Using the REPORT Procedure in a Batch Environment: Dealing with Format Limitations and RAM Requirements

Michael J. Marzano, Nabisco Biscuit Company, East Hanover, New Jersey

ABSTRACT

The emergence of the REPORT procedure offers another convenient method of creating batch reports. This procedure is more flexible than the PRINT, TABULATE, or MEANS procedures and more convenient and easier to use than a custom-coded DATA step with PUT statements. But convenience and flexibility are sometimes not enough. Two of the more limiting factors of the REPORT procedure are how it handles page/line “wrapping” and the extensive memory requirements required for processing large amounts of data. This paper will present a method of using a simple SAS Macro to address these two shortcomings to allow large amounts of data to process with minimal memory requirements and provide additional control over page/line wrapping.

INTRODUCTION

One of the key features of the Report procedure is the “ACROSS” variable which allows you to setup a report like a table with dynamic control of rows and columns. One of the problems typically encountered occurs when the number of columns exceeds the width of the report (linesize). When this happens, SAS will automatically “wrap” the remaining columns on to the next page, but SAS will not repeat the first column which typically identifies the column contents of that row.

SAMPLE DATA

As an example, suppose I am creating a report that lists inventory by item number for 26 locations. I’d like to see the item number as the first column and the locations would form the next 26 columns across the page. I could add an additional column to provide a total for that item number and I could add a total at the bottom of each column to show a total for each location.

The report was produced as designed ... with the exception that only 9 locations (columns) can physically fit on a page that is 133 characters wide. As a result, the remaining locations print on the following pages, but the item numbers do not. The last page (top-half) of this document contains an example of the second page of this report. As you can see, the information is useless unless you can match it up with the first page which would indicate the item each row of information were referring to. So you constantly need to refer back to the first page in order to determine that the numbers in the 15th row are for a certain item number. Obviously this can be confusing, time consuming, and inconvenient.

There is a solution that will allow PROC REPORT to handle situations like this. The “trick” is to make PROC REPORT think that every page is a “new” report so that it will print the item numbers in the first column rather than to understand it to be a continuation of the previous page. The key to this is to build in a mechanism that will “release” to the PROC REPORT step only those records for the amount of locations that may fit on a page. In other words, if there are 33 inventory records per location and there are 26 locations, then the file I have has 858 records or observations in it. If all 858 are passed to the PROC REPORT step at once, the result is the example of locations wrapping without repeating the item numbers. But if I could sort the 858 records (by location) and then make 3 passes at the PROC REPORT step, release 297 records (9 x 33) the first 2 passes and 264 (8 x 33) the third and final pass, I would now have a report looking the way I want it to.

The method to accomplish this is to embed a MACRO within the SAS code to establish the required “loop” to make the repetitive passes through the PROC REPORT step. But before this loop can be established, it must be determined how many locations (in total) there are and how many will fit across a page.
You can determine how many locations will fit across a page simply by counting the space required as specified in the DEFINE statement. To determine the total number of locations, a data step was added to count the number of locations and using the SYMPUT function, create a global parameter that contains the total number of locations. Now a DO loop can be created like:

```
%DO I = 1 %TO &NORECS %BY NOCOLS ;
```

This loop alone, however, will only create multiple copies of the same “ugly” report that existed before. In order to control the “release” of locations and records to the PROC REPORT step, the following data step must be added:

```plaintext
DATA SELECT ;
SET LOCATION ;

HILIM=&I+9 ;
LOLIM=&I ;

IF LOLIM<= _N_<HILIM ;
```

In this example, the work file LOCATION contains the locations that are to print. 9 locations or columns can fit across a page (in addition to the first column for item numbers). After executing, SELECT will contain the next 9 locations that are to be printed. SELECT will then be merged/matched with the data file and then passed on to the REPORT step to produce the report. Again, the final page of this document [bottom-half] contains an example of page 2 of a report now coded as outlined above. As you can see, this page 2 is much more meaningful and useful than the prior example.

This same method can be employed to reduce the amount of RAM required to complete the PROC REPORT step. I have had some large print jobs (printing a large amount of records) that have ABENDED in the MVS environment because they required additional memory beyond the 8 megabyte ceiling imposed by the data center. But by passing just a small number of records to PROC REPORT in multiple passes, the process completed with minimal memory requirements.

**CONCLUSION**

Using the PROC REPORT procedure in a batch (nonwindowing) environment offers many conveniences, but has two noticeable shortcomings in dealing with “page wrapping” and RAM

**SAS Products Referenced:** Base SAS software; SAS Macros; REPORT Procedure

**Operating Systems:** MVS; VM/CMS

**Intended Audience:** Beginner/Intermediate