

I Have a Secret: How Can I Hide Small Numbers from Public View?

Fred Edora, South Carolina Department of Education, Columbia, SC

ABSTRACT

Due to many federal, state, and local regulations, maintaining privacy is not simply a requirement but also of utmost importance when reporting education data to the public. Special education data is no exception. Because of the highly sensitive nature of data associated with students, we must be vigilant in ensuring that data is transparent while also meeting all necessary privacy regulations. In most cases surrounding publicly reported data, numbers 10 and under must stay hidden and cannot be reversed engineered. How can this be achieved? SAS® is no stranger to privacy and has multiple methods on hiding small numbers in publicly reported data. This paper will cover those methods and the pros and cons of each, and is intended for beginning or intermediate SAS users.

INTRODUCTION

In education, we want to be able to make decisions based on valid and reliable data. Student data is highly valued and can be accessed through a variety of sources. However, the act of suppressing data varies by agency and regulation. State education staff reporting data must be sensitive to federal, state, and local laws and regulations protecting the privacy of individual level student data (also known as Personally Identifiable Information, or "PII"). To do this effectively, it is necessary to understand different methods within SAS to suppress small numbers so that the public cannot view the data or make possible conclusions ("reverse engineer") from the data. Data is suppressed in public reports in various ways. This paper will discuss two methods within SAS to suppress small values.

METHOD 1: USING PROC FORMAT TO MASK SMALL VALUES

Option one is to create a numeric format that hides data based on the values that are set in the format statement, while preserving the original data within the dataset. Within the proc format statement, the specified values are replaced by a character of the programmer's choice. In the example below, the format "hide" is created, values 1-5 are replaced by an asterisk, and all other numeric values are shown with a "comma8." format:

```
proc format;
  value hide
    1-5 = '*'
    other = [comma8.];
run;
```

You can then use this format to hide the numbers within the report. Three common procedures where you can apply the new format include PROC REPORT, PROC FREQ, and PROC TABULATE. Below is a sample dataset as well as how each of procedures handle the custom format.

```
data htest;
input ID school $ age grade $;
cards;
3433 ABC 7 A
2220 ABC 8 A
1833 ABC 6 A
1838 ABC 9 D
9998 ABC 7 B
1348 ABC 7 A
1488 ABC 7 D
3842 ABC 6 A
2310 ABC 8 C
1113 ABC 7 B
4831 ABC 9 A
4411 DEF 12 A
```

```

9988 DEF 13 A
2135 DEF 13 A
3821 DEF 12 B
1893 DEF 11 B
3841 DEF 13 B
0932 DEF 13 C
1233 DEF 13 A
7734 DEF 12 A
1012 DEF 13 D
2383 DEF 12 B
4831 DEF 12 B
8099 DEF 13 B
3423 DEF 13 B
4098 GHI 15 B
9982 GHI 18 A
2093 GHI 17 B
9983 GHI 17 A
3283 GHI 16 B
1024 GHI 14 F
3810 GHI 16 B
0976 GHI 14 B
1293 GHI 15 A
1384 GHI 15 A
1121 GHI 14 D
8334 GHI 18 F
;
run;

```

PROC FREQ

The first procedure where you can apply the custom format is PROC FREQ. This can be used when you have an n-way table or a cross-tabulation. In the resulting output, the custom format is applied to the data values as well as the cumulative totals if the values are between one and five.



Caution #1: When using PROC FREQ, you cannot use the custom format for a one-way frequency distribution. Attempting to use the format will have no effect on the resulting table. If you need to hide numbers in a one-way frequency table, it's better to use PROC REPORT.

```

proc freq data=htest;
    table grade*school / missing norow nocol nocum nopercnt format=hide.;
run;

```

Table of grade by school					
		school			Total
		ABC	DEF	GHI	
grade					
A	Frequency	6	*	*	15
B	Frequency	*	7	*	14
C	Frequency	*	*	0	*
D	Frequency	*	*	*	*
F	Frequency	0	0	*	*
Total	Frequency	11	14	12	37

Display 1. All of the values between 1-5 are hidden with an asterisk as noted in the PROC FORMAT statement.

PROC TABULATE

The following code produces a table with two columns, “Freq” and “Masked,” which illustrate the difference as to how the custom format treats numbers within the report. The first column shows all the values from the htest dataset, while the second column shows the values with the custom format being applied. This code can be useful if you only need to mask certain columns (versus masking all columns in the report).

```
proc tabulate data=htest;
  class grade;
  table grade, N='Freq' N='Masked'*F=hide.;
run;
```

	Freq	Masked
grade		
A	15	15
B	14	14
C	2	*
D	4	*
F	2	*

Display 2. The “Freq” column shows all values, while the “Masked” column shows values with the custom format being applied.

PROC REPORT

```
proc report data=htest;
  column grade n;
  define grade / group missing;
  define n / format=hide.;
run;
```

grade	n
A	15
B	14
C	*
D	*
F	*

Display 3. All values between 1 and 5 are masked with the asterisk.

Within the REPORT procedure, you can also use the COMPUTE statement to apply the format defined in the COLUMNS statement. In special education, a common variable that is used is the “Least Restrictive Environment” variable which defines the specific educational environment of a student receiving special education services in the public school system. This variable is of interest in federal reporting as well as with academic researchers. Below is an example using the ‘ReportingLREPlacement’ variable within a program created for our datasets which deliver reports using special education student data. In order for the code to work properly, the format must be used in a COMPUTE and CALL DEFINE statement.

```
proc format;
  value suppress
  0-10 = '*'
  . = '*';
run;
```

The “suppress” custom format is defined here before running the PROC REPORT code.

```

proc report data=***testdataset*** (where=(AgeChildCount ge 6)) completecols
completerows spanrows;
  columns AgeChildCount ReportingLREPlacement, (n pct);
  define AgeChildCount / 'Age (as of the Child Count Date)' id
  group;
  define ReportingLREPlacement / 'Least Restrictive Environment'
  across;
  define n / 'Count';
  define pct / computed 'Column Percent' f=percent8.;
  rbreak after / summarize ol ul;

```


.
. OTHER SAS CODE
.

This “suppress” format was defined to hide values 10 or below within the defined ‘ReportingLREPlacement’ variable

```

compute ReportingLREPlacement;
  call define (_col_, 'format', 'suppress. ');
endcomp;


```

 **Caution #2:** Because this method only hides the numbers behind the asterisk, reverse engineering is still possible because calculated totals do not take into account the values suppressed by the formatting procedure. This may not be an issue if there are multiple suppressed cells (and thus, multiple small values) within the same row or column. However, if there is only one suppressed value in a row or column, a savvy person may be able to manually add up the sum of the remaining values, calculate the difference between the sum and the reported total, and determine the suppressed value from the difference.

How can we reduce the likelihood of someone being able to reverse calculate suppressed values?

METHOD 2: USE THE DATA STEP TO CHANGE SMALL VALUES TO MISSING

Another possible solution is to use the data step to turn values into missing values so that the data is both hidden from the dataset and smaller values won’t be used in larger cumulative totals that may be calculated using other SAS procedures. The next sample dataset shows how suppression is achieved. The dataset contains five variables: dist (school district code), county (county name), disability (letter codes A-D), num (number of students), and salary (average salary of all students in that location/disability). Before producing any reports, we want to remove all data that has 10 or fewer in the ‘num’ variable, along with the accompanying average salary for that disability and location.

 **Caution #3:** Backup the original dataset and use an alternate dataset prior to testing and/or implementing! If you do not backup the dataset, you will lose the original data.

Below is the SAS code using a sample dataset with test data to show how this is achieved:

```

data dtest;
  input @1 dist @3 county $4. @8 disability $2. @10 num @14 salary 5.;

```

```

datalines;
4 RICH A 350 25000
4 RICH B 7 14000
4 RICH C 65 16000
4 RICH D 80 31000
3 LEXI A 100 22000
3 LEXI B 12 14000
3 LEXI C 30 16000
3 LEXI D 10 19500
2 KRSH A 16 34000
2 KRSH B 2 27500
2 KRSH C 5 13000

```

Sample data variables

Variable	Meaning
dist	District Code
county	District County
disability	Disability Code
num	Number of Adult Students
salary	Average Salary of Employment

```

2 KRSH D 4 41500
1 NEWB A 31 24000
1 NEWB B 7 22000
1 NEWB C 14 21500
1 NEWB D 3 31000
0 FAIR A 8 18000
0 FAIR B 5 11000
0 FAIR C 4 12000
0 FAIR D 12 27500

```

```
;
```

```
run;
```

```

proc sort data=dtest;
  by dist disability num;
run;

```

```

data dtest;
  set dtest;
  by dist disability;

  if num ne . and num le 10 then do;
    num = .;
    salary = .;
  end;
run;

```

This changes the numeric value to missing, and thus "erases" the original value.



dist	county	disability	num	salary
0	FAIR	A	.	.
0	FAIR	B	.	.
0	FAIR	C	.	.
0	FAIR	D	12	27500
1	NEWB	A	31	24000
1	NEWB	B	.	.
1	NEWB	C	14	21500
1	NEWB	D	.	.
2	KRSH	A	16	34000
2	KRSH	B	.	.
2	KRSH	C	.	.
2	KRSH	D	.	.
3	LEXI	A	100	22000
3	LEXI	B	12	14000
3	LEXI	C	30	16000
3	LEXI	D	.	.
4	RICH	A	350	25000
4	RICH	B	.	.
4	RICH	C	65	16000
4	RICH	D	80	31000

Display 4: Resulting dataset in SAS showing the missing values

If you run procedures on this dataset, the procedure will take into account the new missing values. Here we are using PROC MEANS as an example to show the differences.

The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
num	20	38.2500000	78.4124554	2.0000000	350.0000000
salary	20	22025.00	8221.40	11000.00	41500.00

Before changing small numbers to missing

The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
num	10	71.0000000	102.9109648	12.0000000	350.0000000
salary	10	23100.00	6603.03	14000.00	34000.00

After changing small numbers to missing

Display 4: The effect of the missing values using the PROC MEANS statement

CONCLUSION

Protecting PII is of utmost importance, especially when reporting any type of government data. Education data is no exception. Many of us want data and reports to help us make important program and policy decisions but we must also balance convenience with security. SAS provides some flexibility in ways that you as a programmer can protect sensitive data in public reports. This paper provides some methods and procedures on how to protect PII and hide small numbers in SAS.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please feel free to contact the author for more information:

Fred Edora
 Team Lead, Data and Technology
 South Carolina Department of Education
 Office of Special Education Services
 1942 Laurel Street
 Room B107
 Columbia, SC 29201
 Phone: 803-734-0388
 Email: fedora@ed.sc.gov

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.