PROC REPORT – Land of the Missing OBS Column

Ray Pass
Forma Therapeutics

As full-featured as PROC REPORT is with all of its bells, whistles, bangles and bows, it's missing a few plain old-fashioned features that we are all accustomed to from our day-to-day mundane workings with PROC PRINT. One of these is the automatic printing of observation numbers in the standard output. They can be turned off in PROC PRINT with the `noobs` option, but there is no option to turn them on in PROC REPORT. That is not to say that it can't be done in REPORT; it's just not there as an option toggle switch.

Consider the following standard PROC PRINT code and it's resulting output:

```
*-------------------;
proc print data=test;
run;
*-------------------;
```

OBS	REPVAR1	REPVAR2	REPVAR3
1	aaa	726	736
2	bbb	187	675
3	ccc	365	746

Now here is the same dataset as displayed by the following vanilla PROC REPORT code:

```
*-------------------------;
Proc report data=test nowd;
Run;
*-------------------------;
```

REPVAR1	REPVAR2	REPVAR3
aaa	726	736
bbb	187	675
ccc	365	746

It's easy to see what's missing. Creating obs numbers in PROC REPORT is really very easy, but in order to understand the methodology needed to do so, we have to be aware of one of the basic building block constructs of the procedure. If you've been using the full power of PROC REPORT, you're aware that there are six types of variables in the procedure: DISPLAY, ANALYSIS, ORDER, GROUP, COMPUTED and ACROSS. There is however another dimension of variable types used in REPORT, but not one that is immediately obvious however because you don't actually use it to label variables. Variables that appear in a column statement, whether or not they actually appear in the procedure output, and whether or not they appear in a "compute block", are categorized in PROC REPORT as ***report variables***. Variables that appear *only* in a "compute block", and *not* in a column statement, and therefore not in the output, are known as ***DATA step variables***.

DATA step variables are used in the computation of COMPUTED variables. They may or may not have the same name as variables contained in the source data set for the procedure, but they do NOT

refer to variables in that source data set. A key differentiating feature that distinguishes ***DATA step variables*** from ***report variables*** is when they are initialized. ***DATA step variables*** are initialized to missing *once only* at the beginning of the construction of the report. ***Report variables*** on the other hand, are re-initialized to missing at the beginning of each output row of the report. It is this difference that allows us to create observation numbers.

Now consider the following code and its resulting output:

```
*-----------------------------------------;
proc report data = test nowd;

   column obs badobs repvar1 repvar2 repvar3;

   define obs     / computed;
   define badobs  / computed;
   define repvar1 / display;
   define repvar2 / display;
   define repvar3 / display;

   compute obs;
      dsobs + 1;
      obs = dsobs;
   endcompute;

   compute badobs;
      badobs + 1;
   endcompute;
run;
*-----------------------------------------;
```

OBS	BADOBS	REPVAR1	REPVAR2	REPVAR3
1	1	aaa	726	736
2	1	bbb	187	675
3	1	ccc	365	746

Let's examine the three variables `dsobs, obs` and `badobs.` Variable `dsobs` does not appear in the column statement, and therefore does not appear in the output. It is a ***DATA step variable*** and not a ***report variable***. It is therefore initialized to missing once only, at the beginning of the report processing, and is ***not*** re-initialized at the beginning of each output row. This variable, `dsobs,` is incremented by 1 at each row of the report via the `dsobs + 1;` statement in the compute block. Variable `obs` on the other hand, ***does*** appear in the column statement (and also in the output) and *is* a ***report variable***. It is therefore re-initialized at the beginning of each row of the report, but is then immediately set equal to the newly incremented value of `dsobs.` The last variable we want to examine, `badobs` is also a ***report variable***. It is therefore re-initialized to missing at the beginning of each row, and is then directly incremented by 1. Because it is always re-initialized to missing before being incremented, and because unlike `obs`, it is not set equal to the newly incremented and not re-initialized value of `dsobs`, it always has a value of 1.

In summary, to obtain an observation counter that can be output with the report, you use a ***DATA step variable*** as a counter by incrementing it from row to row. You then set a ***report variable*** equal to the incremented value of the ***DATA step variable*** at each row so that you can output the row counter.

Following this simple procedure will allow you to bring back the missing OBS. It is also however a valuable technique which can be used in many more interesting applications using PROC REPORT.

The author of this paper can be contacted as follows:

Ray Pass
e-mail: rpass@formatherapeutics.com
e-mail: raypass@att.net