

MERGING LARGE INTERNET AND PAPER SURVEY DATASETS FOR A 22-YEAR LONGITUDINAL COHORT STUDY

Thomas E Corbeil, MCS; Tyler C Smith, MS; Besa Smith, MPH;
Margaret AK Ryan, MD MPH

Department of Defense Center for Deployment Health Research at the
Naval Health Research Center, San Diego, CA

ABSTRACT

Merging two disparate datasets is a fairly common necessity that can be easily performed by most analysts who have a passing familiarity with SAS® software. Those who have actually taken this step, however, know that there are seemingly endless pitfalls that must be avoided in order to successfully produce the desired dataset. Using a real dataset example, this paper describes the rudiments of PROC IMPORT, ODBC calls, and the MERGE and SET statements. These SAS functions have been successfully used to merge paper-collected survey data with internet-collected survey data, producing a dataset containing nearly 80,000 survey observations and over 350 variable fields.

INTRODUCTION

Effective data analysis begins with good data. You may have written the SAS version of the Great American Novel, but if your data have been imported or modified incorrectly, your best efforts at analysis will inevitably be frustrated – or just plain wrong. This unpleasant reality has become particularly important to me while working on the Millennium Cohort Study, a survey-based project that has, in its opening stages, brought in roughly 80,000 surveys, including both internet and paper survey responses. With this study serving as a continuing example, I will use this paper to discuss some of the abilities of PROC IMPORT and other procedures for converting external data into usable SAS datasets, and also to investigate appropriate methods of combining these separate datasets once they have been brought into the SAS environment.

GETTING LARGE DATASETS INTO SAS

There are several different ways of getting your data into SAS, whether you are typing it in directly or importing it. With a dataset as large as the one in our example, your choices become more limited, but there are still a variety of options, one or more of which may be particularly suitable for your unique circumstances.

Option #1: PROC IMPORT for text files

If the data you are trying to import are in a text file format, the procedure for transferring this data into a SAS dataset is fairly simple:

```
proc import DATAFILE= 'c:\Survey\web.txt'  
OUT=webdata DBMS=CSV REPLACE;
```

In this case, data from internet survey respondents have been saved in a text file with comma-separated values (CSV). The DATAFILE statement reveals the location of the file to be imported. The OUT option names the new SAS dataset.

Notice here the file extension for our external data - .txt. PROC IMPORT will accept this file format as long as the fields in the file are tab-delimited. Our file, however, has comma-separated values, which leads to the need for a DBMS (database management system) specification. As you can see in the line of code above, CSV is the correct identifier for this field.

If your text file is neither tab-delimited nor comma-separated, you must use the DLM option for your DBMS specification:

```
proc import DATAFILE= 'c:\Data\web.txt'
OUT=webdata DBMS=DLM REPLACE;
DELIMITER='delimiter';
```

The DBMS=DLM option tells SAS that your text file is delimited by a character other than a comma or a tab; the DELIMITER option tells SAS which character you are using to delimit your variable fields.

Finally, the REPLACE option tells SAS to overwrite any existing SAS dataset with the same name as the one specified by the OUT option; if you do not use REPLACE, SAS will not overwrite the previously existing set.

Option #2: PROC IMPORT for other file types

In addition to text files, PROC IMPORT can also read in files from other applications. If you are using SAS/ACCESS, you can use the following specifications:

<i>Available DBMS Specifications*</i>		
Identifier	Input Data Source	Extension
ACCESS	Microsoft Access 2000 or 2002 table	.MDB
ACCESS97	Microsoft Access 97 table	.MDB
DBF	dBASE file	.DBF
WK1	Lotus 1 spreadsheet	.WK1
WK3	Lotus 3 spreadsheet	.WK3
WK4	Lotus 4 spreadsheet	.WK4
EXCEL	Excel 2000 or 2002 spreadsheet	.XLS
EXCEL4	Excel Version 4 spreadsheet	.XLS
EXCEL5	Excel Version 5 spreadsheet	.XLS
EXCEL97	Excel 97 spreadsheet	.XLS

* Table taken from SAS OnlineDoc, v. 9: "PROC IMPORT Statement." Copyright (c) 2002 SAS Institute Inc., Cary, NC, USA.

In the Millennium Cohort study, all paper surveys are scanned into the computer, and the resulting data are stored in an Access database. To retrieve these data, the following code could be used:

```
proc import DATAFILE='c:\Data\MilCo.mdb'
TABLE='PaperData.mdb' OUT= paperdata
REPLACE;
```

The only difference in coding here is the TABLE option; for Access tables, you must specify the name of the specific table that you want to import from the database that is named in the DATAFILE option. For files with the right extensions named in the DATAFILE and TABLE options, no DBMS specification is required.

One last important note: when using PROC IMPORT with text, Excel, or Lotus files, variable names are picked up from the first row in the file. If you do not want this, you must specify the GETNAMES=NO option.

Option #3: using ODBC

ODBC (Open Database Connectivity) is an easy alternative to the IMPORT procedure if you do not have SAS/ACCESS and need to import data from a SQL-compatible database (yes, Excel files also work with ODBC). With ODBC, you merely specify a call on an external data source in your LIBNAME statement. In order for this to happen, however, you must set up your ODBC specifications. Assuming you are working in a Windows environment, you can access the Data Sources (ODBC) icon in the Administrative Tools folder of your Control Panel; this will enable you to configure the connections that you might need.

Once your ODBC is set up, you simply need to use the DSN (data source name) in your LIBNAME statement:

```
libname MilCohort odbc;
```

Use this libname like you would any other:

```
data paper;
set MilCohort.tb1PaperData;
```

Because the ODBC reference in the LIBNAME statement has already referenced the database where the table is located, all that had to be done in the data step was the identification of the desired table in the database. In this particular situation, I wanted to make these data available to other programmers as well:

```
libname CommonData 'x:\Mill Cohort';
libname MilCohort odbc;

data CommonData.paperdata;
    set MilCohort.tblPaperData;
run;
```

I have now created a permanent dataset stored in the 'Mill Cohort' folder on a network drive (drive X). (For more information on accessing Excel or Access through the LIBNAME statement, see the paper by Kee Lee cited in the references.)

This is by no means an exhaustive list of options for getting your data into the SAS environment. There are plenty of other methods out there, but let us now move on to the subject of what to do with disparate datasets once they've been converted into SAS format.

MERGING SAS DATASETS

Now that both the web data and the paper data have been converted into SAS datasets, we need to find a way to combine the two sets into one large dataset for analysis. Depending on the nature of our data (and what we want to achieve), there are a few different ways this can be accomplished. A quick note: before taking any drastic measures, make sure that your corresponding variables share the same names, formats, and lengths!

Using the SET statement

You can use the SET statement within your DATA step to accomplish either of two things: concatenating or interleaving.

1. Concatenate. When you concatenate two or more datasets, you are simply stacking them one on top of the other. Suppose that we were interested in concatenating the paper and web datasets:

ID	ZipCode	Age	Service
20045	92103	26	Army
30100	04005	.	AirForce
10547	85308	34	Navy
19999	99163	40	Marines

ID	ZipCode	Age	Service	Shirt
12512	27513	43	Navy	L
14053	87102	18	Army	XL
20687	92111	28	Army	M
30100	45015	55	AirForce	XL

The code for concatenating is very simple:

```
data both;
    set paperdata
        webdata;
run;
```

This will result in a dataset that looks like this:

ID	ZipCode	Age	Service	Shirt
20045	92103	26	Army	.
30100	04005	.	AirForce	.
10547	85308	34	Navy	.
19999	99163	40	Marines	.
12512	27513	43	Navy	L
14053	87102	18	Army	XL
20687	92111	28	Army	M
30100	45015	55	AirForce	XL

Starting December 2001, individuals who completed the survey online were offered a free T-shirt; the SHIRT field contains the size of the T-shirt that they requested. Because this variable was not included in the paper dataset, the value for this field is set to missing for paper data records when the two datasets are combined.

2. Interleave. Interleaving is different from concatenating in that the records in the combined dataset are sorted based on the variable(s) specified in the BY statement:

```
proc sort data=paperdata;
    by ID ZipCode;
data both;
    set paperdata
        webdata;
    by ID ZipCode;
run;
```

Datasets must be sorted prior to interleaving; our web data were already sorted by id and zip code, but the paper data were not – this is why PROC SORT appears before the DATA step above.

Here is the resulting set:

ID	ZipCode	Age	Service	Shirt
10547	85308	34	Navy	.
12512	27513	43	Navy	L
14053	87102	18	Army	XL
19999	99163	40	Marines	.
20045	92103	26	Army	.
20687	92111	28	Army	M
30100	04005	.	AirForce	.
30100	45015	55	AirForce	XL

Because our last two records share the same ID number, they are sorted by ZipCode, the second variable specified in the BY statement.

Using the MERGE statement

The MERGE statement takes interleaving to the next level. While it too requires a BY statement (since it also sorts records) and only combines datasets that have already been sorted, the difference between the two methods is that the MERGE statement overlays data from the first set with data from the second.

Suppose, for people that submitted a survey both on paper and on the internet, we want to eliminate the paper survey and use only the data submitted on the internet for that person:

```
data both;
  merge paperdata
        webdata;
  by ID;
run;
```

Because WEBDATA is our second dataset, it will overwrite variables with the same names in the PAPERDATA set with its own values:

ID	ZipCode	Age	Service	Shirt
10547	85308	34	Navy	.
12512	27513	43	Navy	L
14053	87102	18	Army	XL
19999	99163	40	Marines	.
20045	92103	26	Army	.
20687	92111	28	Army	M
30100	45015	55	AirForce	XL

If we wanted to do it the other way around, with paper as the primary data, we could simply swap the two dataset names in the MERGE statement, with the following result:

ID	ZipCode	Age	Service	Shirt
10547	85308	34	Navy	.
12512	27513	43	Navy	L
14053	87102	18	Army	XL
19999	99163	40	Marines	.
20045	92103	26	Army	.
20687	92111	28	Army	M
30100	04005	.	AirForce	XL

Notice in the last record, the variable AGE has a missing value; even though this variable was present in the WEBDATA record, it was overwritten as a missing value. The variable SHIRT, however, does not appear in the PAPERDATA set, so the value it has in WEBDATA remains.

Using the UPDATE statement

Suppose that we wanted to keep the web data for subject #30100, but we find that the zip code from the paper survey is the most recent. In this situation, the UPDATE statement is our best option:

```
data both;
  update webdata
        paperdata;
  by ID;
run;
```

The UPDATE statement operates similarly to the MERGE statement, with the exception that it does not overwrite nonmissing values with missing values. Using the UPDATE statement will produce the following result:

ID	ZipCode	Age	Service	Shirt
10547	85308	34	Navy	.
12512	27513	43	Navy	L
14053	87102	18	Army	XL
19999	99163	40	Marines	.
20045	92103	26	Army	.
20687	92111	28	Army	M
30100	04005	55	AirForce	XL

CONCLUSION

Importing and combining datasets are two basic procedures, necessary for everyday SAS programming, that can quickly become very complex and confusing. Keeping the basics of these methods in mind can help in making sure that your final dataset is the one you imagined it would be!

SAS software is a registered trademark of SAS Institute, Inc. in the USA and other countries.

REFERENCES

Delwiche, Lora D. and Slaughter, Susan J., *The Little SAS Book: A Primer*, 2nd ed. Cary, NC: SAS Institute Inc., 1998.

Lee, Kee, "Accessing MICROSOFT EXCEL and MICROSOFT ACCESS Through the Use of a Simple Libname Statement." SAS Users Group International, 2002.

(<http://www2.sas.com/proceedings/sugi27/p025-27.pdf>)

SAS Online Documentation, version 9: "PROC IMPORT Statement." Cary, NC: SAS Institute Inc., 2002.

CONTACT INFORMATION

Tom Corbeil
Department of Defense Center for
Deployment Health Research, Naval Health
Research Center, San Diego
(619) 553-7598
corbeil@nhrc.navy.mil

ACKNOWLEDGEMENTS

Thank you to Bill Honner and Jim Whitmer for all of their assistance with this paper.

Approved for public release: distribution unlimited.

This research was supported by the Department of Defense, Health Affairs, under work unit no. 60002.