats are now traversed until the first instance of a PREV code prior to code mapping is found.

❺ A sample of entries from the modified conversion table is selected for full annotation. Some are already depicted graphically in Figure 2.

Figure 7. Selected entries from the annotated conversion table. Annotation is made possible by an application of format concatenation described in this section.

Curr- Year- Prev	Code	Description	When
51883-1998-51881	51883	CHRONIC RESPIRATORY FAIL	1998
	51881	ACUTE RESPIRATRY FAILURE	1998
	51881	RESPIRATORY FAILURE	1997
51884-1998-51881	51884	ACUTE & CHRONC RESP FAIL	1998
	51881	ACUTE RESPIRATRY FAILURE	1998
	51881	RESPIRATORY FAILURE	1997
64510-2000-64500	64510	POST TERM PREG-UNSP	2000
	64500	deleted	2000
	64500	PROLONGED PREG-UNSPEC	1999
E9672-1996-E9670	E9672	BATTER BY MTH/STM	1996
	E9670	BATTER BY FATH/STEPFTH	1996
	E9670	CHILD ABUSE BY PARENT	1995

Technique #2: Code Mapping

Two major activities are involved with *code mapping*, used for generating uniform code assignments across a predefined span in time. The first, *code reassignment,* occurs when usable, *threaded* PREV values from the code conversion table match codes in the baseline HCFA download. Threading replaces any intermediate mappings with the oldest ones. For example

Current Code	**Effective Date**	**Previous Code**
V42.89	1997	V42.8
V42.84	2000	V42.89
becomes:		
V42.89	1997	**V42.8**
V42.84	2000	**V42.8**

This means that both V42.89 and V42.84 are mapped to V42.8. Most of the time, however, threads contain only one final mapping from the conversion table.

Map-types also need to be added to the descriptive label of a converted code. This part of the process uses **all** the entries in the conversion table, not just the usable ones. Transaction files contain a full gamut of codes. Therefore, it is possible that all mapping indicators displayed in Figure 8 will appear in the output.

Figure 8. Code Map-Type Indicators

Indicator	Type	Mapping Action
None	No mapping	No reassignment
*	Hierarchical	Reassignment
**	Lateral	Reassignment
!	Unusable	No reassignment

Remember, code mapping is only feasible when a single ancestor code can be identified without ambiguity.

CURR codes with a *many-to-one* relationship to PREV codes can be mapped:

```
Current Code     Eff Date  Previous Code
783.40-783.43      2000    783.4
```

CURR codes with a *one-to-many* relationship to PREV codes cannot be mapped:

```
Current Code     Eff Date  Previous Code
V45.79             2000    255.8,289.59,388.8,
                           569.49,577.8,V45.89
```

Fortunately a format can be used to return a single (PREV) *label* or code for many (CURR) *start* values.

•Constructing the CMAPF format for Code Reassignment

Like the format concatenation annotation example, usable entries from the code conversion table are modified so that a separate record is generated for each code within a CURR range. The modified data set SMAP then comes into THREADS below with a *descending* sort on YEAR. With the direction of the sort, later records are processed first. This way the data step "looks ahead" with multiple SET statements suggested by Ron Cody in *Longitudinal Data and SAS: A Programmer's Guide* (114-116). Looking ahead, in this instance, however, involves going backwards in time!

```
data threads(keep=thread curr prev year);
  array rev{&nobs} $1 rev1-rev&nobs; ❶
  retain rev1-rev&nobs ' ';
  retain srchcode;
  do i=1 to totobs;
    set sMap point=i nobs=totobs;❷
    if i=totobs then ❸
     do;
       if rev[i] eq ' ' then
        do;
          thread+1;
          output; ❹
        end;
        stop;
     end;
    else
     do;
       if rev[i]=' ' then
        do;
          srchcode=prev;
          thread+1;
          output; ❹
          iplus1=i+1;
          do j=iplus1 to totobs;
            set SMap point=j nobs=totobs; ❷
            if rev[j] eq ' ' and
            curr eq srchcode then
             do;
               rev[j]='*';
               output; ❹
               srchcode=prev;
             end; /*IF REV[J]*/
          end; /*J*/
        end; /*IF REV[I]*/
     end; /*ELSE DO*/
    end; /*I*/
  run;
```

❶ REV[ ] for *reviewed* stores information from subsequent SET statements in the data step.
❷ Multiple SET statements are highlighted.
❸ The last record may or may not reference a new thread.
❹ All intermediate mappings are generated in this data step. Subsequent executions of PROC SQL replace intermediate mappings with final ones.

After intermediate mappings have been replaced with final ones, the THREADS data set is transformed into the $CMAP format with CURR and PREV codes being renamed as START and LABEL respectively. This way, V4284 can be mapped to its earliest predecessor, V428 by applying a format:

```
mapped_code = put(raw_code,$cmapf.);
```

•Constructing the CLMAPF format for map-typing DX codes

The CLMAPF format for map-typing is constructed by applying a set of SQL commands to baseline HCFA descriptive labels for downloaded codes:

```
proc sql noprint;
 create table CLMap as ❶
 select coalesce(RejStrt,dxcode) as start,
        coalesce(RejLabel,dxdesc) as label
 from data.BL1996 full join data.parsedRejects
 on dxcode = RejStrt and year gt "1996";

 create table CLMapF as ❷
 select distinct CL.start,
  left(trim(put(coexist,$CoEFmt.)))||
   CL.label as label
 from CLmap CL left join data.dxCodeMap CM
 on CL.start EQ CM.label;
quit;
```

❶ Descriptions from the HCFA 1996 download are replaced with unusable mappings(!) when indicated.
❷ Map indicators (*,**) from an application of $COEFMT on COEXIST in DXCODEMAP are pre-pended onto HCFA download descriptions.

Labels produced by an application of $CLMAPF are seen in the application that follows, so output from the SQL procedure is not shown here. To summarize just two formats are needed for code mapping: $CMAPF maps derivative codes to their earliest predecessors and $CLMAPF identifies the mapping type through label modification.

•Application: Map-Typing and tallying cross-year frequencies for Hospitalized Patients

Both format concatenation and code-mapping techniques are used to generate the HOSP data set. Format concatenation detects errors and retrieves the original description, ODESC, for a DX code. Code mapping produces uniform codes with associated mapping labels. The output along with additional annotation is shown in Figure 9, and the variable COUNT is summarized in Figure 10.

```
data hosp(keep=eId year rawDxCode mapDxCode
               count oDesc mapDesc);
 length dx1-dx5 $5;
 length rawDxCode mapDxCode $5;
 length thisFmt $8;
 length year $4
 length oDesc $25;
 length mapDesc $60;
 array dx $5 dx1-dx5;
 infile cards missover;
 input year n dx1-dx5; /*n=#codes per pt*/
 eId=_n_; /*eId=Patient Episode ID*/
 thisFmt=put(year,$yrDynF.); ❶
```

```
if(index(thisFmt,'ERROR')) eq 0 then
 do i=1 to n;
  rawDxCode=dx[i];
  oDesc=putc(rawDxCode,thisFmt); ❶
  if substr(ODesc,length(ODesc)) eq '*' or
  ODesc eq ' ' then ❷
   count=0;
  else
   count=1;
  MapDxCode=put(RawDxCode,$CMAPF.); ❸
  MapDesc=put(MapDxCode,$CLMAPF.); ❸
  output;
 end;
cards;
    1998 4 4824 48241 04104 3888
    2001 2 0078 0075
    2000 1 V4579
    2000 1 00321
run;
```

❶ Format concatenation is used for detecting obsolescence and for assigning time-delimited descriptions to the variable ORIGDESC.
❷ If a code is obsolete(*) or non-existent in the given time period, COUNT is not incremented.
❸ Codes are mapped, then labeled.

Figure 9. Mapped HOSP Data Set with a few abbreviations

O	eId	Year	Raw Code	Map Code	Count	OrigDesc
1	1	1998	4824	4824	0	STAPH PNEUM*
2	1	1998	48241	4824	1	STAPH AUR PNEUM
3	1	1998	04104	04104	1	ENTERO GROUP D
4	1	1998	3888	3888	1	DISOR OF EAR NEC
5	2	2001	0078	0078	1	PROTOZ INTEST DIS
6	2	2001	0075	0078	1	CYCLOSPORIASIS
7	3	2000	V4579	V4579	1	ACQ ABS ORGAN NEC
8	4	2000	00321	00321	1	SALMONELLA MENING

O	eId	MapDesc
1	1	*STAPHYLOCOCCAL PNEUMONIA
2	1	*STAPHYLOCOCCAL PNEUMONIA
3	1	STREPTOCOCCUS GROUP D
4	1	!V45.79->255.8,289.59,388.8 etc (2000)
5	2	**PROTOZOAL INTEST DIS NEC
6	2	**PROTOZOAL INTEST DIS NEC
7	3	!V45.79->255.8,289.59,388.8 etc (2000)
8	4	SALMONELLA MENINGITIS

Annotations are referenced by OBS (**O**) number:
❶ The raw code is obsolete in 1998. Therefore, COUNT set to 0. Hierarchical mapping is involved.
❷ The raw code is valid. Therefore, COUNT is set to 1. Hierarchical mapping is involved.
❸ The raw code is valid. There is NO code mapping, but a coding specialist should be consulted to see if there is actually a change in diagnosis.
❹ The raw code is valid, but it cannot be mapped (unusable PREV code).
❺,❻ The same patient episode yields a duplicate mapping.
❼ The raw code is valid, but it cannot be mapped (unusable CURR code).
❽ No code mapping. No change in descriptive label.

When code 0075 is mapped to 0078 in the HOSP data set, a duplicate code is generated for the same hospitalization. Output from PROC TABULATE in Figure 10 shows that totals are affected by the removal of such duplicates. Similar inconsistent results are also displayed graphically in Figure 4.

Indications for Code Mapping

Code mapping for cross-year frequencies is not recommended for very large data sets containing millions of records that summarize patient hospitalizations. An additional set of mapped codes would have to be created for each patient record. One way to circumvent this problem is to exclude any codes that change during the time on study. If patients were followed from 1996 to 2002, for example, frequencies for 473 out of the 15,361 codes in the most recent HCFA download could not be calculated.

If, on the other hand, a study is very focused and fairly small, all codes could easily be mapped, and a coding specialist might be able to transform additional unusable mappings into usable ones that reflect special characteristics of the study population. Yet another alternative is to bypass the transaction files all together and store multiple mappings of health codes in software applications that manipulate longitudinal data. Code ranges in staging software, for example, should be able to accommodate patient data entered over a period of years.

Figure 10. HOSP Data Frequencies

Code Type	Dx Code	Freq
1) Raw	4824	0
	48241	1
	04104	1
	3888	1
	0078	**1**
	0075	**1**
	V4579	1
	00321	1
	All	**7**
2) Mapped	Dx Code	
	4824*	1
	04104	1
	3888!	1
	0078**	**1**
	V4579!	1
	00321	1
	All	**6**

Cross-year frequencies for patient data yield different totals depending on how codes are defined. The mapping type is affixed to the mapped code in this listing.

Future Directions: Issues with ICD-10-CM

According to a private communication from Joyce M. Frazier, Medical Systems Specialist affiliated with the CDC,

the ICD-10-CM/ICD-9-CM diagnoses crosswalk has not yet been developed, but the ICD-10-CM diagnostic codes are in the final review stage of development. Nevertheless, an ICD-9<>ICD10 *Translator* has been developed for mortality codes by the World Health Organization (WHO) that incorporates six of the nine update patterns from the conversion table summarized in Table 2. Table 4 below lists the *Translator* relationships with the corresponding update patterns from Table 2. Unfortunately, it appears that there will be some difficulty finding ICD-9 predecessors for all the newly defined ICD-10 codes.

Table 4. ICD-10 Mortality code *Translator* relationships matched to Update Patterns for ICD-9 Code Revisions

ICD-9 Update Pattern	Can Map?	ICD-10 Relationship
❶1 PREV,1 CURR	Y	Both ICD-9 and ICD-10 are unique
❷Code Del (*)	N	ICD-9 code is discontinued in ICD-10
❸Code Identity		
❹Compound Maps		
❺M PREV,1 CURR	N	ICD-10 code unique, but ICD-9 is not
❻Matched M-M		
❼1 PREV,M CURR	Y	ICD-9 code unique, but ICD-10 is not
❽Pure ADD	Y	ICD-10 code does not exist in ICD-9
❾Unmatched M-M	N	Neither code unique

If the trend towards increased refinement in diagnoses continues into ICD-10, the workable update pattern ❼ from Table 4 should again be observed more frequently than pattern ❺ in the final version of the DX cross-walk table. There is some corroboration for this trend in the article *New International Classification of Diseases (ICD-10): The History and Impact* cited in the Reference section. Table 2 from the article shows *comparability ratios* for leading causes of death calculated by coding the 1996 national mortality file once in ICD-9 and then again in ICD-10. Ten out of the 15 leading causes of death are assigned a ratio greater than 1.0 indicating that the number of codes in these categories has continued to increase.

Summary and Conclusions

The aim of this paper has been to identify ICD-9 code update patterns so that results from longitudinal data analyses can be accurately reported. Update patterns are categorized as *usable* where a recently modified code is unequivocally mapped to a single ancestor code, and as *unusable* otherwise. Two techniques have also been described for handling the usable updates. *Format concatenation* retrieves time-dependent ICD-9 DX code descriptions, and *code mapping* produces uniform, comparable code assignments across a pre-defined span of time. Applications are provided to round out the definitions of the techniques. An annotated CDC conversion table is constructed with an application of format concatenation, and frequencies are calculated with a very small hospital episode data set illustrating all-possible update scenarios.

What has impressed us the most about completing this assignment is the complexity of the update process. Almost half of the usable updates defy conceptualization and are better described as *lateral* instead of *hierarchical*. The problem also does not look like it will disappear with the adoption of the ICD-10 standard. If the *comparability ratio* is understood correctly, all ICD-10 codes for longitudinal data analysis recorded during the cross-over period of time will need to be mapped to their ICD-9 predecessors. When code assignments are adjusted and duplicate codes are removed, the comparability ratio should return to an expected value of 1.0.

References and Citations

Cody, R. *Longitudinal Data and SASÒ: A programmer's Guide*. Cary, NC: SAS Institute, Inc., 2001.

Colorado Department of Public Health and Environment. *New International Classification of Diseases (ICD-10): The History and Impact*. http://www.cdphe.state.co.us/hs/ March, 2001. No. 41.

Frazier, Joyce M. Email To: awilson@umdnj.edu. Subject: *ICD-10 to ICD-9 Crosswalk*. 24 April 2002.

SAS Institute Inc., 1999. SAS® *Procedures Guide: Reference, Version 8*. 2 vols. Cary, NC: SAS Institute, Inc.

Scerbo, M., C. Dickstein, and A. Wilson. *Health Care Data and the SASÒ System.* Cary, NC: SAS Institute Inc., 2001.

Shoemaker, J. *Advanced Techniques to Build and Manage User-Defined SASÒ FORMAT Catalogs*. Proceedings of the 11th Annual NorthEast SAS® Users Group Conference. Pittsburgh, PA, pp. 102-107, 1998.

Watts, P. *Using Format Concatenation in SASÒ Software to Decode Data in Longitudinal Studies.* Proceedings Of The 12th Annual Northeast Sas Users Group Conference. Washington, D.C., pp. 680-686, 1999.

Wilson, A. C. and M. Scerbo. *Dealing with Health Care Data using the SASÒ system*. Proceedings of the 11th Annual NorthEast SAS® Users Group Conference. Pittsburgh, PA, pp. 118-123, 1998.

Contact Information

Perry Watts
wattsp@dca.net

Alan C. Wilson
awilson@UMDNJ.EDU