SAS To EXCEL: An Integrated Reporting System

Fran Cohen, Westat, Inc., Rockville, MD

ABSTRACT

Often we need to produce reports from a set of data. Frequently the set of reports gets expanded upon or the programmer is asked to modify the reports already written. Instead of writing new SAS programs for each request, the programmer can use this approach to efficiently and more easily create camera-ready output. This paper describes an integrated approach to creating formatted Microsoft Excel reports from SAS data. A driver program builds all possible combinations of specified analysis variables for a set of reporting variables. Individual report programs read in pre-formatted Excel templates specifying the variables to be written on each report and write the Excel reports. Commands to open, close, write to an Excel file, and save as output Excel files are given in SAS via DDE. Since all the processing is done at the front end, only minor changes are needed to modify and add reports.

OVERVIEW

The first step in the system is to identify the class variables used to construct the reporting variables. To accomplish this, macro code is written to create summaries of the reporting variables. A program identifies the groupings used to increment the reporting variables by reading a file called 'levels', meaning class variable levels. Each call to the macro increments each reporting variable for each value of the class variable, creating the output records, which contain the values of the class variable along with the summarized values of the reporting variables. After the macro has been run for all the class variables specified, the output observations are concatenated into one file, which is used for all the reports.

In the second step, the Excel spreadsheet templates are manually formatted for each report. A template can contain a single or multiple sheets for each report.

In the third step, the individual SAS report programs are written to transfer the data from the summary file to the Excel spreadsheet. This program contains the code to read in the pre-formatted Excel templates specifying the variables to be written on each report and write the Excel reports. Commands to open, close, write to an Excel file and save as output Excel files are given in SAS via DDE.

Others have written about automated transfer of data from SAS to Excel. Mumma (1999) compared DDE, PROC IMPORT and other methods of transfer. Vyverman (2001, 2002) discussed in some detail DDE with regard to issuing specific commands given from SAS to Excel. The emphasis here is in creating one SAS database that will serve the purpose of creating many reports with minor changes to the main program. When a new report is requested with additional class variables, all that is required is to add a record to the levels file containing class variables and rerun the program. When a new report is requested with additional reporting variables, a line or two of code are added to the incrementing macro and the program is rerun. In each of these cases, a new template and SAS report are written, usually with only minor changes from the previous ones.

AN EXAMPLE

I created a fictitious data set about tennis players to illustrate the system. Our data set will contain two class variables: `agegrp` and `sex` of player. The reporting variables will be type of headgear worn to aid in the sun (`sunglasses` or `hats`) and type of gear worn to protect the (aging!) body (`ankle supports, elbow brace` or `knee brace`). Here is what the actual output data from step one looks like after all three programs discussed below have been run. This is a simplification of the original application, which involved computing weighted and unweighted counts and rates.

c_level	agegrp	sex	c_sungl	c_hats	c_ankle	c_elbow	c_kneeb
0	1		4	1	0	4	1
0	2		2	3	0	4	1
0	3		3	2	0	2	3
0	4		3	2	1	2	2
0	5		4	1	0	1	4
1	1	1	2	1	0	3	0
1	1	2	2	0	0	1	1
1	2	1	2	0	0	2	0
1	2	2	0	3	0	2	1
1	3	1	2	1	0	1	2
1	3	2	1	1	0	1	1
1	4	1	1	1	0	1	1
1	4	2	2	1	1	1	1
1	5	1	2	1	0	1	2
1	5	2	2	0	0	0	2

The first column c_level, contains an identifier of the class variables from the levels file. The next two columns, `agegrp` and `sex`, are the class variables, and they are followed by the five reporting variables: `c_sungl`, `c_hats`, `c_ankle`, `c_elbow`, and `c_kneeb`. The counts represented here are the incremented counts for each value of the class variables: the number of people who use sunglasses or hats, ankle supports, elbow braces or knee braces when they play tennis. Let's start with a simple example of creating a report of "Types of Body Protection by Age Group of Tennis Player".

We need to increment our counts within the class variable agegrp. Here is the levels file as it appears initially.

```
/*class variables*/
/*The file tells the SAS program which
variable groupings to use to increment
reporting variables. It contains parameters
for BY-variable processing levels.*/

/* The four parameters of
varnames,varfreqs,firstdotvar,level number
are passed to the macro that does the
incrementation. In this case the first
agegrp is passed to varnames, the second
agegrp is passed to varfreqs, the third
agegrp is passed to firstdotvar and the
identifier 0 is passed to level number.*/

agegrp,agegrp,agegrp,0,
/*age group of players*/
```

Each reporting variable in the report is to be incremented within each agegrp. We have five categories of age group: <20, 21-30,31-40,41-50 and 51+.

We might want the Excel table for Report 1 to look like this:

Age Group			
	Ankle Supports	**Elbow Brace**	**Knee Brace**
<20	0	4	1
21-30	0	4	1
31-40	0	2	3
41-50	1	2	2
51+	0	1	4

How are we going to tell SAS which variables from which observations in the SAS data set to write?

Vyverman (2001, 2002) has discussed formatting of Excel worksheet cells using Microsoft Version 4 XML, or Macro Language commands. This is another approach, but it is difficult to find documentation of these commands. I thought that perhaps I could use the VBA code to capture commands, but as Vyverman (2001, 2002) notes this does not work. An easier approach is for the programmer to use Excel to manually format a table and save it for later use. Once this Excel file is created, it is read each time the reports are run.

The formatted template, Report1.xls, looks like this:

Age Group			
	Ankle Supports	**Elbow Brace**	**Knee Brace**
_age	_c_ankle	_c_elbow	_c_kneeb

Note that in addition to the row headings, there are some variables beginning with underscores. These identify the actual variables that are required by this particular report. A master array is created of all the variables in the system that could appear in any report, and the variables specified by the template identify which of those in particular, should be written for a particular report. Since all the variables in the array must be of the same type, the numeric variables are converted to character by the driver program and that is why in these examples they are preceded by a 'c' in the templates and in the output file shown above. A PROC FORMAT is used to assign a format to each of the reporting variables, and this format is used to identify the variable in the template. Here is the actual code from the log of the report:

```
proc format;
value $master "c_sungl" = 1 "c_hats" = 2
"c_ankle" = 3 "c_elbow" = 4 "c_kneeb" = 5
"age" = 6 ;
```

Each of the reporting variables is identified in the format statement. For example, the variable c_ankle has a value of 3 in the value statement of the PROC FORMAT. Now let's look at some code in the Report 1 program.

Before writing to Excel, the SAS program issues commands via DDE to open Excel. Since I created a new report for each day of the week, I deleted the previous week's report before creating the new one.

```
%excelopen(&shellpath,&outpath,&report,&day
ofweek);

options noxwait noxsync;
x "f:\weswin2\dllshare\$exc97.exe";
run;
data _null_;
x = sleep(5);
run;
filename commands dde 'excel|system';
run;
x "erase
c:\tennisexample\output\report1\report1_mon
.xls";
run;
```

```
data _null_;
file commands;
put
"[open(""c:\tennisexample\report1.xls"")]";
run;
```

Now I subset the data to the records of interest for this particular report.

```
data masterfile;
set
report.master_report(where=(left(c_level)="
0"));

/*Notice that '0' identifies the records
reporting at the agegrp level.*/

%let numcolumns = 4;
/*how many columns will be filled*/

%let indataarea = r4c1:r4c4;
/*what to read from the input excel
template spreadsheet*/

%let outdataarea = r4c1:r8c4;
/* what area to fill in on the excel
template spreadsheet*/
```

The ContinuousFill macro contains the code to read in the formatted template, identify the variables to write to the report, matches and extracts them from the masterfile. DDE commands will then be issued to save the output file and close Excel via DDE.

Some key parts of the log illustrating the reading of the template and writing to it are reprinted below with some annotations.

```
%continuousfill(&report,&sheet,&numcolumns,
&indataarea,&outdataarea);

filename table dde "excel|[report1.xls]all
players!r4c1:r4c4" notab;
run;

data datashell;
informat column1 column2 column3 column4
$20.;

infile table dlm='09'x dsd missover;
/*This reads in the template for report 1.*/
input column1 column2 column3 column4 ;
run;

data datatable;
set masterfile(in=in1);

if in1 then set datashell point=nobs
nobs=nobs;

array mas c_sungl c_hats c_ankle c_elbow
c_kneeb age;

array columns column1 column2 column3
column4;

do i = 1 to 4;
if substr(columns(i),1,1) = '_' then do;
tempvar=upcase(substr(columns(i),2,19));
columns(i) = mas(put(tempvar,$master.));
end;
end;

drop i;
run;
```

This step illustrates how the template variables are linked to the proper variables in the SAS dataset, referred to as the masterfile. A temporary variable is created from the column name in the template, after stripping the leading underscore. For instance, when tempvar is equal to `c_ankle` , the column associated with column 2 is equal to the 3rd item in the master array, which happens to be equal to `c_ankle`. When we look at the dataset we see that the first five rows pertain to Report 1. For values of 1, 2, 3, and 5 of the `agegrp` variable, no one uses ankle supports. But one person in `agegrp` 4 uses ankle supports. These counts are shown in the Excel spreadsheet for Report 1.

```
filename table dde "excel|[report1.xls]all
players!r4c1:r8c4" notab;
run;

data _null_ ;
retain tab '09'x;

set datatable;
file table;

put column1 tab column2 tab column3 tab
column4 tab ;

/*The Excel file is written.*/
run;
```

Now we save the new report. Note that the template is preserved, and a new file is saved each time the program is run.

```
%excelclose(&outpath,&report,&dayofweek);

options noxwait noxsync;
filename commands dde 'excel|system';
run;

data _null_;
file commands;
put
"[save.as(""c:\tennisexample\output\report1
\report1_mon.xls"")]";
put '[quit]';
run;
```

Suppose we want to show Report 1 by sex as well as age group. This can easily be accomplished. All we have to do is add another instruction in the levels file to aggregate the reporting variables by the `agegrp` (5) by `sex` (2) combinations.

Here is the levels file as it appears with the new record.

```
agegrp,agegrp,agegrp,0,
/*age group of players*/
agegrp sex, agegrp*sex,sex,1,
/*age group by sex*/
```

In order to create a template for Report 2, the `Report1.xls` template file is opened; another tab is added; the one for 'All Players' is renamed so we now have a sheet for 'Male' and a sheet for 'Female'. The file is saved as a new template, `Report2.xls`.

The Report 1 program only needs some minor modifications to create this new report: the file is now subset to the level 1 cases and the `sex` is specified for each sheet. The only changes needed are shown in bold italics. The program is saved as Report 2.

```
data masterfile;
set
report.master_report(where=(left(c_level)=
"1" and sex="&sexval"));

/*Notice that we are now subsetting to the
level 1 records for a particular sex.*/

%let numcolumns = 4;
%let indataarea = r4c1:r4c4;
%let outdataarea = r4c1:r8c4;

%continuousfill(&report,&sheet,&numcolumns,
&indataarea,&outdataarea);
```

The output for Report 2 follows:

Types of Body Protection by Age Group and Sex of Tennis Player: **Male**

Age Group	Ankle Supports	Elbow Brace	Knee Brace
<20	0	3	0
21-30	0	2	0
31-40	0	1	2
41-50	0	1	1
51+	0	1	2

Types of Body Protection by Age Group and Sex of Tennis Player: **Female**

Age Group	Ankle Supports	Elbow Brace	Knee Brace
<20	0	1	1
21-30	0	2	1
31-40	0	1	1
41-50	1	1	1
51+	0	0	2

Now suppose we decide we need to see all of the reporting variables by `agegrp` and `sex`. Let's create Report 3. No additional levels are needed. All reporting variables have been created by the driver program at the front end, so all we need to do is modify the template from Report 2. We create a template for Report 3 with the added columns for the newly added reporting variables. The Report 3 program can be used with only a modification to the number of columns read and the number of columns written.

```
%let numcolumns = 6;
%let indataarea = r4c1:r4c6;
%let outdataarea = r4c1:r8c6;
```

The output for Report 3 follows:

What Tennis Players Use to Protect Themselves from Sun and Wear and Tear by Age Group and Sex: **Male**

Age Group	Hat	Sunglasses	Ankle Supports	Elbow Brace	Knee Brace
<20	1	2	0	3	0
21-30	0	2	0	2	0
31-40	1	2	0	1	2
41-50	1	1	0	1	1
51+	1	2	0	1	2

What Tennis Players Use to Protect Themselves from Sun and Wear and Tear by Age Group and Sex: **Female**

Age Group	Hat	Sunglasses	Ankle Supports	Elbow Brace	Knee Brace
<20	0	2	0	1	1
21-30	3	0	0	2	1
31-40	1	1	0	1	1
41-50	1	2	1	1	1
51+	0	2	0	0	2

In conclusion, a little planning at the front end goes a long way in efficiency when multiple Excel report files are desired.

Due to space limitations, I haven't included all the code in this paper, but will be glad to e-mail it to you.

DISCLAIMER: The contents of this paper are the work of the author and do not necessarily represent the opinions, recommendations, or practices of Westat.

ACKNOWLEDGEMENTS

The author would like to thank Brice Hart, Sharon Hirabayshi, and Ian and Marianne Whitlock for their insights and ideas that contributed to this paper.

CONTACT INFORMATION

Fran Cohen, Westat, Inc.
1650 Research Boulevard
Rockville, MD 20850
Email: cohenf1@westat.com

REFERENCES

Mumma, M.T. *The Redmond to Cary Express- A Comparison of Methods to Automate Data Transfer Between SAS and Microsoft Excel.* Proceedings of the 12th Annual Northeast SAS Users Group Conference, 1999.

Vyverman, K. *Using Dynamic Data Exchange to Export Your SAS Data to MS Excel-Against all ODS, Part 1.* Proceedings of the 26th Annual SAS Users Group International Conference, 2001.

Vyverman, K. *Creating Custom Excel Workbooks from Base SAS with Dynamic Data Exchange: A Complete Walkthrough.* Proceedings of the 27th Annual SAS Users Group International Conference, 2002.

TRADEMARK NOTICE