Paper BB66

Combining Multiple Date-Ranged Historical Data Sets with Dissimilar Date Ranges into a Single Change History Data Set

Jim Moon, County of Fairfax VA

ABSTRACT

This paper describes a method that uses some simple SAS® macros and SQL to merge data sets having related data and containing rows with varying effective date ranges. The data sets are merged into a single data set that represents a serial list of snapshots of the merged data, as of a change in any of the effective dates. While simple conceptually, this type of merge is often problematic when the effective date ranges are not consecutive or consistent, when the ranges overlap, or when there are missing ranges from one or more of the merged data sets. The technique described is used by the Fairfax County Human Resources Department to combine various employee data sets (Employee Name and Personal Data, Personnel Assignment and Job Classification, Personnel Actions, Position-Related data, Pay Plan and Grade, Work Schedule, Organizational Assignment, and so on) from the County's SAP-HCM ERP system into a single Employee Action History/Change Activity file for historical reporting purposes. The technique currently is used to combine nineteen data sets, but is easily expandable by inserting a few lines of code using the existing macros.

INTRODUCTION

Programmers are often called upon to produce reports showing data selected from different tables or SAS data sets, related by a common key (referred to in this paper as “Id”), but having different effective date ranges. In essence, we must "match up" the data from all the related data sets, and be able to provide a "snapshot" of the data values from each data set, at any given point in time.

STATEMENT OF THE PROBLEM

THE TROUBLE WITH TRADITIONAL TECHNIQUES

It's complicated! In practice, matching date ranges across tables using a simple JOIN results in a cartesian product with every row in the range matched with every row in every other range that overlaps. This effect is amplified when multiple data sets are involved, as each subsequent join involving another data set with a different date range produces another cartesian product. Ideally, we would like to match rows across all data sets to a specific date, rather than match a range to another range.

Consider the complexity of the following SQL statement that selects overlapping id/date ranges across just three tables:

```
SELECT *
FROM TABLEA a, TABLEB b, TABLEC c
WHERE a.Id = b.Id   AND
      b.Id = c.Id AND
( ( b.BegDat <= a.BegDat  and b.EndDat >= a.BegDat ) OR
     b.BegDat between a.BegDat and a.EndDat ) AND
( ( c.BegDat <= b.BegDat  and c.EndDat >= b.BegDat ) OR
     c.BegDat between b.BegDat and b.EndDat ) AND
( ( c.BegDat <= a.BegDat  and c.EndDat >= a.BegDat ) OR
     c.BegDat between a.BegDat and a.EndDat );
```

In practice, when many date-ranged tables are involved, we may need to match date ranges across dozens of tables. With every additional date-ranged data set, the coding becomes increasingly complex, cumbersome, and difficult to understand and maintain.

There can be no gaps! Another issue that arises when joining multiple date-ranged data sets is that a missing date range in any one of the joined data sets removes all rows for the missing date range from the result. We would like to preserve those “missing range” rows in the output data set and produce “missing values” in variables for those observations.

So, the question remains: How can we match multiple data sets with varying effective date ranges and correctly “marry” the rows that are effective within the aggregate date ranges across all the data sets? And, gracefully handle “missing” ranges as well? And, be expandable as new data sets are added?

THE SOLUTION - OVERALL APPROACH

Step #1: Let's approach this problem from a simpler perspective. First, create a list of specific dates where the values in any of the data sets could have changed for each "id". The only possible dates the data in the any of the data sets could have changed are the endpoints of the data ranges, i.e. the start and end dates. Therefore, if we extract all the range start and end dates for each "id" into a single list, we have distilled a finite list of possible change dates for each "id" across all the data sets. Typically, end dates are "start date minus 1 of the next range", so for this paper we will only consider start dates in our list of possible change dates.

Step #2: Once we have a finite list of change dates for each "id", the rest is simple! We select the rows from each data set where the Change Date is BETWEEN the range Start Date and End Date. If a particular row spans multiple Change Dates, then multiple rows will be produced, one row for each Change Date encompassed by its Date Range. (A SAS macro is provided to simplify the coding for selecting the "Change Date" rows from each of the multiple tables.)

Step #3: We then MERGE all the data sets by Id/Change Date, producing variables with "missing" values in observations with missing date ranges. We, thereby, achieve our desired result of a comprehensive Change History data set, keyed by Id and Change Date, having all variables from all the input data sets, "as they were" on the Change Date.

Figure 1 depicts the overall approach showing three date-ranged data sets. Moving from left to right, Step #1 uses SELECT DISTINCT to produce a list of unique Id/Change Dates across the three data sets. Step #2 uses SELECT... BETWEEN to expand the original data sets into multiple rows encompassed by each row's date range. Step #3 merges all the expanded data sets and produces "missing values" for variable in data sets having no matching date range.

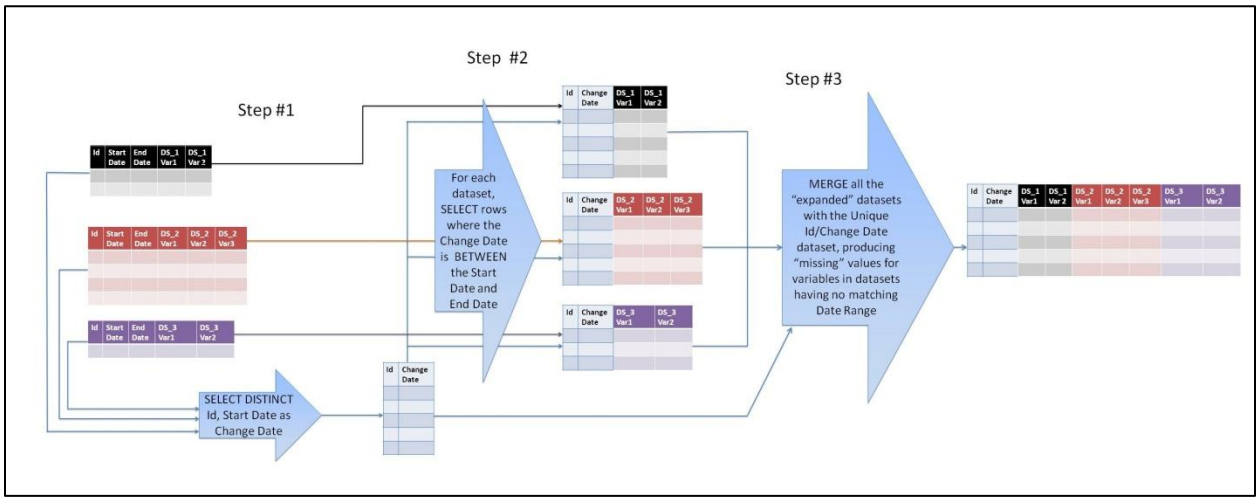

Figure 1 The Solution - Overall Approach

STEP-BY-STEP DESCRIPTION AND SAMPLE CODE

SAMPLE DATA

For our simplified example, we'll use some fictitious Employee Data, shown in Figure 2. There are four date-ranged data sets, EmpName, Job, WorkLoc and Emp401K, each having common variables Id, StartDate and EndDate, and each having one other variable unique to that data set. Three employees are represented in our example: Joan, Cathy and Mary. In practice, each data set will contain many variables specific to that data set, and there will be thousands (perhaps millions) of rows in each date-ranged table.

EmpName

Obs	Id	StartDate	EndDate	FName
1	120	11/01/2010	12/31/9999	Joan
2	440	09/29/2009	12/31/9999	Cathy
3	555	05/05/2009	12/31/9999	Mary

```
------------------------------------------------------------
Job
Obs     Id     StartDate       EndDate    Jobtitle
 1     120    11/01/2010    06/15/2011    Programmer II
 2     120    06/16/2011    01/28/2013    Programmer III
 3     120    01/29/2013    12/31/9999    System Architect
 4     440    09/29/2009    12/31/9999    Bus Analyst II
 5     555    05/20/2009    06/20/2011    Programmer II
 6     555    06/21/2011    12/31/9999    Analyst III
------------------------------------------------------------
WorkLoc
Obs     Id     StartDate       EndDate    JobLocation
 1     120    11/01/2010    07/20/2012    Bldg 22
 2     120    07/21/2012    12/30/9999    Main Office
 3     440    10/02/2009    12/30/9999    Main Office
 4     555    05/05/2009    05/10/2009    Main Office
 5     555    05/11/2009    06/20/2009    Bldg 4
 6     555    06/21/2009    06/21/2009    Bldg 12
 7     555    08/22/2009    12/05/2009    Main Office
 8     555    12/06/2009    06/20/2011    Bldg 5
 9     555    06/21/2011    12/31/9999    Computer Lab
------------------------------------------------------------
Emp401K
Obs     Id     StartDate       EndDate    DedPct
 1     120    10/05/2012    12/30/9999    0.05
 2     555    08/22/2009    12/31/2011    0.02
 3     555    01/01/2012    12/31/2012    0.10
 4     555    01/01/2013    12/31/9999    0.00
```

Figure 2 - Contents of Sample Date-Ranged Data sets

STEP #1 - SELECT THE UNIQUE CHANGE DATES

Create a data set that contains all the unique ids and begin dates. This is the “universe” of all the Id/Change Dates on which a change could possibly have occurred.

For each date-ranged table, using SELECT DISTINCT, we create a table of unique ID and Start Dates. Combine all the ID/Start Date tables using UNION ALL and eliminate duplicates. We are left with a table of all the unique ID/Change Dates across all the tables (see Figure 3).

```
/*****************************************************/
/* Step #1 - Select the Unique Change Dates.         */
/* Extract the Ids and Start Dates from each         */
/* historical dataset and eliminate duplicates.      */
/* A dataset of Change Dates is produced that        */
/* contains all the unique Ids and Change Dates.     */
/* This is the universe of all Id/Change Dates on    */
/* on which a change could have occurred.            */
/*****************************************************/

proc sql;
create table ChangeDates as
select distinct
    Id
  , StartDate as ChgStartDate  format=MMDDYY10.
from
(     select Id,StartDate        from EmpName
      union all
      select Id,StartDate        from Job
      union all
      /*Insert additional date-ranged datasets, as needed */
      select Id,StartDate        from WorkLoc
      union all
      select Id,StartDate        from Emp401K)
order by Id, ChgStartDate;
quit;
```

ChangeDates

Obs	Id	ChgStart Date
1	120	11/01/2010
2	120	06/16/2011
3	120	07/21/2012
4	120	10/05/2012
5	120	01/29/2013
6	440	09/29/2009
7	440	10/02/2009
8	555	05/05/2009
9	555	05/11/2009
10	555	05/20/2009
11	555	06/21/2009
12	555	08/22/2009
13	555	12/06/2009
14	555	06/21/2011
15	555	01/01/2012
16	555	01/01/2013

Figure 3 - Step #1: Select all unique Id/ChangeDates across all date-ranged data sets

STEP #2 - SELECT ROWS "BETWEEN" DATE RANGE

For each date-ranged table, create a new "expanded" table with rows that match the “universe” of Change Dates for that ID (from Step #1). In other words, select rows from each table where the ID matches ***and the Change Date is BETWEEN the Begin and End Dates***. Each date-ranged row produces one or more rows, a row for each Change Date encompassed by its Date Range (see Figure 4). For ease of maintenance and coding, a macro is used to apply the same process to each of the data sets.

```
/*****************************************************/
/* Step #2 - Select rows "between" the date ranges.  */
/* For each date-ranged dataset, select rows from    */
/* each dataset where the Id matches and the         */
/* Change Date is BETWEEN the Begin and End Dates.   */
/*****************************************************/
%macro select_between(dsname, sortkey, uniqkey);
/*****************************************************/
/* This macro will be executed for each date-ranged  */
/* dataset that will be included in the Change       */
/* History table.  For the provided dataset, the     */
/* macro extracts the rows having a BeginDate/EndDate*/
/* that matches the Change Date for a given Id.      */
/* Duplicates, if any, are removed.                  */
/*****************************************************/
proc sql;
create table between_&dsname as
select
    a.Id
  , a.ChgStartDate
  , i.*
from ChangeDates a , &dsname i
where a.Id = i.Id and
    a.ChgStartDate between i.StartDate and i.EndDate;
quit;

proc sort data=between_&dsname
out=sort1_&dsname;
by &sortkey;
proc sort data=sort1_&dsname nodupkey
out=uniq_between_&dsname;
by &uniqkey;
%mend select_between;

%select_between(EmpName, Id ChgStartDate descending StartDate descending
EndDate, Id ChgStartDate);
```

```
%select_between(Job, Id ChgStartDate descending StartDate descending EndDate,
Id ChgStartDate);

/*Insert additional macro executions for each additional dataset, as needed */

%select_between(WorkLoc, Id ChgStartDate descending StartDate descending
EndDate, Id ChgStartDate);

%select_between(Emp401K, Id ChgStartDate descending StartDate descending
EndDate, Id ChgStartDate);
```

The SAS System

uniq_between_Job

Obs	Id	ChgStart Date	StartDate	EndDate	Jobtitle
1	120	11/01/2010	11/01/2010	06/15/2011	Programmer II
2	120	06/16/2011	06/16/2011	01/28/2013	Programmer III
3	120	07/21/2012	06/16/2011	01/28/2013	Programmer III
4	120	10/05/2012	06/16/2011	01/28/2013	Programmer III
5	120	01/29/2013	01/29/2013	12/31/9999	System Architect
6	440	09/29/2009	09/29/2009	12/31/9999	Bus Analyst II
7	440	10/02/2009	09/29/2009	12/31/9999	Bus Analyst II
8	555	05/20/2009	05/20/2009	06/20/2011	Programmer II
9	555	06/21/2009	05/20/2009	06/20/2011	Programmer II
10	555	08/22/2009	05/20/2009	06/20/2011	Programmer II
11	555	12/06/2009	05/20/2009	06/20/2011	Programmer II
12	555	06/21/2011	06/21/2011	12/31/9999	Analyst III
13	555	01/01/2012	06/21/2011	12/31/9999	Analyst III
14	555	01/01/2013	06/21/2011	12/31/9999	Analyst III

The SAS System

uniq_between_WorkLoc

Obs	Id	ChgStart Date	StartDate	EndDate	JobLocation
1	120	11/01/2010	11/01/2010	07/20/2012	Bldg 22
2	120	06/16/2011	11/01/2010	07/20/2012	Bldg 22
3	120	07/21/2012	07/21/2012	12/30/9999	Main Office
4	120	10/05/2012	07/21/2012	12/30/9999	Main Office
5	120	01/29/2013	07/21/2012	12/30/9999	Main Office
6	440	10/02/2009	10/02/2009	12/30/9999	Main Office
7	555	05/05/2009	05/05/2009	05/10/2009	Main Office
8	555	05/11/2009	05/11/2009	06/20/2009	Bldg 4
9	555	05/20/2009	05/11/2009	06/20/2009	Bldg 4
10	555	06/21/2009	06/21/2009	06/21/2009	Bldg 12
11	555	08/22/2009	08/22/2009	12/05/2009	Main Office
12	555	12/06/2009	12/06/2009	06/20/2011	Bldg 5
13	555	06/21/2011	06/21/2011	12/31/9999	Computer Lab
14	555	01/01/2012	06/21/2011	12/31/9999	Computer Lab
15	555	01/01/2013	06/21/2011	12/31/9999	Computer Lab

The SAS System

uniq_between_Emp401K

Obs	Id	ChgStart Date	StartDate	EndDate	Ded Pct
1	120	10/05/2012	10/05/2012	12/30/9999	0.05
2	120	01/29/2013	10/05/2012	12/30/9999	0.05
3	555	08/22/2009	08/22/2009	12/31/2011	0.02
4	555	12/06/2009	08/22/2009	12/31/2011	0.02
5	555	06/21/2011	08/22/2009	12/31/2011	0.02
6	555	01/01/2012	01/01/2012	12/31/2012	0.10
7	555	01/01/2013	01/01/2013	12/31/9999	0.00

Figure 4 - Step #2: Resulting "expanded" tables - Each Data Set has a row for each Change Date encompassed by its Date Range

We'll also create a data set with the Change Date Ranges based on the unique Change Dates from Step #1. This DATA step calculates the Change End Date, which is one day less than the Change Start Date of the next row. Figure 5 shows the resulting data set, having both Change Start Date and Change End Date.

```
/*************************************************/
/* Create the Change Date Range dataset based on the */
/* unique Change Dates.  This provides the range     */
/* encompassed between each Change Date.             */
/*************************************************/
data Change_Date_Range (rename = (ChgEffDate = ChgStartDate));
set ChangeDates;
by Id;
retain ChgEffDate ChgEndDate 8;
if first.Id then ChgEffDate = ChgStartDate;
else do;
       ChgEndDate = ChgStartDate - 1;
       output;
       ChgEffDate = ChgStartDate;
end;
if last.Id then do;
       ChgEndDate = '31dec9999'd;
       output;
end;
attrib ChgEffDate  format=mmddyy10.
       ChgEndDate  format=mmddyy10.;
drop ChgStartDate;
run;
```

The SAS System
Change_Date_Range

Obs	Id	ChgStart Date	ChgEndDate
1	120	11/01/2010	06/15/2011
2	120	06/16/2011	07/20/2012
3	120	07/21/2012	10/04/2012
4	120	10/05/2012	01/28/2013
5	120	01/29/2013	12/31/9999
6	440	09/29/2009	10/01/2009
7	440	10/02/2009	12/31/9999
8	555	05/05/2009	05/10/2009
9	555	05/11/2009	05/19/2009
10	555	05/20/2009	06/20/2009
11	555	06/21/2009	08/21/2009
12	555	08/22/2009	12/05/2009
13	555	12/06/2009	06/20/2011
14	555	06/21/2011	12/31/2011
15	555	01/01/2012	12/31/2012
16	555	01/01/2013	12/31/9999

Figure 5 - Change Date Ranges

Step #3 - GREAT BIG MERGE!

Now, we MERGE all the tables from Step #2 with the table of unique Id/Change Date Ranges in a multi-file MERGE by Id/Change Date. SAS does the hard work of matching all the data sets in one simple step, and ensures that observations with “missing” values are generated. The result is a “Big Flat Table" we'll call the Change History Master data set, having all the variables from all the date-ranged tables in a single data set (see Figure 6). There is a single row for each ID/Change Date, with values "as of" the Change Date, and "missing values" are produced, as appropriate, for variables in data sets having no row covering the Change Date.

```
/*************************************************/
/* Step #3 - Great Big Merge!                    */
/* Merge all the datasets from Step #2 with the  */
/* dataset of unique Id/Change Dates, producing the  */
/* Change History Master dataset.                */
/*************************************************/
data Change_History_Master;
merge
    Change_Date_Range
```

```
        uniq_between_EmpName
        uniq_between_Job
        /*Insert additional datasets as needed */
        uniq_between_WorkLoc
        uniq_between_Emp401K
        ;
    by Id ChgStartDate;
    drop StartDate EndDate;
    run;
```

Change_History_Master

Obs	Id	ChgStartDate	ChgEndDate	FName	Jobtitle	JobLocation	DedPct
1	120	11/01/2010	06/15/2011	Joan	Programmer II	Bldg 22	.
2	120	06/16/2011	07/20/2012	Joan	Programmer III	Bldg 22	.
3	120	07/21/2012	10/04/2012	Joan	Programmer III	Main Office	.
4	120	10/05/2012	01/28/2013	Joan	Programmer III	Main Office	0.05
5	120	01/29/2013	12/31/9999	Joan	System Architect	Main Office	0.05
6	440	09/29/2009	10/01/2009	Cathy	Bus Analyst II		.
7	440	10/02/2009	12/31/9999	Cathy	Bus Analyst II	Main Office	.
8	555	05/05/2009	05/10/2009	Mary		Main Office	.
9	555	05/11/2009	05/19/2009	Mary		Bldg 4	.
10	555	05/20/2009	06/20/2009	Mary	Programmer II	Bldg 4	.
11	555	06/21/2009	08/21/2009	Mary	Programmer II	Bldg 12	.
12	555	08/22/2009	12/05/2009	Mary	Programmer II	Main Office	0.02
13	555	12/06/2009	06/20/2011	Mary	Programmer II	Bldg 5	0.02
14	555	06/21/2011	12/31/2011	Mary	Analyst III	Computer Lab	0.02
15	555	01/01/2012	12/31/2012	Mary	Analyst III	Computer Lab	0.10
16	555	01/01/2013	12/31/9999	Mary	Analyst III	Computer Lab	0.00

Figure 6 - The FInal Result! Change History Master Data Set - one row for each Id/Change Date having values "as of" the Change Date, "missing values" inserted for missing Date Ranges

Once we have the Change History Master data set, we can now easily select a snapshot of the data as it existed on any date across all the date-ranged data sets. We don't even have to know ahead of time what date we will be looking for.

The basic WHERE clause for selecting a snapshot from the Change History Master for any desired date is:

```
WHERE <snapshot date> between ChgStartDate and ChgEndDate
```

SAMPLE QUERIES

Query 1: Select "snapshot" of all variables for a specific date

```
/*****************************************************/
/* Select the unique row for each employee with a    */
/* snapshot of data on January 1, 2012.              */
/*****************************************************/
proc sql;
select *
from Change_History_Master
where '01JAN2012'd  between ChgStartDate and ChgEndDate;
quit;
```

The SAS System

Change_History_Master

Id	ChgStartDate	ChgEndDate	FName	Jobtitle	JobLocation	DedPct
120	06/16/2011	07/20/2012	Joan	Programmer III	Bldg 22	.
440	10/02/2009	12/31/9999	Cathy	Bus Analyst II	Main Office	.
555	01/01/2012	12/31/2012	Mary	Analyst III	Computer Lab	0.1

Figure 7 - Results from Query 1 - "Snapshot" of All Variables, As Of Jan 1, 2012

Query 2: Select "snapshot" of all variables for a specific date, with additional criteria

```
/*************************************************************/
/* Select all employees who were Programmers on June 1, 2011.  */
/*************************************************************/
proc sql;
select *
from Change_History_Master
where '01JUN2011'd  between ChgStartDate and ChgEndDate
  and JobTitle contains "Programmer";
quit;
```

```
                                The SAS System
                              Change_History_Master

     Id  ChgStartDate  ChgEndDate  FName       Jobtitle             JobLocation       DedPct
    ---------------------------------------------------------------------------------------
    120    11/01/2010  06/15/2011  Joan        Programmer II        Bldg 22                .
    555    12/06/2009  06/20/2011  Mary        Programmer II        Bldg 5              0.02
```

Figure 8 - Results from Query 2 - "Programmers" As Of June 1, 2011

Query 3: Select "snapshot" of all variables for a specific date, with additional criteria

```
/*****************************************************/
/* Select employees contributing to 401K as of       */
/* Jan 15, 2013.                                     */
/*****************************************************/
proc sql;
select *
from Change_History_Master
where '15JAN2013'd  between ChgStartDate and ChgEndDate
  and DedPct > 0;
quit;
```

```
                                The SAS System
                              Change_History_Master

     Id  ChgStartDate  ChgEndDate  FName       Jobtitle             JobLocation       DedPct
    ---------------------------------------------------------------------------------------
     120    10/05/2012  01/28/2013  Joan         Programmer III       Main Office         0.05
```

Figure 9 - Results from Query 3 - 401K Deduction Percent > 0 on Jan 15, 2013

CONCLUSION

Combining multiple date-ranged data sets using traditional methods of complicated multi-table JOIN, or a series of OUTER JOINs, will not provide reliable results when the date ranges overlap or are missing. While requiring a three-step process, the simple technique described in this paper will unequivocally provide the desired results of creating a comprehensive Change History data set, containing a “snapshot” of the values of every observation “as of” the Change Date.

The crux of this technique is approaching the problem from a different perspective. Rather than attempting some form of iterative comparison of “date ranges to date ranges to date ranges”, we use a simple process that creates a finite aggregate list of specific “Change Dates”, and then we apply a traditional unambiguous selection using “WHERE...BETWEEN”, followed by an efficient MERGE process to combine the multiple merge data sets into a single Change History data set.

REFERENCES

Moon, James (2014), "Combining Multiple Date-Ranged Historical Data Sets with Dissimilar Date Ranges into a Single Change History Data Set", Proceedings of SAS Global Forum (March 13, 2014)

ACKNOWLEDGEMENTS

Special thanks to "SAS Nerd Extraodinaire" Kirk Paul Lafler for valuable feedback and suggestions, and to my SAS colleague at Fairfax County, Madhukar Ravi, for encouragement and support in preparing this paper.

CONTACT INFORMATION

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

James Moon
County of Fairfax, Virginia
Human Resources Information Systems Division (HRIS)
Dept of Human Resources
12000 Government Center Parkway, Suite 270
Fairfax, Virginia 22035
Email: james.moon@fairfaxcounty.gov
Direct 703-324-3497
Fax 703-324-3945

Disclaimer: The views expressed in this article are solely those of the author in his private capacity and do not necessarily reflect the position or policy of the Fairfax County, Virginia government.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.