Countdown of the Top 10 Ways to Merge Data

Paper TU06

David Franklin
Independent SAS Consultant
TheProgramersCabin.com

Introduction

Inspiration

- Most programmers only know two or three ways to merge data, but there others – some may be faster and quicker, but depends on a number of factors including number of observations and variables, sorted, or indexed, just to name a few

Survey

- A random (not very scientific) sample of around 100 people at PharmaSUG and NESUG were asked to name the ways they merged data
- Some of the sample respondents may have been influenced by wine and other alcohol products at the time which probably skewed the results

- **Not in the paper, but presented here will be a speed comparison of the methods**

The Data

Dataset: PATDATA

SUBJECT	TRT_CODE
124263	A
124264	A
124265	B
124266	B

Dataset: ADVERSE

SUBJECT	EVENT
124263	HEADACHE
124266	FEVER
124266	NAUSEA
124267	FRACTURE

Notes

- Subjects 124264 and 124265 are in PATDATA but not in ADVERSE, subject 124267 is in ADVERSE but not in PATDATA, and subject 124266 has two observations.
- Paper is looking at a 1-1 or 1-many merge, but some of these methods will handle many-many.

10. PEEK(C) AND POKE

```
01 DATA alldata0;
02  ARRAY f{&xpatdata.} $6 _TEMPORARY_; /*Store SUBJECT values*/
03  ARRAY g{&xpatdata.} $1 _TEMPORARY_; /*Store TRT_CODE values*/
04  LENGTH trt_code $1;
05  DO i=1 TO &xpatdata.;
06    SET patdata (RENAME=(trt_code=trt_dict));
07    CALL POKE(CATS(subject),
08    ADDR(f[1])+((i-1)*6),6);
09    CALL POKE(CATS(trt_dict),
10    ADDR(g[1])+((i-1)*1),1);
11  END;
12  DO i=1 TO &xadverse.;
13    SET adverse;
14    trt_code='';
15    DO j=1 TO &xpatdata.;
16      IF subject=PEEKC(ADDR(f[1])+((j-1)*6),6) THEN DO;
17        trt_code=PEEKC(ADDR(g[1])+((j-1)*1),1); OUTPUT;
18      END;
19      IF ^MISSING(trt_code) THEN LEAVE;
20    END;
21    IF MISSING(trt_code) THEN OUTPUT;
22  END;
23  DROP i j trt_code_dict;
24 RUN;
```

9. MERGE WITH UPDATE

- Has a little trouble with handling multiple observations in the master dataset - it is well documented that for the master dataset to be updated correctly there must be a unique key in the master dataset
- One way to get around this is to use the RETAIN statement where a temporary variable carries the TRT_CODE value across multiple records

```
01 DATA alldata0 (DROP=_trt_code);
02   LENGTH _trt_code $1; /*Temp variable containing TRT_CODE*/
03   RETAIN _trt_code '';
04   UPDATE adverse (in=a) patdata;
05   BY subject;
06   IF a;
07   IF FIRST.subject THEN _trt_code=trt_code;
08   ELSE trt_code=_trt_code;
09 RUN;
```

- A WARNING message indicating that "The MASTER data set contains more than one observation for a BY group" will appear but will be redundant since the value is carried from one observation to another by the variable _TRT_CODE

8. MERGE WITH ARRAY

```
01 DATA _null_; * Get the number of obs in PATDATA and ADVERSE;
02   SET sashelp.vtable;
03   WHERE libname='WORK' and memname in('PATDATA','ADVERSE');
04   CALL SYMPUT('X'||memname,put(nobs,8.));
05 DATA alldata0;
06   LENGTH trt_code $1;
07   * Create a 2 dim. array with SUBJECT and TRT_CODE values;
08   ARRAY f{&xpatdata.,2} $6 _TEMPORARY_;
09   DO i=1 TO &xpatdata.;
10     SET patdata (RENAME=(trt_code=trt_code_dict));
11     f{i,1}=PUT(subject,6.); f{i,2}=trt_code_dict;
12   END;
13   * Go though each record of ADVERSE & find a match;
14   DO i=1 TO &xadverse.;
15     SET adverse;
16     trt_code='';
17     DO j=1 TO &xpatdata.;
18       IF subject=INPUT(f(j,1),best.) THEN DO;
19         trt_code=f{j,2}; OUTPUT; END;
20       IF ^MISSING(trt_code) THEN LEAVE;
21     END;
22     IF MISSING(trt_code) THEN OUTPUT;
23   END;
24   DROP i j trt_code_dict;
25 RUN;
```

A Trivia Moment

Who was the first to fly non-stop across the Atlantic? Was it:

A: Charles A. Lindbergh

B: John Alcock and Arthur Brown

C: Walter Wellman

D: Amelia Earhart

E: Albert Read

A Trivia Moment – the answer

Who was the first to fly non-stop across the North Atlantic? Was it:

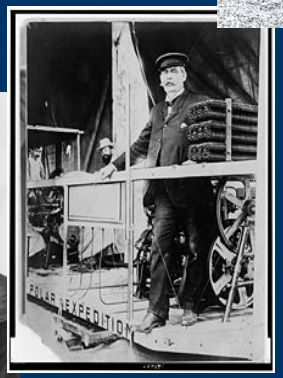
A: Charles A. Lindbergh -- May 20 - May 21, 1927

B: John Alcock and Arthur Brown
June 14 - June 15, 1919 (16h12m) – same distance today ~3h15m

C: Walter Wellman
October 1910 – by balloon, failed, but traveled over 1000 miles

D: Amelia Earhart -- May 20, 1932

E: Albert Read -- May 8 - May 31, 1919



7. MERGE WITH MODIFY

```
01 DATA adverse alldata0;
02   DO p=1 TO totobs;
03     _iorc_=0;
04     SET patdata point=p nobs=totobs;
05     DO WHILE(_iorc_=%sysrc(_sok));
06       MODIFY adverse KEY=subject;
07       SELECT (_iorc_);
08         WHEN (%sysrc(_sok)) DO; /*Match Found*/
09           SET patdata POINT=p; OUTPUT alldata0; END;
10         WHEN (%sysrc(_dsenom)) _error_=0; /*No Match*/
11         OTHERWISE DO; /*A major problem somewhere*/
12           PUT 'ERR' 'OR: _iorc_ = ' _iorc_ /
13               'program halted.'; _error_ = 0;
14           STOP; END;
15       END;
16     END;
17   END;
18   STOP;
19 RUN;
```

- creates the ALLDATA0 dataset but it is necessary to put that as the second dataset in the DATA statement
- dataset ADVERSE has an index called SUBJECT applied before the datastep is run

6. MERGE WITH HASH TABLE

- Trumpeted by SAS since version 9.1, and used by database programmers in other languages, this is considered one of the fastest ways to merge data in two datasets.
- Many papers have been written about this recent feature, how it works, and their use within SAS – refer online

```
01 DATA alldata0;
02   IF _n_=0 THEN SET patdata;
03   IF _n_=1 THEN DO;
04     DECLARE HASH _h1 (dataset: "PATDATA");
05     rc=_h1.definekey("SUBJECT");
06     rc=_h1.definedata("TRT_CODE");
07     rc=_h1.definedone();
08     call missing(SUBJECT,TRT_CODE);
09   END;
10   SET adverse;
11   rc=_h1.find();
12   IF rc^=0 THEN trt_code=" ";
13   DROP rc;;
14 RUN;
```

- PATDATA gets loaded into a hash table, then the ADVERSE dataset is loaded into the datastep and the match is made using the FIND() method.

A LITTLE BREAK IN THE COUNTDOWN ...

- Now that we are half way though our countdown, lets just take a moment to view a rather obscure method for merging our two datasets, the use of the CALL EXECUTE:

```
01 DATA _null_;
02   SET patdata;
03   CALL EXECUTE("DATA alldat;"||
04                " SET adverse;"||
05                " WHERE subject='"||STRIP(subject)||"';"||
06                " trt_code='"||STRIP(trt_code)||"';"||
07                "PROC APPEND BASE=alldata0 DATA=dat0 FORCE;"||
08                "RUN;");;
09 RUN;
```

- Uses CALL EXECUTE to add TRT_CODE from PATDATA to ADVERSE by SUBJECT, appending the result each time to the dataset ALLDATA0.

5. POINT OPTION IN THE SET STATEMENT

```
01 DATA alldata0;
02   SET adverse;
03   DROP _: match;
04   match=0;
05   DO i=1 TO xnobs;
06     SET patdata (rename=(subject=_subject)) NOBS=xnobs POINT=i;
07     IF subject=_subject THEN DO;
08       match=1; OUTPUT; END;
09   END;
10   IF match=0 THEN DO; * Output AE record if no match in CM;
11     CALL MISSING(trt_code); OUTPUT; END;
12 RUN;
```

- The key to this method is that the dataset takes each observation in AE and then tries to match this observations with an observation in PATDATA -- if there is a match, then the observation will be output, and if there is no matchthe AE record will still be output with no value for the variable TRT_CODE
- Because all observations in AE are read and matched will all observations in PATDATA, it is possible to do a many to many merge -- effectively, it is loop within a loop.
- What would happen at line 7 with IF SUBJECT=_SUBJECT AND EVENT=:”F”?

4. MERGE WITH FORMAT

```
01 DATA fmt;
02   RETAIN fmtname 'TRT_FMT' type 'C';
03   SET patdata;
04   RENAME subject=start trt_code=label;
05 PROC FORMAT CNTLIN=fmt;
06 DATA alldata0;
07   SET adverse;
08   ATTRIB trt_code LENGTH=$1 LABEL='Treatment Code';
09   trt_code=PUT(subject,$trt_fmt.);
10 RUN;
```

- In the example a character format TRT_FMT is created from the PATDATA dataset, and then this format is used to set the TRT_CODE variable within the ADVERSE dataset
- This method is useful as the data does not have to be sorted or indexed beforehand.

3. MERGE WITH SET-KEY

```
01 DATA alldata0;
02   SET adverse;
03   SET patdata KEY=subject /UNIQUE;
04   DO;
05     IF _IORC_ THEN DO;
06       _ERROR_=0;
07       trt_code='';
08     END;
09   END;
10 RUN;
```

- Before this type of merge can work the dataset PATDATA must have an index created inside it, using either the INDEX statement inside a DATASETS or SQL procedure, or INDEX option inside a DATA step.
- It is important to have the DO loop as if no match is found then TRT_CODE will be set to missing - if this is not done then unexpected results may occur.

2. MERGE WITH SQL

```
01 PROC SQL;
02   CREATE TABLE alldata0 AS
03     SELECT a.*, b.trt_code
04     FROM adverse a
05            LEFT JOIN
06          patdata b
07     ON a.subject=b.subject;
08   QUIT;
09 RUN;
```

- SQL is a well known language that is very good at working with databases and is liked by many who deal with large datasets.
- The SQL technique is the most common way a many-to-many merge of data is completed.

A Review of the Countdown ...

10. PEEK(C) AND POKE
9. MERGE WITH UPDATE
8. MERGE WITH ARRAY

Who was first to fly non-stop across the North Atlantic

7. MERGE WITH MODIFY
6. MERGE WITH HASH TABLE

A LITTLE BREAK IN THE COUNTDOWN ... (looked at CALL EXECUTE)

5. POINT OPTION IN THE SET STATEMENT
4. MERGE WITH FORMAT
3. MERGE WITH SET-KEY
2. MERGE WITH SQL

To Come ...

- Which method is number one
- Which method wins the speed contest

1. MERGE IN A DATA STEP

- After starting at number ten, we have now come to number one of the countdown, and the winner is of no real surprise. The winner by far, and is therefore the most common way to merge the two datasets is the MERGE statement used inside a datastep.

```
01 DATA alldata0;
02   MERGE adverse (in=a)
03         patdata (in=b);
04   BY subject;
05   IF a;
06 RUN;
```

- This method is the most common way of merging data as it gives most control, with the exception of the POINT= option discussed earlier, in the the way data is to be merged.

Which is the Fastest … only at conference

- This is only at conference, not in the paper!
- Question asked frequently, what is the fastest?
- No single answer as there are many factors – number of records, number of variables, indexed or not, sorted or not, CPU speed, memory available
- The following slide is a guide only – will depend on your site
- Try these out – code is available at my web site – go to TheProgrammersCabin.com, select the PharmaSUG 2010 option on the left hand side and you can see the SAS program

Which is the Fastest ... only at conference

	Number of Unique Subjects					
Method	1k	10k	50k	100k	500k	1M
1. Merge Statement	1.00	1.00	1.00	1.00	1.00	1.00
2. SQL	1.71 (1.71)	0.29 (0.55)	0.25 (0.26)	0.26 (0.27)	0.26 (0.27)	0.26 (0.25)
3. KEY= Option	2.43 (2.57)	2.05 (2.05)	1.95 (1.98)	2.34 (2.32)	2.56 (2.56)	2.51 (2.54)
4. FORMAT Procedure	0.43 (0.14)	0.12 (0.10)	0.13 (0.13)	0.16 (0.17)	0.25 (0.25)	0.28 (0.29)
5. POINT Option	61.71 (61.71)	592.90 (587.27)				
6. HASH Table	0.43 (0.14)	0.10 (0.12)	0.12 (0.12)	0.15 (0.14)	0.21 (0.21)	0.23 (0.24)
7. MODIFY Statement	2.86	1.42	1.30	1.30	1.31	1.26
8. ARRAY	13.14 (13.71)	119.73 (119.32)				
9. UPDATE	1.00	1.15	1.11	1.13	1.14	1.09

Notes:
1. PATDATA is merged with ADVERSE – ratio is 1:10, i.e. there is a average of 10 ADVERSE records to 1 PATDATA
2. Number given is a ratio against MERGE method
3. PATDATA is indexed – ADVERSE in indexed, except where time index is in parentheses
4. POKEC and PEEKC not done as OS dependent

Concorde

Conclusion

- So our countdown of the Top 10 Ways to Merge Data is now complete
- As can been seen there are a number of methods which can be used to merge data, beyond the MERGE statement within a DATA step
- No one method is better than another, and the methods shown here are by no means exhaustive
- It is only though trying these different methods at your installation that you will see resource efficiencies between the methods
- … and along the way we have learnt something about aviation history

Questions and Contact Information

Questions?
Your input is appreciated .....

Aurora

Contact Information
David Franklin
Independent SAS Consultant
Cell: 603-275-6809
E-mail: dfranklinuk@compuserve.com
Web: http://www.TheProgrammersCabin.com
LinkedIn: http://www.linkedin.com/in/davidfranklinnh