Paper 1622-2014

SAS® Solutions to Identifying Hospital Readmissions

Weifeng Fan and Maryam Sarfarazi, UMWA Health and Retirement Funds, Washington, DC

ABSTRACT

Hospital readmission rate has become a key indicator for measuring the quality of health care and is currently adopted by major healthcare stakeholders including the Centers for Medicare & Medicaid Services (CMS), the Agency for Healthcare Research and Quality (AHRQ), and the National Committee for Quality Assurance (NCQA). In the calculation of the readmission rate, it is often a challenging task to identify an eligible hospital readmission from the convoluted administrative claims data. By taking advantage of the flexibility and power of SAS® programming tools, this paper proposes three different solutions - in both DATA step and PROC SQL - to help identify hospital readmissions more efficiently and accurately.

INTRODUCTION

Unnecessary, avoidable hospital readmissions have been proved costly and often cause poor health outcomes to patients. Section 3025 of the Affordable Care Act outlines the Hospital Readmissions Reduction Program, which requires CMS to reduce payments to hospitals with excessive patient readmissions. In response, hospitals nationwide are working hard to bring down the number of readmissions, and reporting readmission rate has become a critical step in this important quality initiative.

Hospital readmissions are defined as inpatient admissions by the same person within a specified time period (30, 60, 90, 180 days, or more) following an index hospitalization. This paper will focus on identifying 30-day readmissions, since 30-day readmission rate is a standard rate most often cited by health care organizations and government agencies. There is no uniform definition of readmission rate, but readmission rate within a specific study period can often be calculated as:

$$\text{Readmission Rate} = \frac{Redmissions}{All\ Admissions}$$

where the denominator refers to all admissions including index hospitalizations and readmissions. It is much easier to calculate the denominator since there is no need to differentiate the type of admissions. The calculation of the numerator gets trickier and entails greater caution. For example, consider a sample dataset with a hypothetical patients' admission information shown in Table 1:

Table 1: Example

Patient ID	Admission Date	Discharge Date
001	20130121	20130131
001	20130228	20130301
001	20130607	20130609
001	20130612	20130615
001	20130701 ?	20130709
001	20130912	20130918

How many readmissions should be counted for this patient, or specifically, should the admission on "20130701" be counted once or twice? There might be different approaches to defining readmissions or even index admissions in this case, but a common method is to track each hospital discharge for potential readmission within the following 30-day time frame and look for one, but no more than one, readmission for each discharge. In this approach, an admission can serve as both a readmission for the previous discharge and a potential index admission for the next admission, but each admission can only be counted as a readmission once. Since each index admission can only be exclusively paired with one readmission, the numerator (readmissions) in the readmission rate formula above can be alternatively replaced with index admissions with at least one readmission. This approach is in line with the AHRQ's methodology and the benchmark study of readmission rates by Jencks et al. Based on this method, the admission on "20130701" should only be counted once as a readmission for the discharge on "20130615", and the total number of readmissions in this case should be 3 instead of 4. When a dataset has thousands or even millions of patients, an effective counting algorithm must be developed to identify and tally readmissions efficiently and accurately and this is when SAS comes to help. This paper proposes three different SAS solutions to help accomplish this challenging task.

DATA PREPARATION

Data cleaning is a necessary step before any actual data analysis. Before applying any of the three SAS solutions proposed below, we need to apply some exclusion criteria often required by readmission rate reporting. We only included admissions where patients were admitted to and discharged alive from an acute care hospital. We excluded from the numerator and denominator any admissions that are transfers to another acute care hospital. We counted readmissions on an "all-cause" basis, but we did exclude any readmissions for rehabilitation. The solutions discussed in this paper do not delve into issues such as risk adjustment and planned readmissions, as they are highly customized depending on the level and the purpose of reporting.

After applying these commonly adopted exclusion criteria, we de-duplicated and sorted our claims data by patient ID, admission date, and discharge date so that each line would represent a unique hospitalization stay for a patient and each patient's hospitalizations would be listed chronologically. For programming purpose, we created a sequence number field ("SeqNo") which serves as a line counter for each patient and would be reset to "1" when SAS reads the first line for the next patient. Additionally, both admission dates and discharge dates need to be converted into SAS date values if they have not been so already. All date fields would be assigned a date format "YYMMDD10." and labeled as "Admission Date" and "Discharge Date", respectively. After being sorted by patient ID and sequence number, the cleaned dataset called "Claims" would look like the following (Table 2):

Table 2: Cleaned Sample Dataset (Claims)

Patient ID	SeqNo	Admission Date	Discharge Date
001	1	2013-01-21	2013-01-31
001	2	2013-02-28	2013-03-01
001	3	2013-06-07	2013-06-09
001	4	2013-06-12	2013-06-15
001	5	2013-07-01	2013-07-09
001	6	2013-09-12	2013-09-18

Our next step is to accurately identify and tally the readmissions shown above.

SOLUTION 1: DATA STEP (VERTICALLY)

The first SAS solution employs a special LAG function in DATA step. The LAG function creates a new reference date field ("Ref_Date") by retrieving the value of the discharge date from the previous line. On each line, the gap between the current admission date and the immediate previous discharge date ("Gap") would be calculated and evaluated. A readmission indicator field ("Tag") would be created and if the gap is no more than 30 days, a value of 1 would be assigned to this field. The field "Readmissions" is the cumulative sum of "Tag" and shows how many readmissions for this patient have been tallied up to this point. Each time when SAS first encounters the claims data of a new patient, all four new fields ("Ref_Date", "Gap", "Tag", and "Readmissions") would be reset to missing values, and a fresh round of tallying would start for the new patient.

Solution 1 Codes:

```
Data Solution1;
  Set Claims;
  By Patient_ID SeqNo;
  Ref_date  = LAG(Discharge_Date);   * Retrieve the value of the previous discharge date;
  Format Ref_Date YYMMDD10.;
  Label    Ref_Date = "Reference Date";
  Gap = Admission_Date - Ref_Date;  * Calculate the gap between the current admission date and
                                      the previous discharge date;
  If First.Patient_ID then do;          * Reset the values of the following fields in case of a new patient ID;
     Ref_date       = .;
     Gap            = .;
     Tag            = .;
     Readmissions = .;
   End;
   If 0 <= Gap <= 30 then Tag = 1;      * Identify a readmission and assign value 1 to Tag;
   Readmissions + Tag;                  * Calculate cumulative readmissions;
Run;
```

Note that when SAS executes the statement "Readmissions + Tag ;", it adds the value of "Tag" to the value of "Readmissions" and retains the new value until the next processing of the statement. This running total has to be reset to a missing value when SAS processes the data for the next new patient ID. The output results are shown as follows (Table 3):

Table 3: Solution1 Results

Patient ID	SeqNo	Admission Date	Discharge Date	Reference Date	Gap	Tag	Readmissions
001	1	2013-01-21	2013-01-31	.	.	.	.
001	2	2013-02-28	2013-03-01	2013-01-31	28	**1**	1
001	3	2013-06-07	2013-06-09	2013-03-01	98	.	1
001	4	2013-06-12	2013-06-15	2013-06-09	3	**1**	2
001	5	2013-07-01	2013-07-09	2013-06-15	16	**1**	3
001	6	2013-09-12	2013-09-18	2013-07-09	65	.	**3**

Solution 1 keeps the original data structure intact and processes the claims data sequentially (vertically) based on the fundamental "thinking process" of Data Step. By taking advantage of the power of LAG function and creating a few helping variables, this solution successfully identifies and tallies readmissions without any additional data management.

SOLUTION 2: DATA STEP (HORIZONTALLY)

The second SAS solution views the claims data from a different angle by transposing them based on patient ID. It utilizes the PROC TRANSPOSE procedure in DATA step and transposes two date fields: admission date and discharge date. Since SAS cannot transpose multiple fields from long to wide in one data step, we need to have two PROC TRANSPOSE procedures to transpose the data: one for admission date and the other for discharge date. We then merge these two new datasets based on patient ID, and after this long-to-wide data reshaping there is only one line for each patient ID. In the newly merged dataset, we create two arrays: One ("A_Date") for admission dates and the other ("D_Date") for discharge dates. Note that for each array, the number of array elements is usually no more than 365 (or 366 in a leap year) in a given calendar year. Finally we can compare array elements A_Date{i} with D_Date{j} where j = i – 1, and evaluate the gap between the current admission date and the immediate previous discharge date. Similar to Solution 1, we can also create a readmission indicator field ("Tag") and a cumulative readmission count field ("Readmissions").

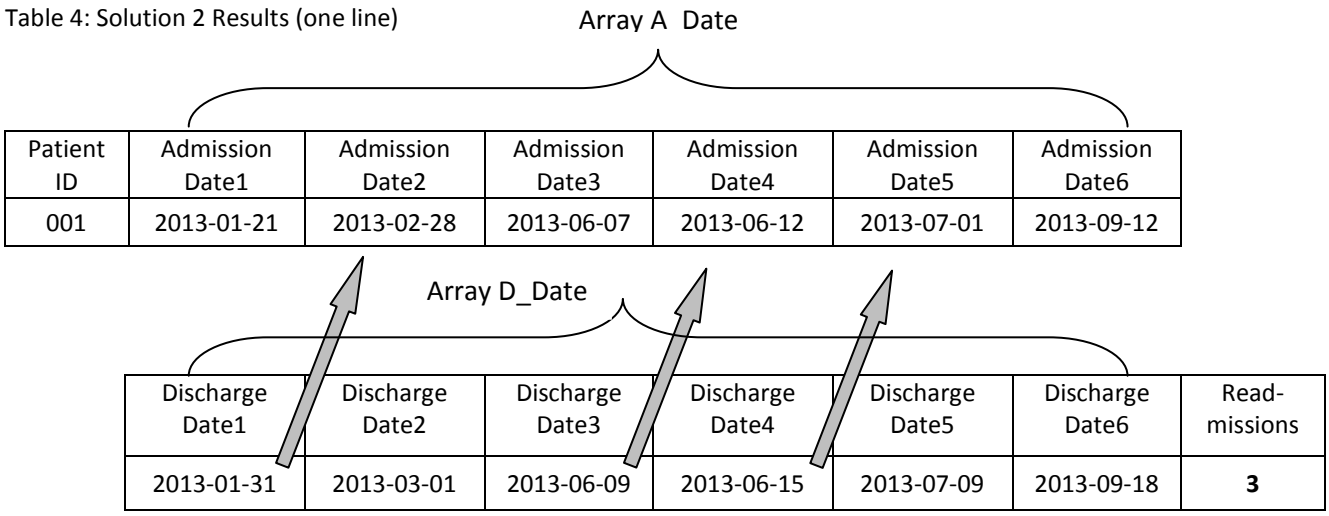
Solution 2 Codes:

```
* Transpose admission dates from long to wide;
Proc Transpose Data=Claims Out=Claims_A (keep=Patient_ID Admission_Date:)
  Prefix = Admission_Date;
  By Patient_ID;
  Var Admission_Date;
Run;
* Transpose discharge dates from long to wide;
Proc transpose data=Claims Out=Claims_D (keep=Patient_ID Discharge_Date:)
  Prefix = Discharge_Date;
  By Patient_id;
  Var Discharge_Date;
Run;
* Merge the admission date file with the discharge date file;
Data Claims_AD;
  Merge Claims_A
        Claims_D;
  By Patient_id;
Run;
* Identify readmissions;
Data Solution2 (drop=i j Tag);
  Set Claims_AD;
  Array A_Date {*} Admission_Date: ; * Create an array for admission dates ;
  Array D_Date {*} Discharge_Date: ;  * Create an array for discharge dates ;
  Readmissions = 0;
  Do i = 2 to dim(A_Date); * Counter i is for admission dates. For each patient, data evaluation always
                             begins with the second admission date;
     j = i - 1;            * Counter j is for discharge dates;
     Tag = 0;
```

```
    If 0 <= A_Date(i) - D_Date(j) <= 30 then Tag=1; * Compare the current admission date with the
                                                      previous discharge date;
    Readmissions  = Readmissions  + Tag;
    If D_Date{j} = . then leave;  * Leave the DO loop if there is no more discharge date to be evaluated;
  End;
Run;
```

Note that we have used two wildcard characters here: the asterisk (*) referring to the total number of array elements in each array and the colon (:) referring to the whole set of date fields in each array. A colon following a variable name prefix selects any variable with a name beginning with that prefix. In this case these two wildcard characters not only help us define an array without knowing the exact number of array elements beforehand but also save us the typing of all admission date and discharge date fields. The subsequent results are shown as follows (Table 4):

Table 4: Solution 2 Results (one line)



Patient ID	Admission Date1	Admission Date2	Admission Date3	Admission Date4	Admission Date5	Admission Date6
001	2013-01-21	2013-02-28	2013-06-07	2013-06-12	2013-07-01	2013-09-12

Discharge Date1	Discharge Date2	Discharge Date3	Discharge Date4	Discharge Date5	Discharge Date6	Read-missions
2013-01-31	2013-03-01	2013-06-09	2013-06-15	2013-07-09	2013-09-18	**3**

While not requiring the mastery of the LAG function, the second solution does take a few extra but brief steps to achieve the same result. This solution transposes the claims data from long to wide and examines each patient's hospitalization experiences in just one line. This horizontal thinking process can be very useful in circumstances that require simultaneous evaluation of multiple observations per subject, and its application to calculating other health outcomes such as medication utilization has been discussed in previous SAS papers.

SOLUTION 3: PROC SQL SELF-JOIN

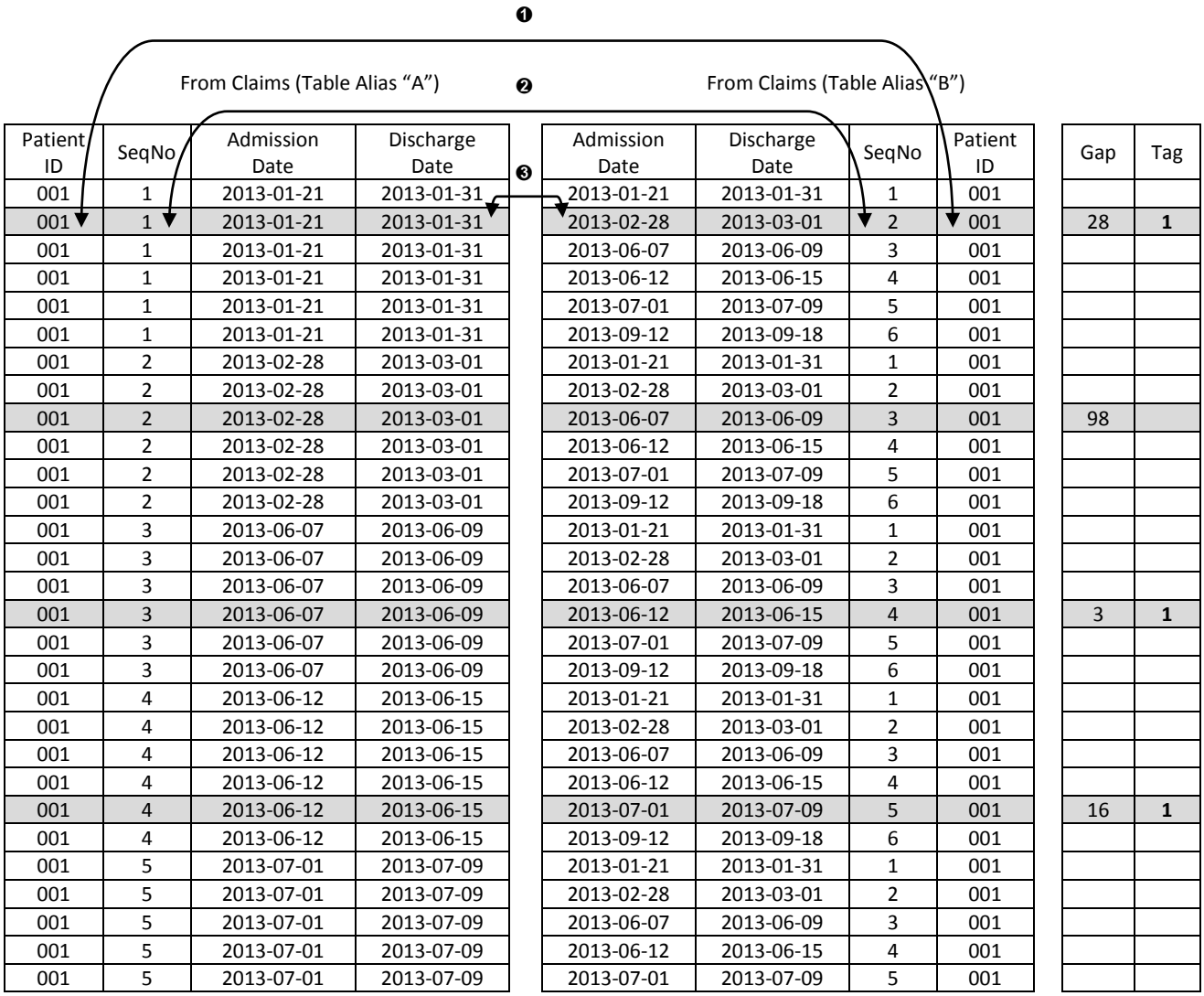
The third solution involves table joining in PROC SQL, which is often a powerful addition to DATA step programming. Unlike most table joins in PROC SQL that involve two or more different tables, the self-join in this solution joins a single table ("Claims") with itself to achieve the desired result. Due to the duplicate keys (patient ID), the self-join here is essentially a many-to-many merge realized in PROC SQL. It creates a Cartesian product which is further subsetted by the joining condition based on patient ID and a WHERE clause. The WHERE clause evaluates sequence numbers in such a way that an admission date would only be compared to the immediate previous discharge date for the same patient. The WHERE clause then evaluates the gap between these two dates and identifies any eligible cases of 30-day readmissions. Similarly, a readmission indicator field ("Tag") can be created for the purpose of tallying the total number of readmissions.

Solution 3 Codes:

```
PROC SQL;
Create Table Solution3 as
   Select A.Patient_ID, A.Discharge_Date label = "Previous Discharge Date",
          B.Admission_Date label = "Admission Date",
          B.Admission_Date - A.Discharge_Date as Gap, 1 as Tag
           From Claims as A join Claims as B
           On A.Patient  ID = B.Patient  ID ❶
           Where B.SeqNo = A.SeqNo + 1 ❷ and
                 B.Admission_Date – A.Discharge_Date between 0 and 30 ❸;
Quit;
```

In the PROC SQL statements above, the same source table ("Claims") is listed twice in the FROM clause and two different table aliases must be assigned here to differentiate them. The output table can select the key (patient ID) from either table, but the gap calculation must be based on an admission date from one table and the immediate previous discharge date from the other. The table self-joining process looks as follows (see Table 5):

Table 5: PROC SQL Self-Join Process



Patient ID	SeqNo	Admission Date	Discharge Date	Admission Date	Discharge Date	SeqNo	Patient ID	Gap	Tag
001	1	2013-01-21	2013-01-31	2013-01-21	2013-01-31	1	001		
001	1	2013-01-21	2013-01-31	2013-02-28	2013-03-01	2	001	28	**1**
001	1	2013-01-21	2013-01-31	2013-06-07	2013-06-09	3	001		
001	1	2013-01-21	2013-01-31	2013-06-12	2013-06-15	4	001		
001	1	2013-01-21	2013-01-31	2013-07-01	2013-07-09	5	001		
001	1	2013-01-21	2013-01-31	2013-09-12	2013-09-18	6	001		
001	2	2013-02-28	2013-03-01	2013-01-21	2013-01-31	1	001		
001	2	2013-02-28	2013-03-01	2013-02-28	2013-03-01	2	001		
001	2	2013-02-28	2013-03-01	2013-06-07	2013-06-09	3	001	98	
001	2	2013-02-28	2013-03-01	2013-06-12	2013-06-15	4	001		
001	2	2013-02-28	2013-03-01	2013-07-01	2013-07-09	5	001		
001	2	2013-02-28	2013-03-01	2013-09-12	2013-09-18	6	001		
001	3	2013-06-07	2013-06-09	2013-01-21	2013-01-31	1	001		
001	3	2013-06-07	2013-06-09	2013-02-28	2013-03-01	2	001		
001	3	2013-06-07	2013-06-09	2013-06-07	2013-06-09	3	001		
001	3	2013-06-07	2013-06-09	2013-06-12	2013-06-15	4	001	3	**1**
001	3	2013-06-07	2013-06-09	2013-07-01	2013-07-09	5	001		
001	3	2013-06-07	2013-06-09	2013-09-12	2013-09-18	6	001		
001	4	2013-06-12	2013-06-15	2013-01-21	2013-01-31	1	001		
001	4	2013-06-12	2013-06-15	2013-02-28	2013-03-01	2	001		
001	4	2013-06-12	2013-06-15	2013-06-07	2013-06-09	3	001		
001	4	2013-06-12	2013-06-15	2013-06-12	2013-06-15	4	001		
001	4	2013-06-12	2013-06-15	2013-07-01	2013-07-09	5	001	16	**1**
001	4	2013-06-12	2013-06-15	2013-09-12	2013-09-18	6	001		
001	5	2013-07-01	2013-07-09	2013-01-21	2013-01-31	1	001		
001	5	2013-07-01	2013-07-09	2013-02-28	2013-03-01	2	001		
001	5	2013-07-01	2013-07-09	2013-06-07	2013-06-09	3	001		
001	5	2013-07-01	2013-07-09	2013-06-12	2013-06-15	4	001		
001	5	2013-07-01	2013-07-09	2013-07-01	2013-07-09	5	001		

001	5	2013-07-01	2013-07-09
001	6	2013-09-12	2013-09-18
001	6	2013-09-12	2013-09-18
001	6	2013-09-12	2013-09-18
001	6	2013-09-12	2013-09-18
001	6	2013-09-12	2013-09-18
001	6	2013-09-12	2013-09-18

2013-09-12	2013-09-18	6	001
2013-01-21	2013-01-31	1	001
2013-02-28	2013-03-01	2	001
2013-06-07	2013-06-09	3	001
2013-06-12	2013-06-15	4	001
2013-07-01	2013-07-09	5	001
2013-09-12	2013-09-18	6	001

65	

After the data filtering by the WHERE clause, the printout result table should look like this (Table 6):

Table 6: Solution 3 Results

Patient ID	Previous Discharge Date	Admission Date	Gap	Tag
001	2013-01-31	2013-02-28	28	1
001	2013-06-09	2013-06-12	3	1
001	2013-06-15	2013-07-01	16	1

Solution 3 correctly identifies the exactly same 30-day readmissions as the previous two solutions do. Compared to the DATA step solutions, PROC SQL self-join requires no special function or extra data management, and it is particularly useful when there is a need to compare and evaluate variables in different observations within the same dataset. Note that the claims data processed here has already been sorted by patient ID and hospitalization dates at the data preparation stage. Sorting is not required by PROC SQL table joining, but our initial sorting is very helpful in this case because it not only makes DATA step solutions much easier but also creates a line counter field ("SeqNo") which can be used in the PROC SQL self-join handily.

CONCLUSION

Both SAS DATA step and PROC SQL can provide efficient solutions to accurately identify hospital readmissions. SAS users can choose any one of the methods based on their data and/or programming preference, or they can use one method to quality check the results achieved from another. Programming logics discussed in this paper can also be expanded and applied to tackle similar challenges in the field of health services research.

DISCLAIMER: The contents of this paper are the sole work of the authors and do not necessarily represent the views or practices of the UMWA Health and Retirement Funds, its Trustees or employees.

REFERENCES

America's Health Insurance Plans, Center for Policy and Research. Working Paper: Simple Methods of Measuring Hospital Readmission Rates, February 2012.

Jencks S, Williams MV and Coleman EA (2009). Rehospitalizations among Patients in the Medicare Fee-for-Service Program. N ENGL J Med (360): 1418-1428.

National Center for Health Statistics, Health Indicator Warehouse. Acute Hospital Readmissions (Percent), <http://healthindicators.gov/Indicators/Acute-hospital-readmissions-percent_279/Profile>

U.S. Agency for Healthcare Research and Quality. HCUP Methods Series: Methodological Issues when Studying Readmissions and Revisits using Hospital Administrative Data (Report # 2011-01).

U.S. Agency for Healthcare Research and Quality. HCUP Methods Series: Overview of Key Readmission Measures and Methods (Report # 2012-04).

Yale New Haven Health Services Corporation/Center for Outcomes Research & Evaluation (YNHHSC/CORE). 2012 Measures Maintenance Technical Report: Acute Myocardial infarction, Heart Failure, and Pneumonia 30-Day Risk-Standardized Readmission Measure, April 16, 2012.

Fan Z, Subsetting SAS Data Set by Using PROC SQL Self-join with Compound Key, Proceedings of SAS User Group International 29, 2004, <http://www2.sas.com/proceedings/sugi29/065-29.pdf>

Lee S, Multiple Column Transposition, PharmaSUG 2001,
<http://www.lexjansen.com/pharmasug/2001/proceed/techtech/tt12_lee.pdf>

Leslie RS, Using Arrays to Calculate Medication Utilization, Proceedings of SAS Global Forum 2007,
<http://www2.sas.com/proceedings/forum2007/043-2007.pdf>

Luo H, That Mysterious Colon (:), Proceedings of SAS User Group International 26, 2001,
< http://www2.sas.com/proceedings/sugi26/p073-26.pdf>

Tian Y, LAG – the Very Powerful and Easily Misused SAS Function, Proceedings of SAS Global Forum 2009,
<http://support.sas.com/resources/papers/proceedings09/055-2009.pdf>

CONTACTINFORMATION

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

Weifeng Fan
UMWA Health and Retirement Funds
2121 K street NW, Suite 350
Washington, DC 20037
wfan@umwafunds.org

Maryam Sarfarazi
UMWA Health and Retirement Funds
2121 K street NW, Suite 350
Washington, DC 20037
msarfara@umwafunds.org