Merging Data Seven Different Ways

David Franklin, Independent Consultant, New Hampshire, USA

Abstract

Merging data is one of the fundamental functions carried out when manipulating data to bring it into a form for either storage or analysis. The use of the MERGE statement inside a datastep is the most common way this task is done within the SAS language but there are others. This paper looks at the seven possible methods, including the use of the MERGE statement, for a one to one, or one to many merge, introducing the SAS code needed to combine the data. No one method is better than another but some pointers will be given on choosing a method for your data.

Introduction

Merging variables from one dataset into another is one of the basic data manipulation tasks that a SAS

programmer has to do. The most common way to merge on data is using the MERGE statement in the DATA step but there are six other ways that can help. First though, some 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
```

This data will be used throughout the poster for each method described.

1 MERGE Statement

Most common code used to merge data – maximum control but data must be sorted or indexed before DATA statement.

```
DATA alldata0;
  MERGE adverse (in=a) patdata (in=b);
  BY subject;
  IF a;
RUN;
```

2 SQL

Another common way, but can be resource hungry if large datasets are involved.

```
PROC SQL;
  CREATE TABLE alldata0 AS
    SELECT a.*, b.trt_code
    FROM adverse a LEFT JOIN patdata b
    ON a.subject=b.subject;
  QUIT;
RUN;
```

3 KEY= option

A method that uses two SET statements, reading data from ADVERSE and then looping through PATDATA to find a match.

```
DATA alldata0;
  SET adverse;
  SET patdata KEY=subject /UNIQUE;
  DO;
    IF _IORC_ THEN DO;
      _ERROR_=0; trt_code='';
    END;
  END;
RUN;
```

4 FORMAT Procedure

Creates a format from PATDATA and then adds TRT_CODE to ADVERSE using the format.

```
DATA fmt;
  RETAIN fmtname 'TRT_FMT' type 'C';
  SET patdata;
  RENAME subject=start trt_code=label;
PROC FORMAT CNTLIN=fmt;
DATA alldata0;
  SET adverse;
  ATTRIB trt_code LENGTH=$1 LABEL='Treatment Code';
  trt_code=PUT(subject,$trt_fmt.);
RUN;
```

5 Hash Tables

Since SAS version 9.1, the use of hash tables to merge data has been available.

```
DATA alldata0;
  IF _n_=0 THEN SET patdata;
  IF _n_=1 THEN DO;
    DECLARE HASH _h1 (dataset: "PATDATA");
    rc=_h1.definekey("SUBJECT");
    rc=_h1.definedata("TRT_CODE");
    rc=_h1.definedone();
    call missing(SUBJECT,TRT_CODE);
  END;
  SET adverse;
  rc=_h1.find();
  IF rc^=0 THEN trt_code=" ";
  DROP rc;;
RUN;
```

6 Loading Unique Dataset into an Array

Unique dataset is put into an array first, then array is used to attach TRT_CODE to ADVERSE when reading dataset ADVERSE in.

```
DATA _null_;
  SET sashelp.vtable;
  WHERE libname='WORK';
  WHERE ALSO memname in('PATDATA','ADVERSE');
  CALL SYMPUT('X'||memname,put(nobs,8.));
DATA alldata0;
  LENGTH trt_code $1;
  ARRAY f{&xpatdata.,2} $6 _TEMPORARY_;
  DO i=1 TO &xpatdata.;
    SET patdata (RENAME=(trt_code=trt_code_dict));
    f{i,1}=PUT(subject,6.); f{i,2}=trt_code_dict;
  END;
  DO i=1 TO &xadverse.;
    SET adverse;
    trt_code='';
    DO j=1 TO &xpatdata.;
      IF subject=INPUT(f(j,1),best.) THEN DO;
        trt_code=f{j,2}; OUTPUT;
      END;
      IF ^MISSING(trt_code) THEN LEAVE;
    END;
    IF MISSING(trt_code) THEN OUTPUT;
  END;
  DROP i j trt_code_dict;
RUN;
```

7 MODIFY Statement

Using PATDATA with the unique records, to update ADVERSE using a MODIFY statement, then putting the data out to a new dataset ADVERSE2 – the MODIFY statement will not add TRTCD to the ADVERSE dataset, but the new dataset ADVERSE2 contains it:

```
DATA adverse adverse2;
  DO p=1 TO totobs;
    _iorc_=0;
    SET patdata point=p nobs=totobs;
    DO WHILE(_iorc_=%sysrc(_sok));
      MODIFY adverse KEY=subject;
      SELECT (_iorc_);
        WHEN (%sysrc(_sok)) DO; /*Match Found*/
          SET patdata POINT=p; OUTPUT adverse2; END;
        WHEN (%sysrc(_dsenom)) _error_=0; /*No Match*/
        OTHERWISE DO; /*A major problem somewhere*/
          PUT 'error: _iorc_ = ' _iorc_ /
          'program halted.'; _error_ = 0; STOP; END;
      END;
    END;
  END;
  STOP;
RUN;
```

Conclusion

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 site that you will see resource efficiencies between the methods.

Contact Information

Your comments and questions are valued and encouraged.

David Franklin
16 Roberts Road, Litchfield, NH 03052
Tel/Fax: 603-262-9160 Email: 100316.3451@compuserve.com
Web: http://ourworld.compuserve.com/homepages/dfranklinuk