Paper 229-29

Using New Features in ODS to Create Master/Detail Reports

Jack Hamilton

First Health

West Sacramento, California

Overview

- This paper will present a few approaches to creating master/detail reports – some old, some new.
- The intent is not to explain code in detail.
- The completed paper, with SAS code, will be available after SUGI from the web site listed in the paper.

Master/Detail Reports

- A Master/Detail report is a report which shows data from two related files.
- The example I will use today is a simplified invoice. The master file contains general invoice information, and the detail file contains line items.
- Master/detail reporting is very common. Class rosters are another example.

Sample Invoice Data

Invoices		
Invoice_Num	**Customer**	**Invoice_Date**
101	Hugo Furst	01Jan2004
102	Freida Peeples	15Jan2004
103	Al E. Loohah	30Jan2004

Sample Lineitem Data

LineItems			
Invoice_Num	**Line_Num**	**Item**	**Cost**
101	1	Widget	12.00
101	2	Gadget	10.00
101	3	Frammet	36.00

Plus additional rows for the other invoices.

In real life, there might be invoices without lineitems, and even vice-versa, but I'm not handling those situations.

What We Want To see

```
Invoice: 101 for Hugo Furst on 01JAN2004

       1  Widget              12.00

       2  Gadget              10.00

       3  Frammet             36.00

       4  Thingies            37.56


          Total:             $95.56
```

Master/Detail Reporting in SAS

- Although master/detail reports are common in both business and non-business life, Base SAS doesn't have a built-in way to handle them without preprocessing the data in some way.
- I think this is a serious deficiency. MS Access can do it. Crystal Reports can do it. Why can't SAS?

What This Paper Does

- This paper presents several ways of creating master/detail reports in the current version of SAS, 8.2.
- It also suggests some new ways to do reporting in version 9.1.

What This Paper Doesn't Do

- This paper doesn't talk much about appearance. The important thing is getting data on the page.
- For fiddling with appearance, use ODS. There are many papers and presentations on the topic at every SUGI.
- An additional new way to control appearance is ODS LAYOUT, which also is not discussed here.

OK, Onwards

I will briefly discuss five older methods:

- So-called “data_null_” reports, which use the data step language to write listing output.
- PROC REPORT
- PROC PRINT with #BYVAL
- Sending data _null_ output to ODS
- Data _null_ reports using data set indexes

New Methods

I will also discuss two new ways to create master/detail reports:

- PROC DOCUMENT
- The data step object-oriented interface to ODS

Zoom, Zoom, Zoom

I'm going to go through the older methods very quickly. Complete code is in my paper.

A good reference for these older reporting methods is the ***SAS Guide to Report Writing: Examples***.

Simple DATA _NULL_ Reporting

- The first step in traditional master/detail reporting in the data step is to join the two files together.
- Typically, this is done with a MERGE statement, which requires both data sets to be sorted.

Simple DATA _NULL_ Reporting

In my example, I will do the merge in a separate step and save the results for later use, but it could be done in the main data step.

```
data sugi29.invoices_and_items;
   merge sugi29.invoices
         sugi29.lineitems;
   by invoice_num;
run;
```

Simple DATA _NULL_ Reporting

Read in the data, using BY to set first. And last. variables, and write to the listing file.

```
data _null_;

   set sugi29.invoices_and_items;
   by invoice_num;

   file print;
```

Simple DATA _NULL_ Reporting

If this is the first record for an invoice, write the header and reset the total:

```
if first.invoice_num then
   do;
   if _n_ ne 1 then
      put _page_;
   put 'Invoice: ' invoice_num
       'for ' customer
       'on ' invoice_date;
   totalcost = 0;
   end;
```

Simple DATA _NULL_ Reporting

Next, write a line for each detail line and update the cost:

```
put @5 line_num  3.
    @10 item     $10.
    @22 cost     comma10.2;


totalcost + cost;
```

Simple DATA _NULL_ Reporting

After the last detail line for an invoice has been written, write out the total:

```
   if last.invoice_num then
      put /
          @10  'Total:'
          @22  totalcost dollar10.2;
run;
```

Simple DATA _NULL_ Reporting

And here's the first page of the results:

```
Invoice: 101 for Hugo Furst on 01JAN2004

       1  Widget                12.00
       2  Gadget                10.00
       3  Frammet               36.00
       4  Thingies              37.56


          Total:               $95.56
```

Simple DATA _NULL_ Reporting

There are two problems with traditional data _null_ reporting:

- It can be very complicated to write (and to understand later).
- It's designed for monospace, lineprinter output.

New Features in Versions 6 and 8

More recent versions of SAS have added useful new features:

- The Output Delivery System (ODS) added additional output destinations and the ability to use proportional, styled fonts.

- PROC REPORT and PROC TABULATE added procedural support for report.

Using #BYVAL to Customize Headers

A common use of data null was simply to add page titles whose text depended on data values.

#BYVAL provides a way to do that without data step code.

#BYVAL Code

```
options nobyline;
title "Invoice #BYVAL1 for #BYVAL2 on
  #BYVAL3";
proc print data=invoices_and_items
         noobs;
    by invoice_num customer
       invoice_date;
    id line_num;
    var item cost;
    pageby invoice_num;sum cost;
    sumby customer;
run;
```

#BYVAL Output

Invoice 101 for Hugo Furst on 01JAN2004

Line_Num	Item	Cost
1	Widget	12.00
2	Gadget	10.00
3	Frammet	36.00
4	Thingies	37.56
		95.56
Invoice_Num		95.56

PROC REPORT

- PROC REPORT is a very powerful reporting procedure, but it's not the topic of this paper. I will give an example, but I recommend looking at the documentation, and at SUGI papers by Lauren Haworth, Sandy McNeill, and Ray Pass.
- PROC TABULATE is often an alternative; see papers by the same suspects or by Dan Bruns.

PROC REPORT Code

```
proc report data=invoices_and_items
     nowindows;

    column invoice_num customer
           invoice_date line_num item cost;

    define invoice_num / order noprint;
    define customer / order noprint;
    define invoice_date / display noprint;
    define line_num / order;
    define cost / sum;
```

PROC REPORT Code

```
 compute before _page_;
      line @1 'Invoice: ' invoice_num 3.
              ' for ' customer $10.
              ' on ' invoice_date
                  worddatx12.;
   endcomp;

   break after customer
         / summarize page dol;

run;
```

PROC REPORT Code

Invoice: 101 for Hugo Furst on 01 Jan 2004

Line_Num	Item	Cost
1	Widget	12.00
2	Gadget	10.00
3	Frammet	36.00
4	Thingies	37.56
		==========
		95.56

PROC REPORT

- There are other ways I could have achieved the same effect.
- PROC REPORT automatically does any needed sorting and summarization; you don't have to write code to do it.
- PROC REPORT can use ODS to customize the appearance of your output.

Sending data _null_ output to ODS

- Without making any substantial changes, you can send data _null_ output to ODS and get some of the benefit of the Output Delivery System.
- I'm mentioning this, but I don't really recommend it – customization is difficult.

Sending data _null_ output to ODS

```
ods listing close;
ods pdf
    file='example4.pdf';

%include 'example1.sas';

ods pdf close;
ods listing;
```

Sending data _null_ output to ODS

Simple data _null_ report

```
Invoice:  101 for Hugo Furst on 1 Jan 2004
        1  Widget              12.00
        2  Gadget              10.00
        3  Frammet             36.00
        4  Thingies            37.56

           Total:             $95.56
```

It is, at least, possible to customize the titles and footnotes easily. The big disadvantage is that you get a fixed number of columns for the entire report.

Using Indexes In data _null_ Reports

- Recent versions of SAS allow you to use the KEY= option on the SET statement to directly read observations with a given value for a variable, provided that the data set is indexed on that variable.
- This eliminates the need for a MERGE to join the master and detail tables.
- Didn't work right before 6.12 (or so).

Using Indexes In data _null_ Reports

The LineItems data set was previously indexed:

```
proc datasets library=sugi29
                  nolist;
   modify LineItems;
       index create Invoice_Num;
run; quit;
```

Using Indexes In data _null_ Reports

```
data _null_;
   set sugi29.invoices;
   file print;
   if _n_ ne 1 then
      put _page_;
   put 'Invoice: ' invoice_num
       'for ' customer
       'on' invoice_date;
   totalcost = 0;
```

Using Indexes In data _null_ Reports

```
_iorc_ = 0;
do while (_iorc_ = 0);
   set sugi29.lineitems
       key=invoice_num;
   if _iorc_ = 0 then
      do;
      put @5 line_num  3.
          @10 item     $10.
          @22 cost     comma10.2;
      totalcost = totalcost + cost;
      end;
end;
```

Using Indexes In data _null_ Reports

```
    put @5 'Total'
        @22 totalcost comma10.2;
    _error_ = 0;

run;
```

Using Indexes In data _null_ Reports

```
Invoice: 101 for Hugo Furst on 1 Jan 2004
       1  Widget                12.00
       2  Gadget                10.00
       3  Frammet               36.00
       4  Thingies              37.56
    Total                       95.56
```

Using Indexes In data _null_ Reports

You might look at this and think "Wow, that's more complicated than the earlier version! Why bother?" You might be right for this example, but if you have more levels of detail – cities within counties within states, for example – it quickly becomes difficult to keep track of all the first. and last. variables needed by the previous method. This method lets you put all the code dealing with one data set in one place. Also, the top level doesn't need to be sorted.

New in V9 – PROC DOCUMENT

- Proc DOCUMENT is a base SAS procedure that lets you reorder the output from SAS procedure and the data step.
- PROC DOCUMENT works with document objects, which are created by the new ODS destination DOCUMENT.

New in V9 – PROC DOCUMENT

There are two steps to the process:

- Create a document object for each section of output
- Replay (i.e. print) the sections in the desired order, which is different from the original order.

Because PROC REPORT does not fully support ODS DOCUMENT, I will use PROC TABULATE to create the sections.

Creating Document Objects

```
ods document name=example7;
proc tabulate data=sugi29.invoices;
   by invoice_num;
   class invoice_num customer;
   var invoice_date;
   keylabel max=' ';
   table invoice_num * customer,
         invoice_date * max
        * format=worddatx12.;
run;
```

Creating Document Objects

```
proc tabulate
      data=sugi29.lineitems;
   by invoice_num;
   class line_num item;
   var cost;
   keylabel sum='';
   table (line_num * item) all,
         cost*sum;
run;

ods _all_ close;
```

Listing Document Objects

```
proc document name=example9;
    list / levels=all;
    run;
quit;
```

Document Objects Listing

```
Listing of: \Work.Example9\
Order by: Insertion
Number of levels: All

Obs Path                                          Type
1 \Tabulate#1                                     Dir
2 \Tabulate#1\ByGroup1#1                          Dir
3 \Tabulate#1\ByGroup1#1\Report#1                 Dir
4 \Tabulate#1\ByGroup1#1\Report#1\Table#1 Table
```

(slightly modified to fit on this slide)

Document Objects

- We'll have a lot of these document objects, one for each BY-group in each PROC TABULATE. By default, they're in the order in which they were created.
- We want to play them back with the objects for each invoice placed together.
- In this example, I'm doing it manually, but in practice you would do it with a program.

Replaying Document Objects

```
ods pdf
file="%sysfunc(pathname(sugi29))\..\examp
le9.pdf" notoc;

ods pdf startpage=never;

proc document name=example9;
  replay
\Tabulate#1\ByGroup1#1\Report#1\Table#1;
  replay
\Tabulate#2\ByGroup1#1\Report#1\Table#1;
run; quit;
```

Replaying Document Objects

		Invoice_Date
Invoice_Num	Customer	
101	Hugo Furst	1 Jan 2004

		Cost
		Sum
Line_Num	Item	
1	Widget	12.00
2	Gadget	10.00
3	Frammet	36.00
4	Thingies	37.56
All		95.56

The appearance needs some work, but you can see that the right data are being written in the right order.

Why Bother?

This technique will come in useful when you have dozens or hundreds of invoices.

- If you have 1000 invoices to create, it will be more efficient to run two PROC TABULATES than 2000 tabulates.

- You can intersperse graphs between tables and text.

The Object-Oriented Interface

- Version 9 supports objects in the data step. Rather than working on data per se, you work on objects, which have methods (similar to functions) and properties (similar to values).
- One of the object types is ODS.

The Object-Oriented Interface

Output is thought of as a series of objects.

- Each page contains a table object containing the invoice information followed by another table containing the lineitem information.
- Each table contains rows, each row contains cells, and each cell contains text.
- Each object can be formatted separately, programatically.

The OO Code

```
data _null_;
   declare odsout Example10();
   set sugi29.invoices end=end;
   Example10.table_start();
   Example10.row_start();
   Example10.format_cell (text: 'Invoice: '
       || put(invoice_num, 3.)
       || ' for ' || trim(customer)
       || ' on ' || left(put(invoice_date,
                             worddatx12.)));
   Example10.row_end();
   Example10.table_end();
```

The OO Code

```
totalcost = 0;
_iorc_ = 0;
Example10.table_start();
Example10.row_start();
Example10.format_cell(text: 'Line Num');
Example10.format_cell(text: 'Item');
Example10.format_cell(text: 'Cost');
Example10.row_end();
```

The OO Code

```
do while (_iorc_ = 0);
   set sugi29.lineitems key=invoice_num;
   if _iorc_ = 0 then
      do;
      Example10.row_start();
      Example10.format_cell(
                     text: put(line_num, 3.));
      Example10.format_cell( text: item );
      Example10.format_cell(
                     text: put(cost, comma10.2));
      Example10.row_end();
      totalcost = totalcost + cost;
      end;
end;
```

The OO Code

```
Example10.row_start();
Example10.format_cell(text: 'Total',
                      column_span: 2,
                      Overrides: "font_weight=bold");
Example10.format_cell(text:
                      put(totalcost, comma10.2)
                      Overrides: "font_weight=bold"));
Example10.row_end();
_error_ = 0;
Example10.table_end();
if not end then
   Example10.page();
run;
```

The OO Output

Invoice: 101 for Hugo Furst on 1 Jan 2004

Line Num	Item	Cost
1	Widget	12.00
2	Gadget	10.00
3	Frammet	36.00
4	Thingies	37.56
Total		**95.56**

Paper Availability

- Check the web page listed in the proceedings

  www.excursive.com/sas/

- It won't be available until a week or so after SUGI (I'm taking the train back to Sacramento).

- Send me email:

  jackhamilton@firsthealth.com

  (listed in the proceedings)

Questions?