Paper 135-2009

ODS 101+

Erik W. Tilanus, Driebergen, the Netherlands

ABSTRACT

The Output Delivery System (ODS) forms an universal layer between SAS® DATA steps and procedures and the output they produce. This facilitates flexible formatting of outputs for a wide variety of output destinations: printers, printer-like files (PDF, postscript), screen (HTML) and so on. A special category is the output destination, which stores procedure output in SAS data sets, also for those procedures that do not have an output option.

This paper discusses the basic ODS statements and shows examples of the output produced. It also hints to the more advanced options for tailoring the output to specific needs.

INTRODUCTION

In the classic IT world, you ran a program and the results were sent to a printer. The standard printer was able to print 132 characters per line and 60 or 66 lines per page. One font, capitals only.

Today is different. If you still want the result on paper, no problem. The average printer today has a wide variety of fonts available in different sizes, so you can tailor how you want to see it. But if you want to inspect your results on screen or via internet, the lay-out as prepared for a printer may not be the most suitable.

This is where ODS, the Output Delivery System comes in. ODS can be considered as a layer between the program result and the output medium. You can compare it to a driver: you tell it what to do and ODS takes care of the how to do it. ODS can tailor your results for a wide range of output formats: it still can handle the legacy printer, but it also creates output in HTML or XML format for web publication; RTF-formatted documents to include in MS-Office applications; PDF-documents, and many more.

For each output element that a procedure may produce, like a title, footnote, column header, statistic and hundreds of other elements, ODS has default lay-out descriptions in so-called templates. These templates define fonts, styles, possibly colors et cetera for each possible output destination. So just telling ODS where to send the output, and in what format, is sufficient to get your results. But the real fun is that you are not limited to the destinations and styles that SAS provides you. With PROC TEMPLATE you can change the ODS templates anyway you want: change fonts, colors, layout of procedure output and so on. In short, you can modify your output to give it a personal touch or make it fit into corporate style specifications.

ODS-THE BASICS

If you run your SAS program, all your output is sent to the output window (when running in the Display Manager), to a text file (in batch processing) or directly to a printer. In general terms: the output is sent to SASList. You may not notice, but even this is handled by ODS. The default destination in ODS is the LISTING destination, which happens to be the “classic” SASList. So if you do not include any ODS directives in your program, it runs the same way as it has been running for decades!

But let us now assume that you want to send your output to an HTML file. You will have to tell SAS where to send it and to format it as HTML. Probably you don't need the listing output anymore so you can close that destination. You can do this with the following statements:

```
ODS LISTING CLOSE;
ODS HTML FILE='path and filename.html';
```

Next you execute the program steps that produce the required output and finally you close the HTML file and return to the standard listing destination (or specify any other destination).

```
ODS HTML CLOSE;
ODS LISTING;
```

If you just close the listing destination with `ODS LISTING CLOSE`; and do not specify another destination, your created output is going nowhere and is lost.

Similarly, you can specify any other supported destination, so for instance ODS RTF FILE=…; would send the output to an RTF file.

DATA FOR THE EXAMPLES

Our examples will be based on a data set with meteorological data[1]. It contains daily minimum and maximum temperatures, duration of sunshine and millimeters of precipitation for the period of 1 January 2001 to 31 July 2008. To get aquainted with the data, the first 20 observations are printed with PROC PRINT (Table 1). TempMin and TempMax are in degrees Celsius, SunHours is formatted as HHMM and Precipitation is measured in millimeters.

[1] Source: Royal Netherlands Meteorological Institute (KNMI)

Table 1: First 20 observations of the meteorological data (PROC PRINT)

```
The SAS System                                         17:00 Tuesday, September 16, 2008   1

                                                                Sun
 Obs      Date             TempMin          TempMax      Hours    Precipitation

   1     01JAN01                 1              6.3      0:00               9.5
   2     02JAN01               5.6             11.2      2:42               0.3
   3     03JAN01               4.5              9.9      5:12                 4
   4     04JAN01               5.8              9.1      0:00               4.9
   5     05JAN01               6.2             10.7      0:00              18.9
   6     06JAN01               4.1              7.6      1:54               2.1
   7     07JAN01               3.6              6.8      0:00               0.3
   8     08JAN01               1.3              5.5      1:30               1.7
   9     09JAN01               1.1              5.5      2:42                 0
  10     10JAN01               0.5              4.1      1:36                 0
  11     11JAN01              -1.8                4      6:36                 0
  12     12JAN01              -3.7              5.2      1:36             0.025
  13     13JAN01              -3.2              3.1      7:00                 0
  14     14JAN01              -5.3              0.7      0:00                 0
  15     15JAN01              -4.8              1.5      7:12                 0
  16     16JAN01              -6.4              1.9      7:12                 0
  17     17JAN01                -9                1      6:42                 0
  18     18JAN01              -8.8             -0.5      0:00               0.4
  19     19JAN01              -1.3              0.3      0:12             0.025
  20     20JAN01                -1              1.3      0:00               1.3
```

EXAMPLE 1:
Now let us create the same output in HTML format, RTF format and PDF format. We can do this all in the same run. Program 1 shows how to do this. Each pair of ODS statements (e.g. ODS HTML FILE…; - ODS HTML CLOSE;) controls sending to the specified destination.

Program 1: Sending output to various destinations

```
   ODS HTML FILE="e:\SASForum\2009\Example1.HTML";
   ODS RTF FILE="e:\SASForum\2009\Example1.RTF";
   ODS PDF FILE="e:\SASForum\2009\Example1.PDF";
   ODS LISTING CLOSE;
   PROC PRINT DATA=KNMI(OBS=20);
   RUN;
   ODS HTML CLOSE;
   ODS RTF CLOSE;
   ODS PDF CLOSE;
   ODS LISTING;
```

Output 1(A-C) shows the resulting output. SAS provides internal Result Viewers, that enable you to view the results within the SAS Windowing environment, but I have chosen to make screen shots of the output when displayed in generic applications for these file formats (Firefox, MS Word and Acrobat Reader).

OUTPUT DESTINATIONS

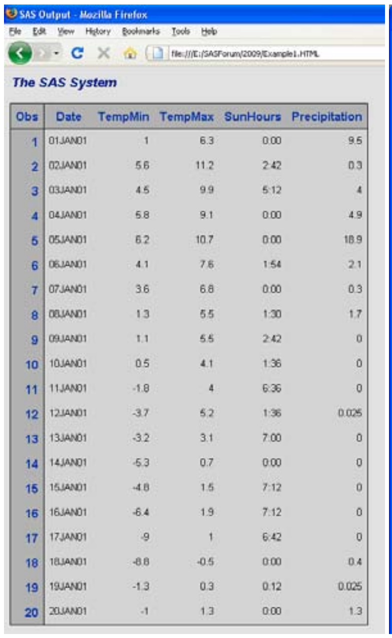
In Example 1 you have been introduced to four of the common ODS destinations: LISTING, HTML, RTF and PDF. There are many more possible destinations. SAS has grouped them into two basic groups: SAS Formatted destinations and Third Party formatted destinations. The Third Party Formatted group is further divided into the markup language destinations, the printer destinations and two other destinations. Taking a somewhat closer look provides the following list of destinations:

1. **Document:** This is in fact an internal SAS destination. It stores the internal structure of the output. Once it is stored, the document can be used to send the output to any other destination, without the need to rerun the program. Compare it to the replay facilities in SAS/GRAPH.
2. **Listing:** The default destination. SASList. Has been there since SAS release 1 (1972)!
3. **Output:** This destination creates a SAS data set that contains the output. This data set can be used for further processing of the procedure output. Compare it to PROC PRINTTO, by which you can send output to a file. Next you can read that file back in and process it further.
4. **HTML:** The first of the third party destinations. In its original form it is creating HTML3.2 formatted files. But today it is part of the Markup family of destinations (next) and creates HTML4 output.
5. **Markup:** This is the collection of all destinations that work with tagged formatting. The standard tagsets that SAS supplies include CHTML and other flavors of HTML, CSV, DOCBOOK, a new RTF destination (TAGSETS.RTF), XML and more. You can also define your own tagset and thus create your own destinations.

6. **PS and PCL:** These destinations create print files for respectively postscript and PCL (HP) printers.
7. **PDF:** This one is trivial: it creates PDF files to be browsed or printed using Adobe Acrobat Reader or a similar program.
8. **RTF:** This destination creates RTF (Rich Text Format) documents which can be used in MS Office. There is a new version, based on the markup family: TAGSETS.RTF. The difference is that the standard RTF destination leaves pagination to MS Office, while in TAGSETS.RTF you can control exactly where on a page you want to put your output.

More destinations may be added over time. Most of them will use the markup family as a basis. In this paper we will focus on the HTML destination and for comparison show similar results in RTF and PDF format. But once you are familiar with these destinations, it should not be hard to work with the other destinations as well. However, you might encounter options that are available in one destination and not in another, because it does not make sense there.

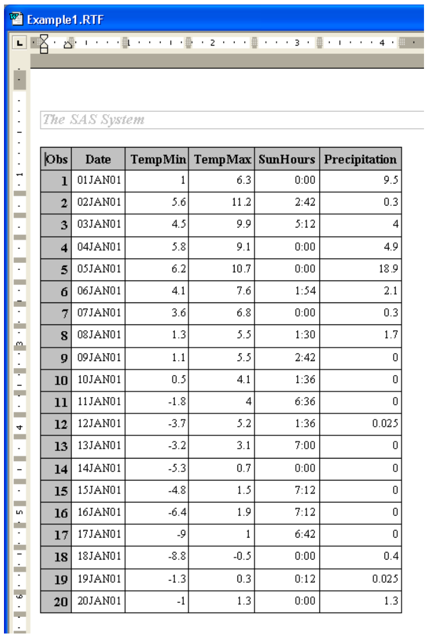
Output 1: Same PROC PRINT output, different destinations


SAS Output - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example1.HTML

The SAS System

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	01JAN01	1	6.3	0:00	9.5
2	02JAN01	5.6	11.2	2:42	0.3
3	03JAN01	4.5	9.9	5:12	4
4	04JAN01	5.8	9.1	0:00	4.9
5	05JAN01	6.2	10.7	0:00	18.9
6	06JAN01	4.1	7.6	1:54	2.1
7	07JAN01	3.6	6.8	0:00	0.3
8	08JAN01	1.3	5.5	1:30	1.7
9	09JAN01	1.1	5.5	2:42	0
10	10JAN01	0.5	4.1	1:36	0
11	11JAN01	-1.8	4	6:36	0
12	12JAN01	-3.7	5.2	1:36	0.025
13	13JAN01	-3.2	3.1	7:00	0
14	14JAN01	-5.3	0.7	0:00	0
15	15JAN01	-4.8	1.5	7:12	0
16	16JAN01	-6.4	1.9	7:12	0
17	17JAN01	-9	1	6:42	0
18	18JAN01	-8.8	-0.5	0:00	0.4
19	19JAN01	-1.3	0.3	0:12	0.025
20	20JAN01	-1	1.3	0:00	1.3

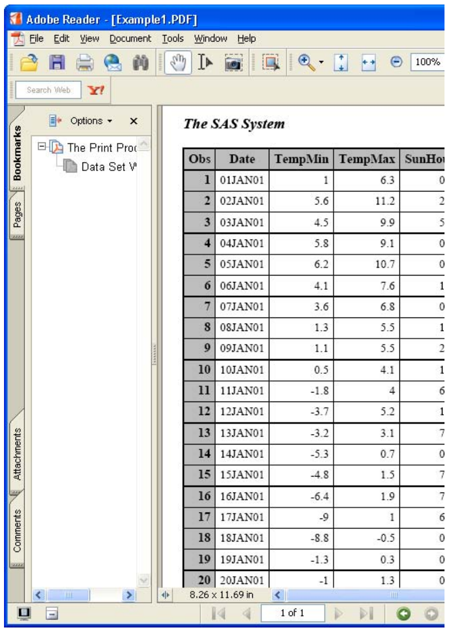
A: HTML



The SAS System

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	01JAN01	1	6.3	0:00	9.5
2	02JAN01	5.6	11.2	2:42	0.3
3	03JAN01	4.5	9.9	5:12	4
4	04JAN01	5.8	9.1	0:00	4.9
5	05JAN01	6.2	10.7	0:00	18.9
6	06JAN01	4.1	7.6	1:54	2.1
7	07JAN01	3.6	6.8	0:00	0.3
8	08JAN01	1.3	5.5	1:30	1.7
9	09JAN01	1.1	5.5	2:42	0
10	10JAN01	0.5	4.1	1:36	0
11	11JAN01	-1.8	4	6:36	0
12	12JAN01	-3.7	5.2	1:36	0.025
13	13JAN01	-3.2	3.1	7:00	0
14	14JAN01	-5.3	0.7	0:00	0
15	15JAN01	-4.8	1.5	7:12	0
16	16JAN01	-6.4	1.9	7:12	0
17	17JAN01	-9	1	6:42	0
18	18JAN01	-8.8	-0.5	0:00	0.4
19	19JAN01	-1.3	0.3	0:12	0.025
20	20JAN01	-1	1.3	0:00	1.3

B: RTF



The SAS System

Obs	Date	TempMin	TempMax	SunHo
1	01JAN01	1	6.3	0
2	02JAN01	5.6	11.2	2
3	03JAN01	4.5	9.9	5
4	04JAN01	5.8	9.1	0
5	05JAN01	6.2	10.7	0
6	06JAN01	4.1	7.6	1
7	07JAN01	3.6	6.8	0
8	08JAN01	1.3	5.5	1
9	09JAN01	1.1	5.5	2
10	10JAN01	0.5	4.1	1
11	11JAN01	-1.8	4	6
12	12JAN01	-3.7	5.2	1
13	13JAN01	-3.2	3.1	7
14	14JAN01	-5.3	0.7	0
15	15JAN01	-4.8	1.5	7
16	16JAN01	-6.4	1.9	7
17	17JAN01	-9	1	6
18	18JAN01	-8.8	-0.5	0
19	19JAN01	-1.3	0.3	0
20	20JAN01	-1	1.3	0

C: PDF

SIMPLE MODIFICATIONS TO THE FORMATTING

ODS is ruled by templates. A template is a collection of instructions on how the output should be presented. It includes table layout descriptions for all table-based procedure output and formatting instructions for all elements in the output: foreground and background colors, font face, style and size, titles, footers, borders and so on. You can create you own templates, but also use the SAS templates as a basis and just modify a few elements in them.

Let us modify the output as presented in Output 1a and apply the following changes:

1. Color the title red
2. Make the observation numbers smaller and not bold
3. Change the font in the table to Times New Roman, make it somewhat bigger and color the text green
4. Change the background of the table into yellow

The tool to change layouts and styles is PROC TEMPLATE. Program 2 shows the required code.

Program 2: PROC TEMPLATE coding to modify some style elements

```
proc template;
define style Forumstyle;                                          ①
   parent=styles.default;                                         ②
   style systemtitle from systemtitle /                           ③
      foreground=red;
   style rowheader from rowheader /                               ④
      font_size=2 font_weight=medium;
   style data from data /                                         ⑤
      font_face="times new roman" font_size=5 foreground=green
      background=yellow ;                                         ⑥
end;
run;
```

In this case we are only changing the style of some elements in the output. So we are going to define a new style: Forumstyle ➀. There is no reason to define a style from scratch. By defining a parent style ➁ you can include all the existing definitions in the parent. Styles.default is one of the standard styles that SAS includes and it is the default for the HTML destination. If you don't include the parent specification and you are not creating a complete template an error will be reported when trying to use the template, since essential elements are missing.

Next we change the style element SystemTitle ➂. Again, we use an existing style element as the basis. This is done with the "from" option. In this case the basis is simply the SystemTitle style in styles.default, so "from systemtitle". It does not need to be the same style element. If there is another style element that comes closer to what you want, you can use that as a basis. So

```
   style systemtitle from systemfooter /...;
```

is completely acceptable. It specifies that SystemTitle will include all style attributes from SystemFooter and then apply the changes. The change of the text color is simply specified by telling that the foreground should be red.

Next we change style RowHeader, the style for the observation numbers ➃ and style Data ➄, which specifies how the data cells in the output should look like. The applied changes are self-explanatory by comparing the statements with the change objectives. Finally we close the define block with an END statement ➅ and run the procedure.

One problem remains: how do we know what the style elements in the output are and what style attributes can be set in any style? The answer is simple: read the documentation. The style elements are listed in appendix 4 in the ODS User's Guide (both in ODS UG for SAS 9.1.3 and SAS 9.2) and the attributes are listed in table 9.1 (ODS UG for SAS 9.1.3) or table 10.3 (ODS UG for SAS 9.2) and explained in the subsequent pages.

Another approach is to look in the style definitions that are included in SASHELP.TMPLMST. How to do that is discussed in the next paragraph.

WHERE ARE TEMPLATES STORED?

Templates are stored in *template stores*, structures that are more or less comparable to catalogs, although they are not the same. The templates that SAS provides are stored in SASHELP.TMPLMST. If you make your own templates and do not specify where to store them, they are stored in SASUSER.TEMPLAT. If nothing else is specified these are the two locations where ODS is looking for templates. To specify where ODS should look for the templates, you can use the ODS PATH statement. In this statement you specify the template stores and the sequence in which they should be searched. You also specify whether the template store is only available to retrieve templates (read-only) or that PROC TEMPLATE may store new or modified templates in it.

The default ODS PATH statement is:

```
   ODS PATH SASUSER.TEMPLAT(UPDATE) SASHELP.TMPLMST(READ);
```

While experimenting with PROC TEMPLATE and styles you may want to use the WORK library for your templates, by specifying:

```
   ODS PATH WORK.TEMPLAT(UPDATE) SASHELP.TMPLMST(READ);
```

If you want to know which styles are available and where they are located, go to the Results window in the SAS Windowing Environment. Right click on Results and select Templates. A short cut for the path through the results window is to issue the command ODSTEMPLATES on the command line. Either way, it will result in a tree-view from which you can select a template that you like. Double clicking it will show its definition. Unfortunately it does not show an example of the resulting output. You have to interpret the PROC TEMPLATE code or run a little test program to see the effect.

Another method to retrieve the style information is by using the SOURCE statement in PROC TEMPLATE, e.g.

```
   Proc template;
      Source Forumstyle;
   Run;
```

will write the source of the Forumstyle style to the SAS Log.

EXAMPLE 2 (HTML)

Let us put all the pieces together and run the program. Program 2b shows the complete source. It starts with the ODS PATH statement that specifies that my own templates should be stored in WORK.TEMPLAT. Then follows the PROC TEMPLATE code as in Program 2. The ODS HTML statement is now augmented with the style option. It specifies that new style Forumstyle is to be used. And finally there is the PROC PRINT.

The result of this program is presented in Output 2.

Program 2b: Creating PROC PRINT output with modified styles

```
   ods path work.templat(update) sashelp.tmplmst(read);
   proc template;
   define style Forumstyle;
      parent=styles.default;
      style systemtitle from systemtitle /
```

```
         foreground=red;
      style rowheader from rowheader /
         font_size=2 font_weight=medium;
      style data from data /
         font_face="times new roman" font_size=5 foreground=green background=yellow ;
   end;
   run;

   ODS HTML FILE="e:\SASForum\2009\Example2.HTML" style=Forumstyle;
   ODS LISTING CLOSE;
   proc print data=knmi(obs=20) ;
   run;
   ODS HTML CLOSE;
   ODS LISTING;
```

EXAMPLE 3 (RTF)

You could do the same for the RTF destination. However you should be aware that the default style for RTF is not styles.default, but styles.rtf. Compare Output 1b with Output 1a. You will notice the differences. Not only grayscale versus color for the background, but also the use of the Times New Roman as font rather than Arial.

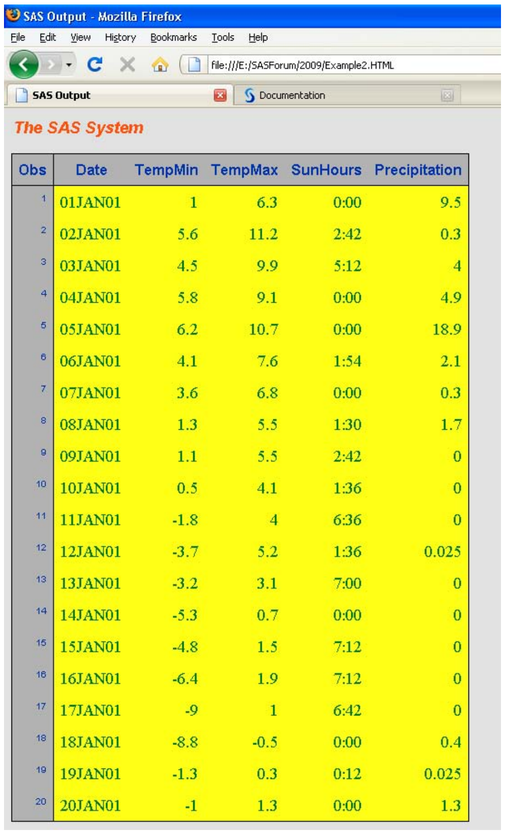
Of course you can use exactly the same PROC TEMPLATE code as in Program 2, which will result in a colored RTF document resembling the HTML output, but an alternative is to use styles.rtf as the parent.

Personally I prefer to use styles.rtf. And in my documents I don't like the gray background and I prefer Arial as the default font in tables. So I use MyRTFstyle, as defined in Program 3 and I can directly copy and paste my results in any document as I have done in Output 3.

Program 3: My preferred style for RTF files

```
   proc template;
   define style MyRTFStyle;
      parent=styles.rtf;
      style data from data /
         font_face="arial";
      style rowheader from rowheader /
         background=white font_face="arial";
      style header from header /
         background=white font_face="arial";
   end;
   run;
```

Output 2: The modified print resulting from Program 2b


SAS Output - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example2.HTML
SAS Output
Documentation

The SAS System

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	01JAN01	1	6.3	0:00	9.5
2	02JAN01	5.6	11.2	2:42	0.3
3	03JAN01	4.5	9.9	5:12	4
4	04JAN01	5.8	9.1	0:00	4.9
5	05JAN01	6.2	10.7	0:00	18.9
6	06JAN01	4.1	7.6	1:54	2.1
7	07JAN01	3.6	6.8	0:00	0.3
8	08JAN01	1.3	5.5	1:30	1.7
9	09JAN01	1.1	5.5	2:42	0
10	10JAN01	0.5	4.1	1:36	0
11	11JAN01	-1.8	4	6:36	0
12	12JAN01	-3.7	5.2	1:36	0.025
13	13JAN01	-3.2	3.1	7:00	0
14	14JAN01	-5.3	0.7	0:00	0
15	15JAN01	-4.8	1.5	7:12	0
16	16JAN01	-6.4	1.9	7:12	0
17	17JAN01	-9	1	6:42	0
18	18JAN01	-8.8	-0.5	0:00	0.4
19	19JAN01	-1.3	0.3	0:12	0.025
20	20JAN01	-1	1.3	0:00	1.3

Output 3: RTF output based on MyRTFStyle copied and pasted into this document (reduced font size for reproduction)

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	01JAN01	1	6.3	0:00	9.5
2	02JAN01	5.6	11.2	2:42	0.3
3	03JAN01	4.5	9.9	5:12	4
4	04JAN01	5.8	9.1	0:00	4.9
5	05JAN01	6.2	10.7	0:00	18.9
6	06JAN01	4.1	7.6	1:54	2.1
7	07JAN01	3.6	6.8	0:00	0.3
8	08JAN01	1.3	5.5	1:30	1.7
9	09JAN01	1.1	5.5	2:42	0
10	10JAN01	0.5	4.1	1:36	0
11	11JAN01	-1.8	4	6:36	0
12	12JAN01	-3.7	5.2	1:36	0.025
13	13JAN01	-3.2	3.1	7:00	0
14	14JAN01	-5.3	0.7	0:00	0
15	15JAN01	-4.8	1.5	7:12	0
16	16JAN01	-6.4	1.9	7:12	0
17	17JAN01	-9	1	6:42	0
18	18JAN01	-8.8	-0.5	0:00	0.4
19	19JAN01	-1.3	0.3	0:12	0.025
20	20JAN01	-1	1.3	0:00	1.3

DATA STEP OUTPUT (FILE PRINT, PUT)

PROC PRINT is of course not the only way to produce reports. Other well-known procedures are PROC TABULATE and PROC REPORT. But there exist also thousands of tailor-made reports, programmed within a DATA step, using FILE PRINT and PUT statements. Can you use ODS in conjunction with FILE PRINT and PUT? The answer is simply: Yes, you can, in fact you do already! Since even if you don't specify anything, ODS still plays it's role in the background.

EXAMPLE 4: SIMPLE DATA STEP OUTPUT IN HTML

Some years ago I wrote a simple program to demonstrate the flexibility of the combination of FILE PRINT and PUT statements, using the N=PS option in the FILE statement. The DATA step in Program 4 is this program. It steps simply through the horizontal positions 1 to 80 and then calculates the vertical position to put an asterisk on the sine curve. Next it also puts a hyphen as the X-axis. Running this data step (without the surrounding ODS statements) would provide the output as in Output 4.

Including the ODS statements the output will be created as an HTML file as in Output 4a.

Program 4: Using FILE PRINT and PUT statements to create a tailor-made report

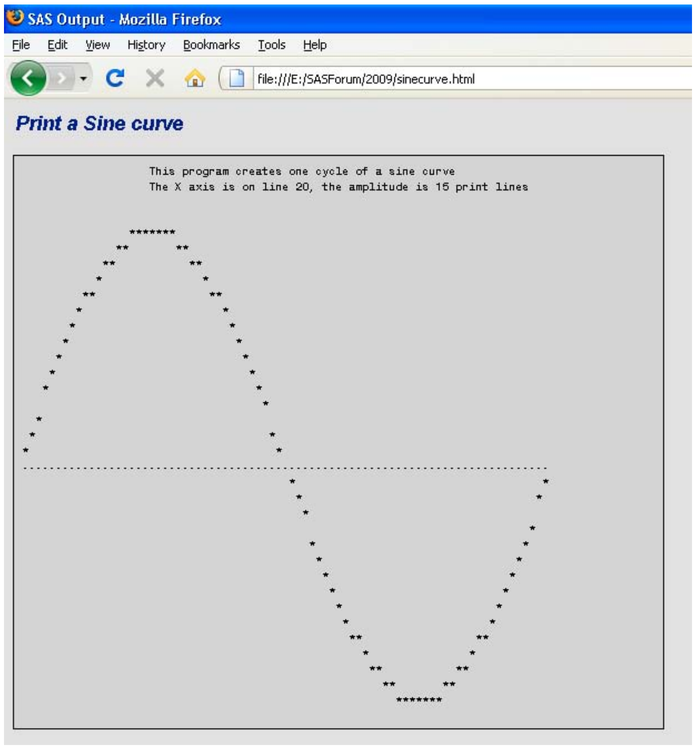
```
ODS LISTING CLOSE;
ODS HTML FILE='e:\sasforum\2009\sinecurve.html';
TITLE 'Print a Sine curve';
DATA _NULL_;
   FILE PRINT N=PS HEADER=MyHeader;
   HorizontalPosition=1;
   DO UNTIL (HorizontalPosition=80);
      VerticalPosition=ROUND(20-
                             15*SIN(2*3.14*HorizontalPosition/80));
      PUT #VerticalPosition @HorizontalPosition '*';
      PUT #20 @HorizontalPosition '-';         ;
      HorizontalPosition+1;
   END;
STOP;
RETURN;
MyHeader:
   PUT #1 @20 "This program creates one cycle of a sine curve";
   PUT #2 @20 "The X axis is on line 20, "
              "the amplitude is 15 print lines";
RETURN;
RUN;
ODS HTML CLOSE;
ODS LISTING;
```

Output 4: Simple Listing result of Program 4, without the ODS statements (reduced font)

```
Print a Sine curve                                         09:18 Friday, September 19, 2008   6
                   This program creates one cycle of a sine curve
                   The X axis is on line 20, the amplitude is 15 print lines


                *******
              **       **
            **           **
           *               *
         **                 **
        *                     *
       *                       *
      *                         *
     *                           *
    *                             *
   *                               *
                                    *
  *
 *                                   *
*                                     *
--------------------------------------------------------------------------------
                                        *                                     *
                                         *                                   *
                                          *
                                                                            *
                                           *                               *
                                            *                             *
                                             *                           *
                                              *                         *
                                               *                       *
                                                *                     *
                                                 **                 **
                                                   *               *
                                                    **           **
                                                      **       **
                                                        *******
```

Output 4a: Same output as in Output 4, including the ODS statements


SAS Output - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/sinecurve.html

Print a Sine curve

```
                   This program creates one cycle of a sine curve
                   The X axis is on line 20, the amplitude is 15 print lines


                *******
              **       **
            **           **
           *               *
         **                 **
        *                     *
       *                       *
      *                         *
     *                           *
    *                             *
   *                               *
                                    *
  *
 *                                   *
*                                     *
--------------------------------------------------------------------------------
                                        *                                     *
                                         *                                   *
                                          *
                                                                            *
                                           *                               *
                                            *                             *
                                             *                           *
                                              *                         *
                                               *                       *
                                                *                     *
                                                 **                 **
                                                   *               *
                                                    **           **
                                                      **       **
                                                        *******
```

ODS INFORMATION WITHIN THE DATA STEP

You can also make it known to the DATA step that you want to actively use ODS. You do this by specifying the ODS= option in the FILE PRINT statement. In this option you can specify columns to be printed and styles to be used. Since the print positions are now determined by the style and column definitions, they don't have to be specified in the PUT statement. PUT _ODS_; simply writes all variables in their assigned positions! For more information on this you are referred to the ODS User's Guide.

MODIFYING PROCEDURE OUTPUT, INCLUDE AND EXCLUDE

The examples so far were dealing with PROC PRINT output. For other procedures the output often consists of more than one part. For instance let us use PROC CORR to find the correlation between hours of sunshine and precipitation. It makes sense to assume that there is a negative correlation!

EXAMPLE 5: SIMPLE PROCEDURE OUTPUT IN HTML

In this example we will run the program without any specials (Program 5). The only statements to notice are the ODS TRACE statements. These will be discussed below. The resulting output is presented in Output 5.

As you can see the output consists of three boxes, output objects in ODS terminology: first a listing of the variables, then a table presenting some basic statistics similar to what PROC MEANS would do and finally the correlation information.

ODS allows you to exclude any of these objects and using PROC TEMPLATE you can modify the appearance of each object individually. But for that purpose you should be able to identify each object. That is the purpose of the ODS TRACE statements (Program 5). When turned on, the SAS Log will contain information about each of them as you can see Listing 1.

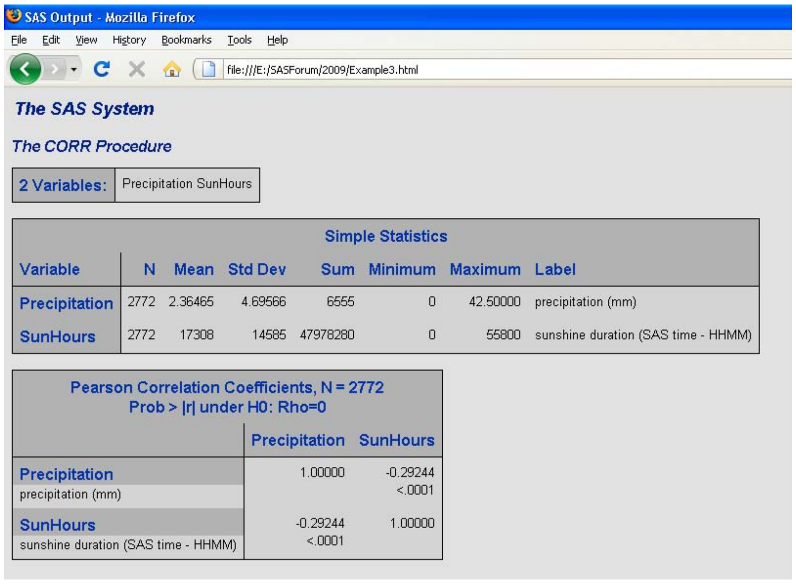
Program 5: Creating standard HTML output with PROC CORR

```
   ODS html FILE="e:\SASForum\2009\Example5.html";
   ods listing close;
   ods trace on;

   proc corr data=knmi;
   var precipitation sunhours;
   run;

   ods html close;
   ods trace off;
   ods listing;
```

Output 5: The standard HTML output from PROC CORR


SAS Output - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example3.html

The SAS System

The CORR Procedure

2 Variables:	Precipitation SunHours

Simple Statistics							
Variable	N	Mean	Std Dev	Sum	Minimum	Maximum	Label
Precipitation	2772	2.36465	4.69566	6555	0	42.50000	precipitation (mm)
SunHours	2772	17308	14585	47978280	0	55800	sunshine duration (SAS time - HHMM)

Pearson Correlation Coefficients, N = 2772 Prob > |r| under H0: Rho=0	Precipitation	SunHours
Precipitation precipitation (mm)	1.00000	-0.29244 <.0001
SunHours sunshine duration (SAS time - HHMM)	-0.29244 <.0001	1.00000

Listing 1: SAS Log of Program 5, showing the ODS TRACE output

```
1098   ODS html FILE="e:\SASForum\2009\Example5.html";
NOTE: Writing HTML Body file: e:\SASForum\2009\Example5.html
1099   ods listing close;
1100   ods trace on;
1101   proc corr data=knmi;
1102   var precipitation sunhours;
```

```
1103  run;

Output Added:
-------------
Name:       VarInformation
Label:      Variables Information
Template:   base.corr.VarInfo
Path:       Corr.VarInformation
-------------

Output Added:
-------------
Name:       SimpleStats
Label:      Simple Statistics
Template:   base.corr.UniStat
Path:       Corr.SimpleStats
-------------

Output Added:
-------------
Name:       PearsonCorr
Label:      Pearson Correlations
Template:   base.corr.StackedMatrix
Path:       Corr.PearsonCorr
-------------
NOTE: PROCEDURE CORR used (Total process time):
      real time           0.19 seconds
      cpu time            0.05 seconds

1104
1105  ods html close;
1106  ods trace off;
1107  ods listing;
```

EXCLUDING ONE OR MORE OUTPUT OBJECTS

From Listing 1 you can see that the name of the object containing the basic statistics is SimpleStats. Assume that you don't want it, you can specify:

```
ODS EXCLUDE SimpleStats;
```

This statement should precede the PROC CORR statement. Similarly there is also an ODS SELECT statement, which defines the opposite: it will only include the specified object(s).

BASIC CHANGES TO PROCEDURE OUTPUT

If you would like to modify certain style elements for all tables, you can do it just as in the earlier examples. For instance the following code will color the row headers in all tables red and color all data cells green.

```
proc template;
define style MyHTML;
parent = styles.default;
style rowheader from rowheader / foreground =red;
style data from data / foreground = green;
end;
run;
ODS html FILE="e:\SASForum\2009\Example6.html"  style=MyHTML;
ods listing close;
proc corr data=knmi;
var precipitation sunhours;
run;
ods html close;
ods listing;
```

CHANGING INDIVIDUAL OUTPUT OBJECTS OF PROC CORR

It is also possible to modify each table (object) in the output separately. In that case you have to modify the table definitions that ODS uses for PROC CORR. The starting point is the trace information. It tells you the names of the table templates that are used. Then you can access the table definitions from the template store to find what has been defined and then modify what you want. The easiest way is to copy and paste the table definition into the program window. Then you have all information together and you can delete the elements that you don't need.

For instance this is the definition of the VarInformation table of PROC CORR in SASHELP.TMPLMST:

```
proc template;
   define table Base.Corr.VarInfo;
      notes "Variable Information";
      dynamic VarWidth;
      column NVars Variables VarNames;
      translate _val_=0 into "";
      define NVars;
         space = 1;
         format = 4.0;
         style = RowHeader;
         id;
         merge;
      end;
      define Variables;
         space = 4;
         style = RowHeader;
         id;
      end;
      define VarNames;
         width_max = VarWidth;
         width = 1;
         flow;
         maximize;
      end;
   end;
run;
```

If we want to change some aspects of this definition, we can edit it, by replacing the DEFINE statement with the EDIT statement and then add the changes to the column definitions that we want and delete all the rest (except of course the END statements). So it could look like this:

```
proc template;
   edit Base.Corr.VarInfo;
      define NVars;
         style = varHeader;
      end;
      define Variables;
         style = varHeader;
      end;
      define VarNames;
         style = varList;
      end;
   end;
run;
```

By comparing the original and the modifications you see that we changed the style information, defining our own styles (which by the way are not yet defined at this point). The reason to change from the original RowHeader style to an own style (varHeader) is that RowHeader is present in all tables. So if I change that style, it is changed in all tables. By defining my own style, I create a style that is unique for this table!

EXAMPLE 6: MODIFIED PROCEDURE OUTPUT

With this as background information it is now possible to modify the output of PROC CORR. What we want to achieve is this:

1. Exclude the basic statistics
2. In the VarInformation table, change the background of the row header into pink and the font of the variable listing into Times new Roman, italic, with a yellow background.
3. In the title of the PearsonCorr table change the background into black and foreground into white
4. The row header of the PearsonCorr table consists in fact of two definitions: one row for the variable name and one for the label. Change the background for the variable name into yellow. The background of the variable labels is turned into yellow, the font into courier and it should be small print.
5. Finally the data in the cells should be in red, while the probability information should be in green.

Program 6 provides the full listing of the PROC TEMPLATE code to accomplish this (except for the exclude, which is simply managed by including an ODS EXCLUDE statement). The first block ① edits the VarInfo table as explained above. The second block edits ② the StackedMatrix table, which is the basis for the PearsonCorr table (see Listing 1). In its definition you find PARENT information ③, specifying from where this definition inherits. The original definition referred to Common.StackedMatrix. Since this is also used by other procedures, we don't want to change that. In stead we refer to Common.StackedMatrixET. Next we create Common.StackedMatrixET by editing Common.StackedMatrix and saving it as Common.StackedMatrixET ④. As you can see new style names are given in each definition. So next is to define all these new styles ⑤. This part should by now be self-explanatory.

Program 6: The PROC TEMPLATE code for the modified PROC CORR output

```
proc template;
edit Base.Corr.VarInfo as base.corr.varinfo;                          ①
   define NVars;
      style = varHeader;
   end;
   define Variables;
      style = varHeader;
   end;
   define VarNames;
      style = varlist;
   end;
end;

edit Base.Corr.StackedMatrix as base.corr.StackedMatrix;              ②
   define RowName;
      varname = Variable;
      parent = Common.Column.RowName;
      style = MyRowName;
   end;
   define RowLabel;
      varname = Label;
      parent = Common.Column.RowLabel;
      style = MyRowLabel ;
   end;
   parent = Common.StackedMatrixET;                                    ③
   split_stack = OFF;
end;

edit Common.StackedMatrix as Common.StackedMatrixET;                  ④
   define head;
      parent = common.header.matrix;
      style = MyHeaderStyle;
   end;
   define matrix;
      parent = common.column.matrix;
      style = MyMatrix1;
   end;
   define matrix2;
      parent = common.column.matrix;
      style = MyMatrix2;
   end;
end;

define style ForumCorr;                                               ⑤
   parent=styles.default;
   style varheader from rowheader/
      background = pink  font_weight=light;
   style varlist from rowheader /
      background = yellow font_face='times new roman' font_style=italic;
   style myRowName from rowheader /
      background = yellow;
   style myRowLabel from rowheader /
      background = white font_size=2 font_face='courier';
   style MyHeaderStyle /
      background=black foreground=white;
   style mymatrix1 from data/
      foreground = red;
   style mymatrix2 from data/
      foreground = green;
end;
run;
```

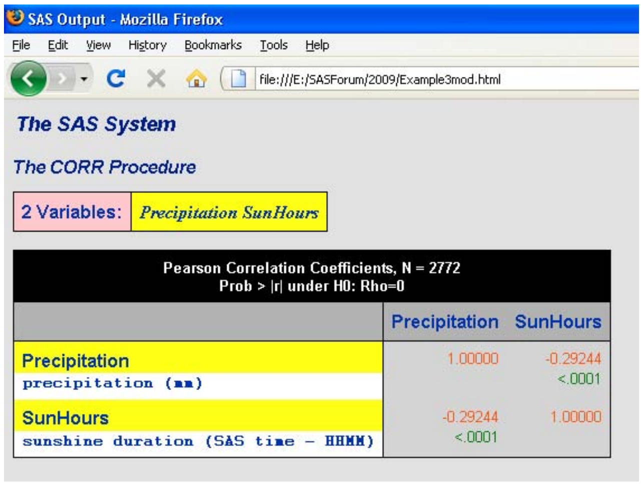
Once the template definition is ready (Program 6) it is rather straightforward to create the modified output::

```
ODS html FILE="e:\SASForum\2009\Example6.html" style=ForumCorr;
ods listing close;
ods exclude SimpleStats;
proc corr data=knmi;
var precipitation sunhours;
run;
ods html close;
ods listing;
```

The result is presented in Output 6.

Be aware of the fact that we modified the table definitions of the tables that PROC CORR uses in Program 6. These modified definitions will apply not only to the current PROC CORR, but to all PROC CORR output until you remove the modified definitions from the ODS search path as defined in the ODS PATH statement (ref. Page 4). This removal can be done in two ways: leave the definitions where they are and change the ODS PATH statement or delete the modified tables from the template store.

Output 6: the modified PROC CORR output


SAS Output - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example3mod.html

The SAS System

The CORR Procedure

2 Variables:	*Precipitation SunHours*

Pearson Correlation Coefficients, N = 2772 Prob > |r| under H0: Rho=0		
	Precipitation	SunHours
Precipitation precipitation (mm)	1.00000	-0.29244 <.0001
SunHours sunshine duration (SAS time - HHMM)	-0.29244 <.0001	1.00000

TABLE OF CONTENTS

The examples so far contained information that fits on one page. But if you have more output, either created by various procedures or just simply long output from one procedure it might be handy to have a table of contents that points exactly to the right point in the output. ODS HTML can create such table of contents on the fly, you only have to specify where it should be filed, e.g.:

```
   ODS HTML FILE="e:\SASForum\2009\bodyfile.HTML"
            CONTENTS="e:\SASForum\2009\contentsfile.HTML";
```

But why not display the table of contents and the output next to each other on the screen? No problem! You just should add a frame file:

```
   ODS HTML FILE="e:\SASForum\2009\bodyfile.HTML"
            CONTENTS="e:\SASForum\2009\contentsfile.HTML"
            FRAME="e:\SASForum\2009\framefile.HTML";
```

BROWSER AND ADDRESSING ISSUES

By including a contents file and a frame file we are going to deal with three files simultaneously. Of course you can specify the addresses as shown above, however it is more attractive to leave the path out and just use the file name. That makes it easier when you move the files to a web server.

For that purpose there is the PATH option in the ODS HTML statement. There you specify the path and in the other options just the file name.

Another issue is a browser (in)compatibility. The above ODS HTML statements work fine when the frame file is opened in MS Internet Explorer, but Firefox reports an error. It indicates that it does not know how to open the file. It requires that the filename be preceded by “file:///”. That can be achieved by adding a BASE option, which includes the part that will be included in the links.

Throwing all these bits and pieces together, we come to this statement:

```
   ODS HTML FILE="bodyfile.HTML"
            CONTENTS="contentsfile.HTML"
            FRAME="framefile.HTML"
            PATH="e:\SASForum\2009\" BASE="file:///";
```

EXAMPLE 7: TABLE OF CONTENTS

To make it sensible there must be something to include in the table of contents. When creating procedure output consisting of more parts (like in PROC CORR), the table of contents will include references to each part of the output. In case of BY-group processing, there will be a link to each BY group.

We are going to use the latter. Remember that our meteorological file contains daily observations for several years, in total some 2770 observations. Let us add a second date variable, which we call Month and format with MONYY. and use that in a BY statement. Program 7 puts everything together and Output 7 shows the result.

Program 7: Generate a table of contents, based on BY values

```
data KNMI;
set KNMI;
Month = date;
Format month monyy7.;
run;

ODS HTML FILE="Example7.HTML"
         CONTENTS="Example7C.HTML"
         FRAME="Example7F.HTML"
         PATH="e:\SASForum\2009\" BASE="FILE:///";
ODS LISTING CLOSE;
proc print data=knmi;
by month;
run;
ODS HTML CLOSE;
ODS LISTING;
```

EXAMPLE 8: MODIFYING THE TABLE OF CONTENTS

Creating a table of contents is fine, but the way it looks may not be very pleasing for your users. For us it may be relevant information that the output is created with PROC PRINT and that the data set is WORK.KNMI. But for a user you probably would like a more descriptive title and just the month as a clickable table of contents item rather than the description “Data Set WORK.KNMI”.

These wishes are easy to honor. First the title. This is accomplished by including an ODS PROCLABEL statement before the start of the PROC PRINT, e.g.:

```
ODS PROCLABEL "Daily sunshine and precipitation data";
```

Output 7: Adding a table of contents, just clicked at JUL2002

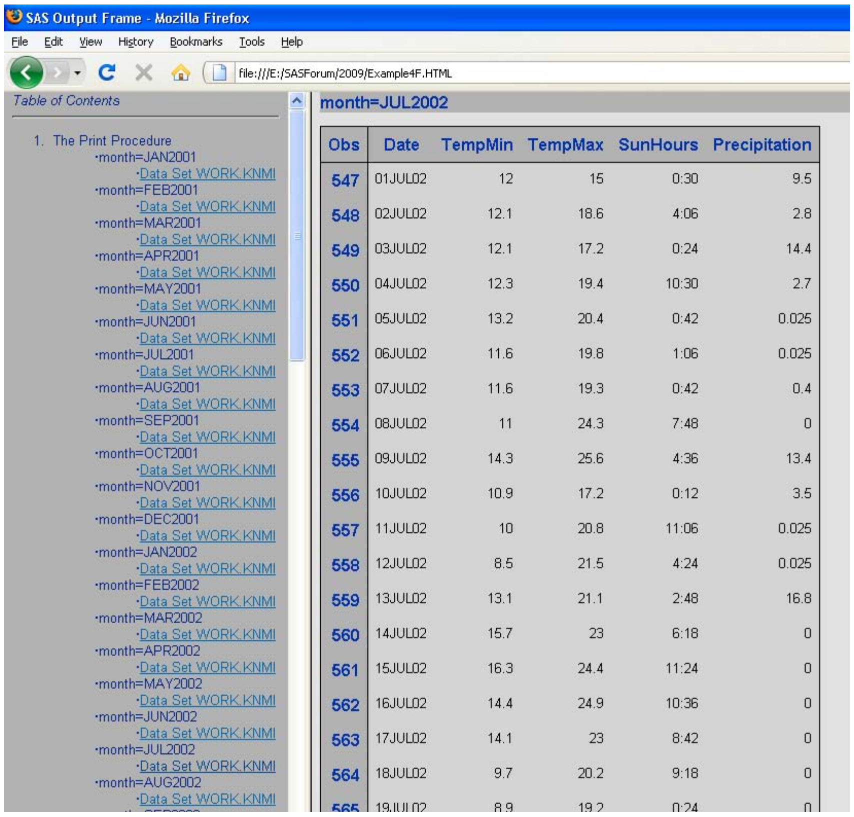

SAS Output Frame - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example4F.HTML

Table of Contents

1. The Print Procedure
- month=JAN2001
  - Data Set WORK.KNMI
- month=FEB2001
  - Data Set WORK.KNMI
- month=MAR2001
  - Data Set WORK.KNMI
- month=APR2001
  - Data Set WORK.KNMI
- month=MAY2001
  - Data Set WORK.KNMI
- month=JUN2001
  - Data Set WORK.KNMI
- month=JUL2001
  - Data Set WORK.KNMI
- month=AUG2001
  - Data Set WORK.KNMI
- month=SEP2001
  - Data Set WORK.KNMI
- month=OCT2001
  - Data Set WORK.KNMI
- month=NOV2001
  - Data Set WORK.KNMI
- month=DEC2001
  - Data Set WORK.KNMI
- month=JAN2002
  - Data Set WORK.KNMI
- month=FEB2002
  - Data Set WORK.KNMI
- month=MAR2002
  - Data Set WORK.KNMI
- month=APR2002
  - Data Set WORK.KNMI
- month=MAY2002
  - Data Set WORK.KNMI
- month=JUN2002
  - Data Set WORK.KNMI
- month=JUL2002
  - Data Set WORK.KNMI
- month=AUG2002
  - Data Set WORK.KNMI

month=JUL2002

Obs	Date	TempMin	TempMax	SunHours	Precipitation
547	01JUL02	12	15	0:30	9.5
548	02JUL02	12.1	18.6	4:06	2.8
549	03JUL02	12.1	17.2	0:24	14.4
550	04JUL02	12.3	19.4	10:30	2.7
551	05JUL02	13.2	20.4	0:42	0.025
552	06JUL02	11.6	19.8	1:06	0.025
553	07JUL02	11.6	19.3	0:42	0.4
554	08JUL02	11	24.3	7:48	0
555	09JUL02	14.3	25.6	4:36	13.4
556	10JUL02	10.9	17.2	0:12	3.5
557	11JUL02	10	20.8	11:06	0.025
558	12JUL02	8.5	21.5	4:24	0.025
559	13JUL02	13.1	21.1	2:48	16.8
560	14JUL02	15.7	23	6:18	0
561	15JUL02	16.3	24.4	11:24	0
562	16JUL02	14.4	24.9	10:36	0
563	17JUL02	14.1	23	8:42	0
564	18JUL02	9.7	20.2	9:18	0
565	19JUL02	8.9	19.2	0:24	0

Secondly the clickable month in the table of contents. This is somewhat more complicated and requires again some PROC TEMPLATE coding, using a number of other style attributes than we have seen so far. In fact it is beyond the basics, but included for those who are curious.

The required PROC TEMPLATE coding is presented in Program 8. Appendix 4 in the ODS User’s Guide provides the names of all the style elements that together form the Table of Contents. So to makes things easy we start with copying all these styles. A closer look shows that not all of them are needed. So they are deleted again. Then starts the process of modifying. We will go through it item by item. We are going to call the new table of contents style TOCworks ①.

In the style IndexItem there is a line prehtml = html('prehtml flyover bullet') ②. PREHTML allows you to insert own HTML coding before the ODS output is generated. The contents of what should be included are defined elsewhere. Leaving out “bullet” removes the bullet before each contents item.

In the style ContentFolder we are not changing anything, but since it depends on IndexItem we include it here to make sure that it inherits from the right IndexItem style. In SAS 9.2 this should not be necessary anymore due to a change in the inheritance rules.

The original ByContentFolder simply copies ContentFolder. We add listentryanchor=on to it, which makes the by-line information in the table of contents clickable.

Also the ContentItem style is in its original form just a copy of IndexItem. It is responsible for the text “Data Set WORK.KNMI”. We turn listentryanchor off, since we don’t want that to be clickable and next we make it vanish with the PREHTML and POSTHTML attributes: we change it into an HTML comment! The additional <BR> inserts an extra carriage control, effectively make the table of contents double spaced.

Once we got this all in place, the modified table of contents is shown in Output 8.

Of course you can add more style modifications to change e.g. the background color or font information. But that has been shown before. So no need to complicates things here further.

Program 8: Modifying the Table of Contents

```
proc template;
define style TOCworks;
   parent=styles.default;                                              ①
   style IndexItem from Container
      "Abstract. Controls list items and folders for Contents and Pages."  /
      leftmargin = 6pt
      posthtml = html('posthtml flyover')
      prehtml = html('prehtml flyover' )                               ②
      listentryanchor = on
      bullet = NONE
      background = _undef_
      foreground = colors('conentryfg');
   style ContentFolder from IndexItem                                  ③
      "Controls the generic folder definition in the Contents file." /
      listentryanchor = off
      foreground = colors('confolderfg');
   style ByContentFolder from ContentFolder                            ④
      "Controls the byline folder in the Contents file." /
      listentryanchor = on ;
   style ContentItem from IndexItem                                    ⑤
      "Controls the leafnode item in the Contents file."  /
      listentryanchor = off
      prehtml="<!-"
      posthtml="-><br>";
   style ContentProcName from IndexProcName
      "Controls the proc name in the Contents file."/
      bullet=none;
end;
run;

ODS HTML FILE="Example8.HTML"
         CONTENTS="Example8C.HTML"
         FRAME="Example8F.HTML"
         PATH="e:\SASForum\2009\" BASE="FILE:///" style=TOCworks;
ODS LISTING CLOSE;
ODS PROCLABEL "Daily sunshine and precipitation data";
proc print data=knmi;
by month;
run;
ODS HTML CLOSE;
ODS LISTING;
```

USING STYLES IN PROC PRINT, TABULATE AND REPORT

Even if you don’t want to play with PROC TEMPLATE you can still take advantage from some of its aspects. In PROC PRINT, PROC TABULATE and PROC REPORT you can specify style attributes to columns or fields directly in the PROC step.

The principle is rather straightforward. In any statement where you define variables to be used in the report and in the PROC statement, you can add a style specification. Its general form is:

```
style(area)=[style-attributes]
```

Output 8: Table of contents after modification

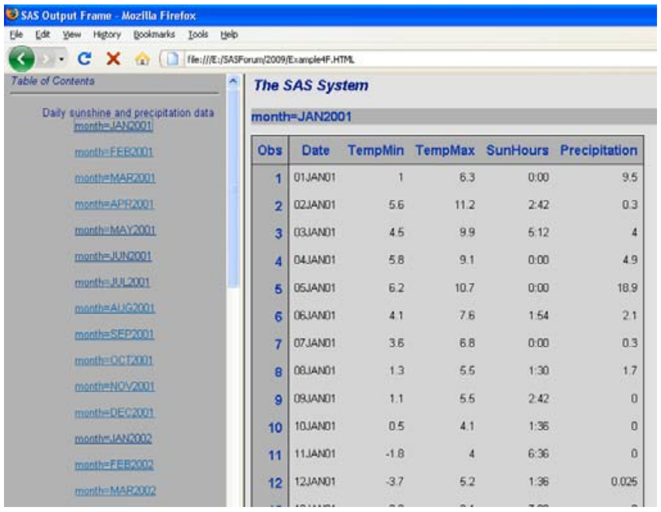

SAS Output Frame - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///E:/SASForum/2009/Example4F.HTML

Table of Contents

Daily sunshine and precipitation data
month=JAN2001
month=FEB2001
month=MAR2001
month=APR2001
month=MAY2001
month=JUN2001
month=JUL2001
month=AUG2001
month=SEP2001
month=OCT2001
month=NOV2001
month=DEC2001
month=JAN2002
month=FEB2002
month=MAR2002

The SAS System

month=JAN2001

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	01JAN01	1	6.3	0:00	9.5
2	02JAN01	5.6	11.2	2:42	0.3
3	03JAN01	4.5	9.9	5:12	4
4	04JAN01	5.8	9.1	0:00	4.9
5	05JAN01	6.2	10.7	0:00	18.9
6	06JAN01	4.1	7.6	1:54	2.1
7	07JAN01	3.6	6.8	0:00	0.3
8	08JAN01	1.3	5.5	1:30	1.7
9	09JAN01	1.1	5.5	2:42	0
10	10JAN01	0.5	4.1	1:36	0
11	11JAN01	-1.8	4	6:36	0
12	12JAN01	-3.7	5.2	1:36	0.025

For instance in PROC PRINT you can specify something like:

```
   var date/style(data)=[font_face="arial" font_style=italic font_size=5]
            style(header)=[background=black foreground=white];
```

This statement defines that the data in the column of the variable Date should be printed in arial, italic with a rather big font size (note: if you want to specify the font size in points, put "pt" after the size, font_size=5pt is of course a very small font) and that the column header should be white on black.

Table 2 shows the areas of the procedure output that you can modify. But be aware, while you should specify the area when working with PROC PRINT and PROC REPORT, you should leave it out in PROC TABULATE. The defined style is derived from where you define it. So the table regarding PROC TABULATE should be interpreted as an indication where you could specify the styles for the given part of the report.

An interesting option when specifying styles in these report procedures is "traffic lighting", changing the foreground or background color of cells based on their value. For this purpose you create a custom format, with color definitions as its labels. Next you specify this format on the foreground or background style attribute.

Table 2: Style elements to be used in reporting procedures

PROC PRINT		
Table		
Obsheader	header	header
obs	data/ column	data/ column
obs	data/ column	data/ column
obs	data/ column	data/ column
Bylabel	total	total
Grandtotal	grandtotal	grandtotal

PROC REPORT		
report		
header	header	header
column	column	column
column	column	column
summary	summary	summary
lines		
lines		

PROC TABULATE			
tabulate			
box		var	var
		keyword	keyword
class	class		
	classlev	proc	proc
classlev	classlev	proc	proc
	keyword	all	all

EXAMPLE 9: STYLES IN PROC PRINT

To demonstrate the use of styles in PROC PRINT we use the same data as in the earlier examples. This time we use the RTF output destination, to make it easy to copy and paste the result into this paper.

Program 9 shows the example. The PROC FORMAT defines a color code for the temperature variables. Next follows the PROC PRINT, where we use separate VAR statements for the columns to enable different style options to be used in each of the columns.

For the Date column we overrule the default RTF font, size and style. The two temperature columns (Tempmin and Tempmax) are printed in Courier bold. Here we apply the traffic lighting by specifying the Tempcolor format in the foreground. The Sunhours column receives a bold font and a yellow background and finally the Precipitation column gets a white on black header. Output 9 shows the result.

Program 9: Using styles in PROC PRINT

```
proc format;
value tempcolor
low-0='blue'
0-10='green'
10-20='yellow'
20-high='red';
run;

ods rtf body='e:\sasforum\2009\rtfprint.rtf';
proc print data=knmi(obs=20);
var date/style(data)=[font_face="arial" font_style=italic font_size=14pt];
var tempmin tempmax/style(data)=[font_face="courier" font_weight=bold
                                 font_size=5 foreground=tempcolor.];
var sunhours/style(data)=[ font_weight=bold background = yellow] ;
var precipitation/style(header)=[background=black foreground=white];
run;
ods rtf close;
```

Output 9: PROC PRINT using styles in the PROC step (partial, reduced font size)

Obs	Date	TempMin	TempMax	SunHours	Precipitation
1	*01JAN01*	1	6.3	0:00	9.5
2	*02JAN01*	5.6	11.2	2:42	0.3
3	*03JAN01*	4.5	9.9	5:12	4
4	*04JAN01*	5.8	9.1	0:00	4.9
5	*05JAN01*	6.2	10.7	0:00	18.9
6	*06JAN01*	4.1	7.6	1:54	2.1
7	*07JAN01*	3.6	6.8	0:00	0.3
8	*08JAN01*	1.3	5.5	1:30	1.7
9	*09JAN01*	1.1	5.5	2:42	0
10	*10JAN01*	0.5	4.1	1:36	0
11	*11JAN01*	-1.8	4	6:36	0
12	*12JAN01*	-3.7	5.2	1:36	-0.1
...	...	...	...	...	...

EXAMPLE 10: STYLES IN PROC TABULATE

As mentioned applying the styles in PROC TABULATE is a little different. You should not include the style element you are modifying. The statement where you define the style determines which style element is modified. Program 10 shows an example. We use the same input data but we make a report with the mean minimum and maximum temperature per month. For traffic lighting we use the same format as defined in Program 9.

In the CLASS statement the style for the class-variable header is defined and in the CLASSLEV statement the formatting of the class levels on the rows. The style in the VAR statement determines the font in the header lines of the columns and finally the styles in the table statement define the presentation of the data in the cells. Output 10 presents a part of the produced table. Compare the definitions in Program 10 with the output and the relationship should be clear.

Program 10: Using styles in PROC TABULATE

```
ods rtf body='e:\sasforum\2009\rtftabulate.rtf';
proc tabulate data=knmi;
class date/style=[font_weight=bold background=yellow];
classlev date/style=[font_face="courier" font_size=12pt background=white];
var tempmin tempmax/style=[font_face="arial"];
table date,
      (tempmin*{style=[font_style=italic font_weight=bold font_size=12pt
          foreground=tempcolor.]}
       tempmax*{style=[font_weight=bold font_size=12pt
          background=tempcolor.]})*mean;
format date monyy.;
run;
ods rtf close;
```

Output 10: PROC TABULATE output using styles (reduced font size)

	Min. temperature, (degrees celcius)	Max. temperature, (degrees celcius)
Date	**Mean**	**Mean**
JAN01	0.03	5.33
FEB01	1.15	7.79
MAR01	1.84	8.03
APR01	4.09	12.58
MAY01	8.52	19.36
JUN01	9.71	20.05
JUL01	13.28	23.61
AUG01	12.95	23.64
SEP01	10.09	17.36
OCT01	10.84	17.96
NOV01	3.46	10.24
...	...	...

THE OUTPUT DESTINATION

The previous chapters discussed the various output formatting options. Now it is time to turn to a completely different output option: the output destination. The output destination creates SAS data sets from output produced by procedures.

We already discussed the concept of output objects: units of information produced by procedures (ref. Listing 1). It are these output objects that can be stored in a data set, but you have to be aware of some fundamental differences with the other destinations.

By default the output destination does not output anything unless you specify it, in contrast to the other destinations that include everything unless you exclude it. The output destination is active only for one procedure, rather than till an ODS CLOSE statement, unless you instruct differently.

To specify the output objects to be written to a data set, you need to know their name. To find these names you can use the ODS TRACE ON statement. Be aware that there must be at least one open destination for ODS TRACE to work. So you better leave the listing destination open to begin with.

In Listing 1 you can see the output objects produced by PROC CORR: VarInformation, SimpleStats and PearsonCorr. So if we want to create data sets for all three of these destinations we specify:

```
ODS OUTPUT VarInformation=VarInfo SimpleStats=Stats PearsonCorr=Corr;
```

Where VarInfo, Stats and Corr are the names of the created data sets. If you leave the data set names out like in:

```
ODS OUTPUT VarInformation SimpleStats PearsonCorr;
```

SAS will use the well known method of assigning the names DATA1, DATA2, DATA3 and so on.

EXAMPLES 11: OUTPUT DESTINATION

We start with the same program as in Program 5. We know already that PROC CORR produces the output objects: VarInformation, SimpleStats and PearsonCorr. Program 11 shows how these objects are stored in data sets.

Program 11: Creating output data sets from output objects

```
ODS OUTPUT VarInformation=VarInfo SimpleStats=Stats PearsonCorr=Corr;
proc corr data=knmi;
var precipitation sunhours;
run;
proc print data=VarInfo;
title 'Varinfo';
run;
proc print data=Stats;
title 'Stats';
run;
proc print data=Corr;
title 'Corr';
run;
```

Output 11: Print of the created data sets of Program 11

```
Varinfo                                                     15:25 Friday, November 7, 2008   1

Obs    NVars    Variables           VarNames

 1        2     Variables:    Precipitation SunHours

Stats                                                       15:25 Friday, November 7, 2008   2

Obs    Variable             NObs          Mean         StdDev           Sum           Min

 1     Precipitation        2772       2.36465        4.69566          6555             0
 2     SunHours             2772         17308          14585      47978280             0

Obs           Max    Label

 1       42.50000    precipitation (mm)
 2          55800    sunshine duration (SAS time - HHMM)

Corr                                                        15:25 Friday, November 7, 2008   3

                                                                                  P
                                                              P                   P
                                                              r                   r
                                                              e                   e
                                                              c                   c
                                                              i                   i         P
   V                                                          p         S         p         S
   a                                                          i         u         i         u
   r                                                          t         n         t         n
   i              L                                           a         H         a         H
   a              a                                           t         o         t         o
O  b              b                                           i         u         i         u
b  l              e                                           o         r         o         r
s  e              l                                           n         s         n         s
1  Precipitation  precipitation (mm)                    1.00000  -0.29244     _        <.0001
2  SunHours       sunshine duration (SAS time - HHMM)  -0.29244   1.00000    <.0001     _
```

COMBINING OR SEPARATING OUTPUT

Sometimes a procedure can produce more than one object with the same name. These tables will be concatenated in the same output data set, unless you specify the option MATCH_ALL with the output specification. Then each object will be placed in a separate data set.

If you have objects of the same name in separate PROC steps you can force the output destination to remain active across PROC's, by adding the option PERSIST=PROC. This option can only be used in combination with the MATCH_ALL option.

If more data sets are created, SAS will use the name in the ODS OUTPUT statement as a base and add sequence numbers to them as needed.

EXAMPLE 12: MORE OUTPUT TABLES IN ONE OUTPUT DATA SET

First we show how data from multiple output objects is combined into one data set. We will use the same input data and create two tables with PROC FREQ. For this purpose we group the TEMPMAX and SUNHOURS variables in ranges by defining two formats (remember: PROC FREQ works on formatted values and note that time values are counted in seconds).

In the PROC PRINT output (Output 12) of the created data set, you find the results of both TABLE statements. Note that SAS provides a character and a numeric variable for the formatted analysis variables.

Program 12: Creating one data set from two tables

```
   proc format;
   value tempmax
   low-0='freezing'
   0-10='cold'
   10-20='moderate'
   20-30='warm'
   30-high='tropical';
   value sunshine
   low-0='clouded'
```

```
0-10800='mostly clouded'
10800-21600='partly clouded'
21600-high='sunny';
run;

ODS OUTPUT OneWayFreqs=FreqOutput;

proc freq data=knmi;
table tempmax;
table sunhours;
format tempmax tempmax. sunhours sunshine.;
run;

proc print data=FreqOutput;
run;
```

Output 12: Resulting output data set from Program 12

```
FreqOutput                                                          15:25 Friday, November 7, 2008   1

                F_Temp                                    Cum      Cum
Obs     Table       Max      TempMax   Frequency Percent Frequency Percent F_SunHours        SunHours

 1  Table TempMax  freezing freezing       30    1.08        30    1.08                              .
 2  Table TempMax  cold     cold          759   27.38       789   28.46                              .
 3  Table TempMax  moderate moderate     1245   44.91      2034   73.38                              .
 4  Table TempMax  warm     warm          698   25.18      2732   98.56                              .
 5  Table TempMax  tropical tropical       40    1.44      2772  100.00                              .
 6  Table SunHours                .       354   12.77       354   12.77  clouded        clouded
 7  Table SunHours                .       814   29.37      1168   42.14  mostly clouded mostly clouded
 8  Table SunHours                .       584   21.07      1752   63.20  partly clouded partly clouded
 9  Table SunHours                .      1020   36.80      2772  100.00  sunny          sunny
```

EXAMPLE 13: MATCH_ALL AND PERSIST=PROC

By adding MATCH_ALL, you create a separate data set per table (Program 13). In this case the data sets are named: FreqOutput and FreqOutput1 each containing the variables and observations pertaining to the right table.

If those tables are created in two different PROC steps (Program 13b) then MATCH_ALL is not enough. In this situation you have to include PERSIST=PROC as well. It does not matter whether there is another step between the two PROC FREQ's. Thanks to the PERSIST option any OneWayFreq output object will be written to a data set until you reset the selection with an ODS OUTPUT statement without any selection, so simply:

```
ODS OUTPUT;
```

For procedures with RUN-group processing, like PROC DATASETS there is also a PERSIST=RUN, which instructs SAS to keep building output data sets until the QUIT; statement is encountered.

Program 13: Creating separate output data sets per table

```
ODS OUTPUT OneWayFreqs(Match_All)=FreqOutput;

proc freq data=knmi;
table tempmax;
table sunhours;
format tempmax tempmax. sunhours sunshine.;
run;
```

Program 13a: Creating separate output data sets for tables across PROC steps.

```
ODS OUTPUT OneWayFreqs(Match_All persist=PROC)=FreqOutput;

proc freq data=knmi;
table tempmax;
format tempmax tempmax.;
run;
proc print data=FreqOutput;
run;
proc freq data=knmi;
table sunhours;
format sunhours sunshine.;
run;
```

Output 13: PROC PRINT output of the two data sets created by Program 13 or Program 13a

```
                                        The SAS System         10:17 Sunday, January 4, 2009   2

                                    F_Temp                                  Cum       Cum
  Obs   _Proc_   _Run_      Table      Max       TempMax   Frequency  Percent  Frequency  Percent

   1     Freq       1  Table TempMax  freezing  freezing        30      1.08         30     1.08
   2     Freq       1  Table TempMax  cold      cold           759     27.38        789    28.46
   3     Freq       1  Table TempMax  moderate  moderate      1245     44.91       2034    73.38
   4     Freq       1  Table TempMax  warm      warm           698     25.18       2732    98.56
   5     Freq       1  Table TempMax  tropical  tropical        40      1.44       2772   100.00
                                         The SAS System         10:17 Sunday, January 4, 2009   4

                                                                                  Cum      Cum
Obs _Proc_ _Run_     Table      F_SunHours     SunHours       Frequency Percent Frequency Percent

  1  Freq      1 Table SunHours clouded        clouded             354   12.77       354   12.77
  2  Freq      1 Table SunHours mostly clouded mostly clouded     2418   87.23      2772  100.00
```

CONCLUSION

The Output Delivery System is much more than a driver that lets you prepare output for many different purposes. Yes, it helps you format your output for a variety of destinations, like data sets, web pages, Office documents and more. But it is also a powerful tool to tailor your output almost any way you like, by enabling formatting your tables and cells, by creation of a table of contents, and by enabling traffic lighting and other tricks.

ACKNOWLEDGMENTS

Henk-Piet Glas and Ellen Lokollo of SAS Institute in the Netherlands helped me with some problems I ran into, while preparing this paper and were so kind to review the concept. Janet Grubber of Duke University Medical Center also provided valuable feed back. Finally my section chair Lisa Eckler gave several suggestions for improvement.

WHERE TO GO FROM HERE - RECOMMENDED READING

Books:

Bernadette Johnson: *Instant ODS: Style Templates for the SAS Output Delivery System*, SAS Press, 2003
Lauren Haworth: *Output Delivery System: The Basics,* SAS Press, 2001
Sunil Gupta: *Quick Results with the Output Delivery System*, SAS Press, 2003
SAS 9.1.3 Output Delivery System: User's Guide (Volumes 1 and 2), SAS Publishing, 2006

A selection of recent SAS Global Forum and SUGI papers:

Pete Lund: *PDF Can Be Pretty Darn Fancy: Advanced ODS Techniques for PDF Output*, SAS GF 2008, 033-2008
Myra Oltsik: *ODS and Output Data Sets: What You Need to Know,* SAS Global Forum 2008, 086-2008
Sunil Gupta: *SAS® ODS Technology for Today's Decision Makers,* SAS Global Forum 2008, 193-2008
Wendi Wright: *Make Your PROC TABULATE Tables Pretty Using ODS Style Options,* SAS GF 2007, 095-2007
Kevin Smith: *The Output Delivery System (ODS) from Scratch,* SAS Global Forum 2007, 219-2007
Eric Gebhart: *ODS Markup, Tagsets, and Styles! Taming ODS Styles and Tagsets,* SAS GF 2007, 225-2007
Rich Mays: *ODS Layout Is Like an Onion,* SUGI-31, 159-31
Eric Gebhart: *The Beginners Guide to ODS Markup: Don't Panic!* SUGI-31, 263-31
Ray Pass, Sandy McNeill: *PROC REPORT – Doing It with STYLE!,* SUGI-31, 116-31
Lauren Haworth: *Applying Microsoft Word Styles to ODS RTF Output* , SUGI-30, 043-30:
Ray Pass, Sandy McNeill: PROC TABULATE: Doin' It in Style!, SUGI-29, 085-29

The SAS9 ODS Tip sheets are a concise reference with "how to" tips per destination

CONTACT INFORMATION

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

Name	Erik Tilanus	Work Phone:	+31 343 517007
Address	Horstlaan 51	Fax:	+31 343 517007
Postal code, City	3971 LB Driebergen	E-mail:	erik.tilanus@planet.nl
Country	the Netherlands	Web:	www.synchrona.nl

At the website you can also find other presentations by the author, held at previous SUGI and SAS Forum meetings.