PharmaSUG 2021 - Paper QT-096

NOBS for Noobs

David B. Horvath, MS, CCP

ABSTRACT

This mini-paper (and the resulting short session) is a focused discussion of the NOBS (number of observations) option on the SET statement. This includes one "gotcha" that I've run into with where clauses: NOBS is set before WHERE processing. If you have a reason to know the number of observations after the WHERE clause, another DATA step is needed.

INTRODUCTION

There are multiple ways of determining the number of records (observations) within a SAS® dataset. These include: programmatic copying, data dictionary tables/views, proc dataset/contents, and the NOBS option on the SET statement. Because I got caught by the timing of when the count is determined so I researched the topic a little further.

- Basic NOBS
- Macro NOBS
- NOBS Catches

In my situation, I needed to kick off a series of rsubmit scripts to process a series of datasets in parallel. Since the number could change over time and I did not want to change that part of the code, I used NOBS to determine how many I had. Unfortunately, a time came when the business decided that we did not need to process certain datasets so I added a where clause. Although they did not process, some of my parallel scripts were empty because I ran out of records before reaching NOBS.

BASIC NOBS

The nobs statement is a handy way of discovering how many observations are in your SAS Dataset. In the following example, a single record dataset is created to demonstrate the process:

```
data simple;
a = 42;
output;
run;

data _null_;
put nobs=;
stop;
set simple nobs=nobs;
run;
```

This code snippet prints the following in the log:

```
NOBS=1
```

The key is the nobs= addition to the set statement. You get to select the variable that contains the value (in this case also “nobs”). The value is determined once and should not change during execution. In simplest terms, the variable will contain the number of observations (records) in the dataset.

MACRO NOBS

Of course, you can push the number of observations variable into a macro variable so it can be used later in the program (not just the current dataset). This behaves like any other created variable. It does turn out that the value is set before the first row is processed. This example shows creation and use:

```
data _null_;
call symput('ALLOBS', nobs);
stop;
set simple nobs=nobs;
run;

data _null_;
put  "number of obs are &ALLOBS.";
stop;
run;
```

The resulting log output is as follows:

```
number of obs are              1
```

You can pull the value in one data step and use it later on as necessary.

NOBS: CATCHES

WHERE CLAUSE

As I mentioned before, misunderstanding the impact of a where clause on the behavior of NOBS is got me researching the topic at a deeper level. It turns out that NOBS is set at the file level – before the where clause executes. The next two blocks of code end up with the same value for NOBS (1); the first without a where clause:

```
data _null_;
put nobs=;
stop;
set simple nobs=nobs;
run;
```

And the second with a where:

```
data _null_;
put nobs=;
stop;
set simple nobs=nobs;
where a = 10;
run;
```

But both print the same result:

```
NOBS=1
```

In my situation, I needed to get the nobs on the output of the step with the where. Because no records needed to be read, the added step to get the nobs from the resulting dataset took very little time.

NOT A NEW VARIABLE

Your nobs variable is special – it will not appear in the output dataset by default. In this example, you see that it does not appear in the NEW dataset when the proc print runs.

```
data new;
set simple nobs=nobs;
run;
proc print data=new; run;
```

Resulting output from proc print:

```
Obs     a
 1     42
```

Even adding the nobs variable in a keep statement is not a fix – that results in the error below:

```
data new (keep=a nobs);
WARNING: The variable nobs in the DROP, KEEP, or RENAME list has never been
referenced.
```

The only solution is to create yet another variable and assign the nobs= variable to the new one. That new one (nnobs in the code snippet below) is just like any other variable:

```
nnobs=nobs;
```

OPTIONS OBS=

Much like the NOBS value not being affected by where clauses, OPTIONS OBS also does not change the value. It doesn't matter whether you set OBS= in an OPTIONS statement or on the dataset itself. The following examples both produce the same result – NOBS shows the full number of obs in the file no matter what the OBS= statement says:

```
options obs=2;
data _null_;
put nobs=;
stop;
set large nobs=nobs;
run;
```

And (obs=)

```
data _null_;
put nobs=;
stop;
set large(obs=2) nobs=nobs;
run;
```

Both print

```
nobs=915803
```

NOT EVERY ENGINE

Probably the biggest catch with the NOBS statement is that not every engine supports it. And SAS will not tell you that it isn't supported. Not an ERROR:, no WARNING:, not even a NOTE! It may fail very quietly returning the value zero or it may return a ridiculously high value. In both of those cases, it is conceivable that the dataset is that size. In particular, I've noticed that the sequential engine returns zero and the XML Engine returns some random number as shown in this example:

```
filename  SXLEMAP "OUR_MAP_FILE.map";
```

```
filename  test2 "OUR_INPUT_FILE.xml";
libname   test2 xml xmlmap=SXLEMAP access=READONLY;

NOTE: Libref TEST2 was successfully assigned as follows:
      Engine:        XML
      Physical Name: TEST2

data _null_;
put nobs=;
stop;
set test2.application nobs=nobs;
run;
```

Prints the following in the log:

```
nobs=9.0071993E15
```

When the file only contained 17,383,357 bytes

CONCLUSION

NOBS can come in very handy helping you understand the size of your datasets so long as you understand the limitations and quirks.

ACKNOWLEDGMENTS

I want to thank the organizers of this great conference, my employer for their willingness to allow me to expand my horizons through events like this, and, last but certainly not least, my spouse Mary who doesn't complain when I spend so much time at the keyboard working on documents like this.

CONTACT INFORMATION

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

David B. Horvath, CCP
+1-610-859-8826
dhorvath@cobs.com
http://www.cobs.com
LinkedIn: https://www.linkedin.com/in/dbhorvath/