Paper 1116-2017

# Protecting the Innocent (and Your Data)

Stanley E. Legum, Ph.D., Westat, Rockville, MD

## ABSTRACT

A recurring problem with large research databases containing sensitive information about an individual's health, financial, and personal information is how to make meaningful extracts available to qualified researchers without compromising the privacy of the individuals whose data is in the database. This problem is exacerbated when a large number of extracts need to be made from the database. In addition to using statistical disclosure control methods, this paper recommends limiting the variables included in each extract to the minimum needed and implementing a method of assigning request-specific randomized IDs to each extract that is both secure and self-documenting.

## INTRODUCTION

In addition to collecting, cleaning and organizing large amounts of data, longitudinal cohort studies need to make the data available to researchers, policy makers, and other interested parties while protecting the privacy of cohort members. Data users who are not the primary researchers sometimes are granted access to limited subsets of the data. Both primary researchers and other data users often need to view the data at different points in the project's life. This leads directly to the need to establish a system for keeping track of which version of the data is being analyzed and distributed each time the data are accessed. Wide and multiple distributions of project data lead to the need to devise means for protecting the identities of cohort members. One technique for doing this is to randomize participant identifiers differently for each distribution of the data; but to do so in a manner that can be replicated when needed.

## VERSIONING DATA

There are many reasons that there can be multiple versions of data, including:

- New data
- Modifications to existing data

### NEW DATA

New data collection can occur because more cohort members are added during the course of the study or because additional data are collected about cohort members as when a follow-up survey is given to the entire cohort or a subset of the cohort. New data can also come by linking to central databases such as the National Death Index (NDI), the Social Security Death Master File (DMF), state or regional cancer registries, and the United States Renal Data System (USRDS), among others.

### MODIFICATIONS TO EXISTING DATA

Modifications to existing data can occur because new or revised data cleaning rules are adopted. For instance, after reviewing the responses to an "Other – Specify" question, the project decides that it is possible either to opcode some of the replies to existing codes or to create a new response category for some of the responses. Modifications can also occur when rules are adopted for creating one or more derived variables. Changes in contact information (new address, new or additional phone numbers) or demographic status (marriage, divorce, death) can also create a need to update data files.

### RECOMMENDATION: KEEP SAME FILE NAMES; BUT CREATE VERSION NUMBERS

Once analysts have started working on data from a project, it is often best to keep the same file names when an updated version of the file is released. This allows the same analysis code to be re-run without having to hard code the names of the newly released files or to set up a mechanism for reading the file names from an external data source.

To avoid massive confusion, it is necessary to have a plan in place for storing both old and new files in such a way that the custodians of the data can quickly and easily recognize with which version of a file they are working. It is also necessary to provide analysts using the files with clear information about which version of each file they are using. It is recommended that a version numbering system be defined early in the project that assigns a clear version number to each file. A versioning plan should consider such things as major versus minor version changes. It is also necessary to document what has changed. Programming specifications should clearly reference the release numbers. Depending on the circumstances, user documentation can be a matter of a text document containing release notes, an updated user manual, or both. Once the versions numbers are assigned to a release, it is important that all files, the associated specifications and documentation, and any reports or publications using these files display the version number.

An easy way to associate a version number with a SAS® file is to place the version number in the internal SAS file label for that file. This label is automatically printed out in CONTENTS procedure listings (see Figure 1) and is available in the Display Manager Properties window for a file (see Figure 2).

| | | | |
|---|---|---|---|
| **Data Set Name** | INDIR.TEST_DATA | **Observations** | 20 |
| **Member Type** | DATA | **Variables** | 4 |
| **Engine** | V9 | **Indexes** | 0 |
| **Created** | Friday, October 10, 2014 03:06:39 PM | **Observation Length** | 32 |
| **Last Modified** | Friday, October 10, 2014 03:06:39 PM | **Deleted Observations** | 0 |
| **Protection** | | **Compressed** | CHAR |
| **Data Set Type** | | **Reuse Space** | NO |
| **Label** | This is a sample internal SAS dataset label -- Version 2.1 | **Point to Observations** | YES |
| **Data Representation** | WINDOWS_32 | **Sorted** | NO |
| **Encoding** | wlatin1 Western (Windows) | | |

**Figure 1. Top of PROC CONTENTS Output Showing Label Field**

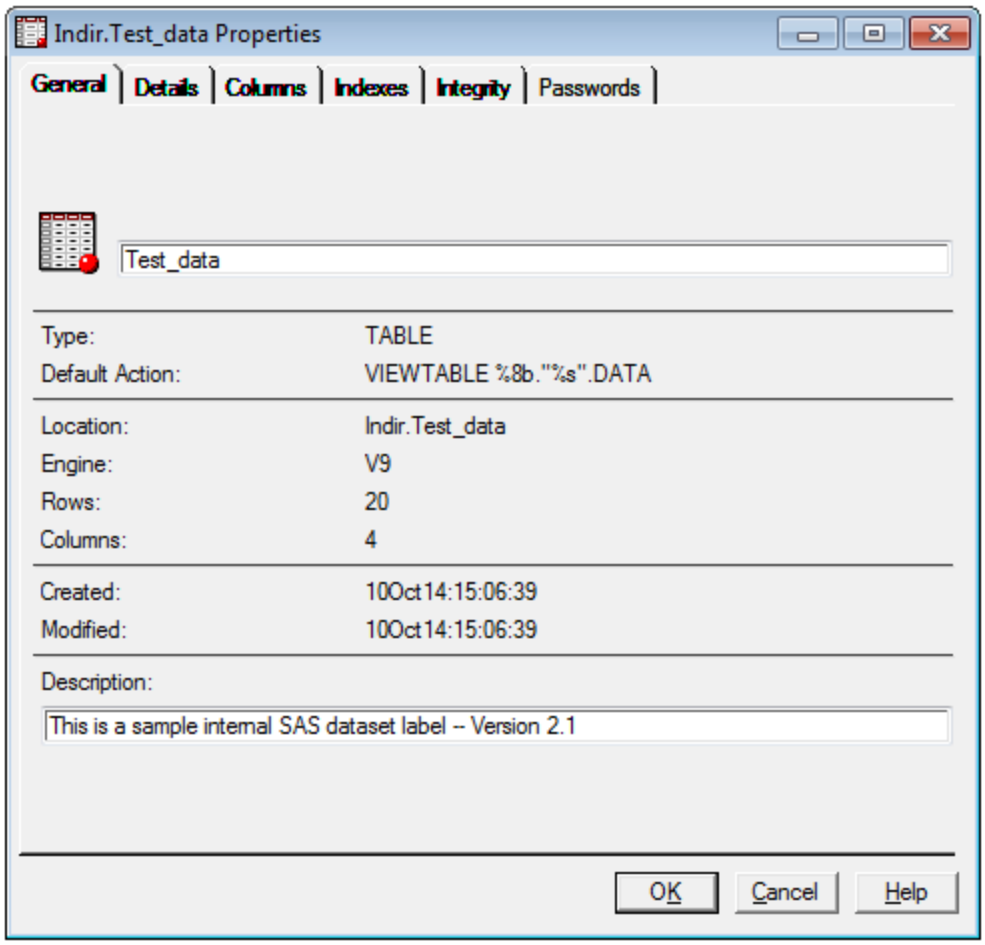


**Figure 2. Display Manager Screen Showing Description Field**

It is also possible to use PROC SQL to read the label from Dictionary Tables and display it in a title or footnote (see Legum (2009)).

It is important that researchers identify which version of a file they use in published papers. Some studies include a requirement to do so in data use agreements completed by researchers. The version references in papers provide a means of clarifying whether small differences in research findings are a result of different analysis procedures or the use of different versions of datasets. This means that the version numbers of data sources used in creating a data extract should be included in the internal dataset label for the extract and should be included in any transmittal letter sent to researchers with the extract.

## TYPES OF DATA REQUESTS

In order to protect the privacy of cohort members, projects need to establish policies regarding access to data. With the exception of a core group, it is usually a good idea to restrict the data to the variables and cohort members which researchers need to perform specific, well-defined analyses. There are commonly three main groups who have a need to access data:

- The group planning and conducting the longitudinal study, including the data collection, data processing, and data analysis staff working with them;
- Freedom of Information Act (FOIA) requestors;

- Government or academic researchers pursuing specific hypotheses.

Usually there are very few restrictions placed on the core planning and processing staff. However, it is common to insulate them from direct access to identifying information such as names, addresses, and social security numbers. FOIA requestors are likely to be requesting the data used to produce the values in published papers. To be prepared for such requests, it is important to keep archived copies of input datasets used for the published analyses. It is also important to keep any routines that produced derived or recoded data (e.g., code to group ages into 5-year age categories) used in the published analyses.

When government or academic researchers outside of the core group are granted access to the data they need to have a mechanism for specifying the subset of the cohort they wish to explore (e.g., all women in a specified age range with a specified medical condition) and the subset of variables that they need for their analyses. For both the FOIA requestors and the non-core group researchers, it is good practice to replace the standard participant IDs with a randomly assigned ID. This disrupts any attempt to combine multiple data sets to gain unauthorized information about participants and thus lowers the risk of inadvertently allowing a participant's identity being discovered. In the examples below, the random participant identifier will be represented by a variable named "RPID."

Often, once a researcher has started analyzing the data in an extract file, he or she realizes that some additional variables are needed. In order to be able to provide the additional variables, it is necessary to keep a crosswalk linking the standard participant identifiers with the randomly assigned ones. This crosswalk needs to be uniquely associated with a specific request and needs to be stored in a secure location.

The examples below use a dataset named "TEST_DATA" that resides in a directory with the LIBNAME "INDIR". It consists of 20 records containing a two-part identifier composed of the variables HOUSE_ID and PERSON_ID plus some data associated with each person. Figure 3 below shows the test data.

| Obs | HOUSE_ID | PERSON_ID | x | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 3 |
| 2 | 1 | 2 | 4 | 5 |
| 3 | 2 | 1 | 4 | 6 |
| 4 | 2 | 2 | 8 | 10 |
| 5 | 3 | 1 | 6 | 9 |
| 6 | 3 | 2 | 12 | 15 |
| 7 | 4 | 1 | 8 | 12 |
| 8 | 4 | 2 | 16 | 20 |
| 9 | 5 | 1 | 10 | 15 |
| 10 | 5 | 2 | 20 | 25 |
| 11 | 6 | 1 | 12 | 18 |
| 12 | 6 | 2 | 24 | 30 |
| 13 | 7 | 1 | 14 | 21 |
| 14 | 7 | 2 | 28 | 35 |
| 15 | 8 | 1 | 16 | 24 |

| | | | | |
|---|---|---|---|---|
| **16** | 8 | 2 | 32 | 40 |
| **17** | 9 | 1 | 18 | 27 |
| **18** | 9 | 2 | 36 | 45 |
| **19** | 10 | 1 | 20 | 30 |
| **20** | 10 | 2 | 40 | 50 |

**Figure 3. Test Data**

## GENETRATING A RANDOMIZED CROSSWALK

The following code (Example 1) accomplishes the task of generating a randomized crosswalk.

```
DATA randomized ;
  set INDIR.test_data (keep = house_id person_id ) ;
  random_order = ranuni(-1) ;
RUN ;

PROC SORT data = randomized ;
  by random_order ;
RUN ;

DATA OUTDIR.XWALK (drop = random_order
                  label = "Crosswalk of Random IDs [RPID] to Study IDs"
                  ) ;
   length RPID $6. ;
   set randomized ;
   RPID = put(_N_,z6.) ;
   label RPID = "Randomized Participant ID" ;
RUN ;
```

**Example 1. Basic code to create a crosswalk with a random ID variable**

Figure 4 displays a randomized crosswalk generated by this code.

| Obs | RPID | house_id | person_id |
|---|---|---|---|
| **1** | 1 | 4 | 2 |
| **2** | 2 | 10 | 1 |
| **3** | 3 | 10 | 2 |
| **4** | 4 | 5 | 1 |
| **5** | 5 | 7 | 2 |
| **6** | 6 | 5 | 2 |
| **7** | 7 | 6 | 1 |
| **8** | 8 | 9 | 2 |
| **9** | 9 | 3 | 1 |

| | | | |
|---|---|---|---|
| **10** | 10 | 8 | 2 |
| **11** | 11 | 6 | 2 |
| **12** | 12 | 2 | 1 |
| **13** | 13 | 4 | 1 |
| **14** | 14 | 9 | 1 |
| **15** | 15 | 7 | 1 |
| **16** | 16 | 3 | 2 |
| **17** | 17 | 8 | 1 |
| **18** | 18 | 1 | 2 |
| **19** | 19 | 2 | 2 |
| **20** | 20 | 1 | 1 |

**Figure 4. Sample Crosswalk with Random ID (RPID)**

It is a simple matter to use a crosswalk such as this to add the random ID (RPID) to a file and then delete the existing identifiers. If it is ever necessary to reverse the process, you can use the crosswalk to add the original identifiers back onto the file and then, if desired, remove the RPID. This situation occurs when a data file with randomized IDs is submitted to an outside agency such as the NDI for matching to its database and the data extracted from the outside database is ready to be incorporated into the project files.

The code in Example 1. has a problem. If you run it again, you will not get the same crosswalk. The variable names will be the same; but the original IDs will appear in a different order. This happens because when RANUNI is given a negative seed value (the -1 in Example 1.) it uses the system clock to generate the seed used when assigning the values of RANDOM_ORDER. We could overcome this problem by assigning a positive seed value in the code; but then all the randomized orders generated by the code would be the same. This situation could lead to the ability to merge separate extracts from the master datasets using these “random” IDs, a situation we would like to avoid.

The solution to this problem is to let the system assign the initial seed number and capture it in a way that can be used later. We can also generalize the code to use a known seed if we ever need to regenerate a crosswalk. This can be accomplished by putting the crosswalk code in a macro and using the macro commands in Example 2 to select the seed (&SeedNum) used to start the random number generation process.

```
%LET RSTART = -1 ;  *** To regenerate a prior crosswalk, replace -1 with known seed ;
%local SeedNum ;
%IF "&RSTART" = "-1" %THEN   %LET SeedNum = %SYSFUNC(ranuni(&RSTART)) ;
%ELSE %IF %SYSFUNC(verify(&RSTART,'0123456789')) = 0 %THEN %LET SeedNum = &RSTART ;
```

**Example 2. Code to Use Either a Known Seed or the Default Value of -1**

In this example, if &RSTART is -1, then the code segment %SYSFUNC(ranuni(&RSTART)) generates a random number that is captured in &SeedNum. If &RSTART is a positive number, then &SeedNum is set to the value of &RSTART. In either case &SeedNum is the value used to create the crosswalk file with the randomized IDs. It can be displayed in the log and, if desired, included in an output file.

It is also prudent to provide a means of informing the person running the program if an invalid value is given for &RSTART. The code in Example 3 accomplishes this.

```
%ELSE %DO ;
    %PUT The value for RSTART is &RSTART.. ;
    %PUT It should be either -1 (the default) or a positive integer less than 2^31-1. ;
    %PUT Processing cancelled. ;
    %LET RSTART = CANCEL ;
 %END ;
 %IF "&RSTART" NE "CANCEL" %THEN %DO ;
        <insert the code from Example 1. here>
 %END ;
```

**Example 3. Code to Report Invalid Values for &RSTART**

To document the seed value used in the log, add the lines in Example 4 before the call to the RANUNI function.

```
        if _N_ = 1 then do ;
            seed_val = &SeedNum  ;
            put " --- " seed_val =   ;
        end;
```

**Example 4. Code to Print Seed Value in the Log**

## ENSURING THAT PROGRAM DOES NOT OVERWRITE EXISTING CROSSWALK FILE FOR SAME REQUEST

Once a crosswalk has been generated and used to produce a data extract to satisfy a data request, it is important not to accidentally overwrite the crosswalk. Placing the code in Example 5 at the beginning of the macro that generates the crosswalk guarantees that this will not occur.

```
%IF %sysfunc(exist(OUTDIR.XWALK)) %THEN %DO ;
    %PUT SAS file XWALK already exists in target directory ;
    %PUT It will NOT be overwritten. ;
    %PUT ;
%END ;
%ELSE %DO;
        <insert code to generate randomized crosswalk here>
%END ;
```

**Example 5. Logic to Ensure That an Existing Crosswalk Is Not Overwritten**

If responding to requests for data extracts is a common occurrence in a project, it can simplify everyone's work to develop a numbering system for uniquely identifying the request. The request number can then be used in tracking all approval and delivery steps associated with the request. It can also be assigned to a subdirectories used only for processing the request and holding the crosswalk and the final extract. It is an easy matter to create macro variables for the output directory (OUTDIR), the data extract files, and, if desired, the name of the crosswalk used in the extract.

## CONCLUSION

In large studies, it is important to use randomized identifiers when responding to requests for data extracts from sources other than core researchers. This prevents extracts from being combined in an attempt to identify the study participants. It is important to add version numbers to source files and to include these numbers in the internal SAS file labels of extracts from those files. Randomizing identifiers is only one facet of a study's procedures to protect the identity of study participants; but it is an important one and is easy to implement

## REFERENCES

Legum, S. E. (2009) "Using Data Set Labels to Make Your SAS® Output Self Documenting," North East SAS Users Group Inc 22nd Annual Conference Proceedings, Burlington, Vermont, September 2009. http://www.lexjansen.com/nesug/nesug09/bb/BB04.pdf

## ACKNOWLEDGMENTS

Many thanks to Michael Raithel for his review and suggestions

## CONTACT INFORMATION

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

Stanley Legum
Westat
1600 Research Blvd.
Rockville, MD 20850
selegum@Westat.com

## DISCLAIMER

The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.