Paper 159-2010

Kick-start with ODS

Erik W. Tilanus, Driebergen, the Netherlands
Ellen Lokollo, SAS Institute, the Netherlands

ABSTRACT

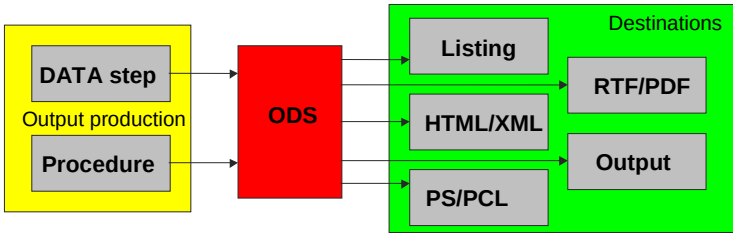
The Output Delivery System (ODS) forms an universal layer between SAS® DATA steps and procedures and the output they produce. This facilitates flexible formatting of outputs for a wide variety of output destinations: printers, printer-like files (PDF, postscript), screen (HTML) and so on. A special category 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

The reason for existence for any program, whether it is a SAS program or written in any other programming language is that it produces some kind of output that helps you further in what you are doing, whether it is business analytics, accounting, statistics or anything else. In a SAS program you can produce output either in a DATA step or in a PROC step. And you probably want to include that output in some type of report whether on paper or published on-line.

The Output Delivery System (ODS) helps you to prepare your output in such way that it best fits your needs.



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 handles 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 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 layout 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.

THE KNMI DATA SET – THE BASIS FOR THE EXAMPLES

Our examples will be based on a data set with meteorological data[1]. It is called KNMI after the Dutch acronym for the meteorological institute. It contains daily minimum and maximum temperatures in degrees Celsius, duration of sunshine as a SAS time value and millimeters of precipitation. The data set spans the period from 1 January 2001 to 17 November 2009, so it contains 3243 observations. To get aquainted with the data, the first 20 observations are printed with PROC PRINT (Table 1). SunHours is formatted with a HHMM format.

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';
```

[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

Obs	Date	TempMin	TempMax	Sun 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

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.

EXERCISE 1

Now let us create the PROC PRINT output shown in Table 1 in HTML format, RTF format and PDF format. We can do this all in the same run or do it one by one. Program 1 shows how to do this. The ODS statements are used in pairs: one to open the destination and one to close it again. Comment and un-comment them in pairs to see the effect.

Program 1: Sending output to various destinations

```
ODS HTML FILE="e:\SASForum\2010\Exercise1.HTML";
ODS RTF FILE="e:\SASForum\2010\Exercise1.RTF";
ODS PDF FILE="e:\SASForum\2010\Exercise1.PDF";
ODS LISTING CLOSE;
PROC PRINT DATA=KNMI(OBS=20);
RUN;
ODS HTML CLOSE;
ODS RTF CLOSE;
ODS PDF CLOSE;
ODS LISTING;
```

OUTPUT DESTINATIONS

In Exercise 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.

VARIANTS ON EXERCISE 1

You probably noticed that the output of Exercise 1 is looking slightly different depending on the destination. The actual formatting is based on 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.

Since we did not specify which template to use, SAS applied the default templates. For HTML output the default is Styles.Default. For RTF the default is Styles.RTF. That explains the difference. But SAS provides a wide variety of templates that you can use to change the look of your output.

Try changing the look of your output by including a style specification like in:

```
ODS HTML
    FILE="e:\SASForum\2010\Exercise1.HTML"
    STYLE= Astronomy;
```

To see which templates are available you can enter the command ODSTEMPLATES on the command line or right click in the Results window and select Templates. Choose any style and check the result.

SIMPLE MODIFICATIONS TO THE FORMATTING

Even though there are many different templates coming with the SAS installation you may want to create you own template, for instance to comply with company house styles. You create your own styles with PROC TEMPLATE. There is no need to specify a template from scratch. You may start from an existing template and just modify a few elements in them. That saves a lot of work.

EXERCISE 2

Assume we want to modify the PROC PRINT output of Exercise 1 as follows:

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 required PROC TEMPLATE code is presented in Program 2.

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;
```

We are going to define a new style: Forumstyle ①. By defining a parent style ② you can include all the existing

definitions from 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 ③. 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 ④ and style Data ⑤, 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 ⑥ 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.

Once you have defined the new template you can use it as presented before, by using the style specification in the ODS statement:

```
ODS HTML FILE="e:\SASForum\2010\Exercise2.HTML" style=Forumstyle;
ODS LISTING CLOSE;
proc print data=sgf.knmi(obs=20) ;
run;
ODS HTML CLOSE;
ODS LISTING;
```

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.

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\2010\bodyfile.HTML"
         CONTENTS="e:\SASForum\2010\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\2010\bodyfile.HTML"
         CONTENTS="e:\SASForum\2010\contentsfile.HTML"
         FRAME="e:\SASForum\2010\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.

Leaving out the PATH specification will store the files in the “Documents and Settings “ folder.

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=" c:\how\tilanus\" BASE="file:///";
```

Note however that the BASE option should be removed when preparing the output files for a web server.

EXERCISE 3

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 3243 observations. Let us add a second date variable, which we call Month and format with MONYY. and use that in a BY statement. Program 3 puts everything together..

Program 3: Generate a table of contents, based on BY values

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

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

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";
```

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. This is beyond the scope of this workshop. For a description you are referred to the paper “ODS 101+”. See references at the end of this paper.

CREATING A TABLE OF CONTENTS IN RTF OR PDF OUTPUT

In RTF or PDF files the table of contents is not a separate file, rather it is included in the main file. Therefore the specification is different. You indicate: CONTENTS=YES. For PDF files this is sufficient. For RTF files you may need to set another option as well: TOC_DATA. In SAS version 9.1.3 this is not required as this option is set by default, however in SAS 9.2 that default switched to NOTOC_DATA, resulting in an empty table of contents.

When opening the RTF file in MS-Office, the table of contents will not be visible directly. You have to right-click on it and select In that case you have to right-click on it and choose “Update field”.

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! The basic output is presented in Output 1. 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.

EXERCISE 4: USE OF ODS TRACE

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 4). When turned on, the SAS Log will contain information about each of them.

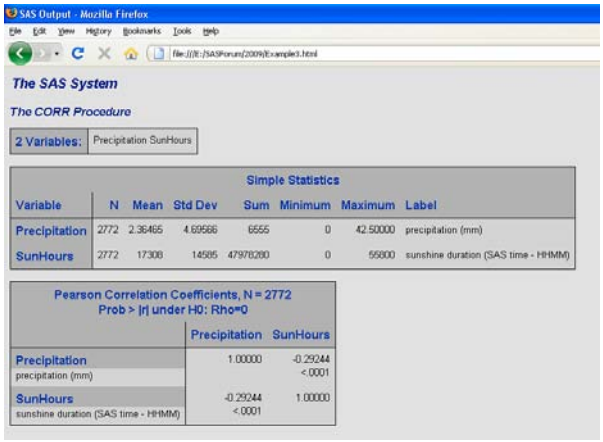
Program 4: Creating standard HTML output with PROC CORR

```
ODS html FILE="'c:\how\tilanus\Exercise4.html";
ods listing close;
ods trace on;

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

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

Output 1: The standard HTML output from PROC CORR



EXCLUDING ONE OR MORE OUTPUT OBJECTS

From the ODS TRACE information 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).

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!

EXERCISE 5: MODIFYING PROC CORR 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 5 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. 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 5: 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 5) it is rather straightforward to create the modified output::

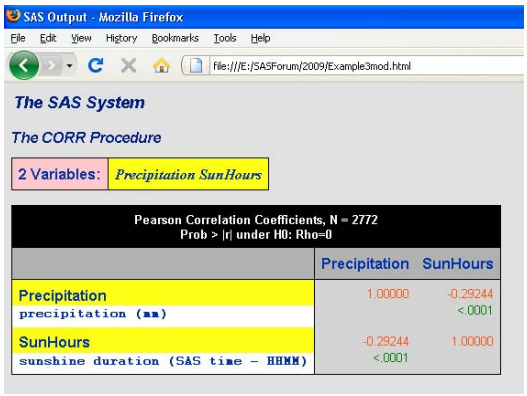
```
ODS html FILE="'c:\how\tilanus\Exercise5.html" style=ForumCorr;
ods listing close;
ods exclude SimpleStats;
proc corr data=knmi;
var precipitation sunhours;
run;
ods html close;
```

```
ods listing;
```

The result is presented in Output 2.

Be aware of the fact that we modified the table definitions of the tables that PROC CORR uses in Program 5. 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 2: the modified PROC CORR output



ODS GRAPHICAL OUTPUT

If your output should contain graphics there are two possibilities.

The first is very straight forward: just use one of the graphical procedures. These could be “classics” like PROC GPLOT but also the new SG procedures, like PROC SGPLOT.

The second possibility is to generate statistical graphics within the statistical procedures. For example PROC CORR can create scatter plots showing the individual points building the correlation. To make use of these graphics you need to activate them by specifying:

```
ODS GRAPHICS ON;
```

The graphics are just another output object and can be identified using the ODS TRACE command.

EXERCISE 6: CREATING GRAPHICS IN PROC CORR

This exercise contains the same PROC CORR as before, however due to the ODS GRAPHICS ON; statement also graphical output is produced and with the ODS TRACE ON; we also know the name of the graphical output object.

Program 6: Specifying statistical graphics

```
libname SGF 'c:\how\tilanus';
ods listing close;
ods html FILE="Example6.html" path='c:\how\tilanus' style=statistical;
*ods listing style=statistical;
ods graphics on;
ods trace on;
/*ods graphics editor;*/

proc corr data=sgf.knmi ;
where year(date)=2009;
var precipitation sunhours;
run;

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

EXERCISE 7: USING PROC SGPLOT

PROC SGPLOT is a new procedure that can produce plots as PROC GPLOT, but also various styles of charts as PROC GCHART does. The options to tailor your output are manifold and go far beyond the options of GPLOT etc. However the syntax is completely different. Teaching how to use PROC SGPLOT is beyond the scope of this workshop. But in this last exercise we want to give you a first impression of it’s capabilities.

Program 7 shows an example. It creates a graph charting the minimum and maximum daily temperatures for 2009 with some additional formatting.

Note the use of the PATH option in the ODS HTML statement. This makes sure that the graphic file will be stored in

the same location as the HTML file. By default the graph will be stored as a *.PNG file with a resolution of 640 x 480 pixels.

Program 7: Creating a graph with PROC SGPLOT

```
libname SGF 'c:\how\tilanus';
ods listing close;
ods html file='exercise7.html' path='c:\how\tilanus';

proc sgplot data=sgf.knmi description='This is a nice picture';
   where year(date)=2009;
   series x=date y=tempmin / curvelabel='Minimum Temperature'
                             curvelabelloc=outside
                             curvelabelpos=max;
   series x=date y=tempmax / curvelabel='Maximum Temperature'
                             curvelabelloc=outside
                             curvelabelpos=max;
   refline 10.7;
   label tempmin="Lineplot with Tempmin and Tempmax";
quit;

ods html close;
ods listing;
```

By changing the ODS HTML statements into ODS RTF … or ODS PDF … you can also create graphics directly in RTF respectively PDF files. In that case the PATH option is not necessary, the graph is embedded in the file. Note that for ODS PDF… the PATH option is not even valid!

CONCLUSION

In the workshop we have demonstrated and exercised the very basics of using ODS to format SAS output the way you want it. We hope that it has inspired you to experiment with ODS and PROC TEMPLATE to make your SAS output look just as you like it.

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

Paper:

Erik Tilanus: *ODS 101+,* SAS Global Forum 2009, 135-2009. This paper contains references to several other relevant SUGI and SGF papers.

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 authors 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

Name	Ellen Lokollo	Work Phone:	+31 356 996928
Address	Flevolaan 69	Fax:	+31 356 996905
Postal code, City	1271 PC Huizen	E-mail:	ellen.lokollo@snl.sas.com
Country	the Netherlands		

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