ABSTRACT
ODS Style templates can be used to control the basic appearance of the output, and, typically, PROC REPORT is used to build the tables that use these styles. ODS Table templates take this standardising to a new level, where the data just has to be prepared in a particular structure, and the template used in a DATA step to generate the standardised table. These templates are already available for use in SAS 9.1.3, so do we really need to continue using PROC REPORT?

TYPICAL PROC REPORT TABLES
The following examples of PROC REPORT steps and output are fairly typical of the programs used every day for clinical trial reporting.

REPORT CODE 1. A GROUPED TABLE WITH A TOTAL SUMMARY LINE

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>60.58889</td>
<td>90.11111</td>
</tr>
<tr>
<td>M</td>
<td>63.91</td>
<td>108.95</td>
</tr>
</tbody>
</table>

PROC REPORT DATA=SASHELP.CLASS NOWINDOWS;
COLUMN (Sex Height Weight);
DEFINE Sex / GROUP;
DEFINE Height / MEAN;
DEFINE Weight / MEAN;
RBREAK AFTER / SUMMARIZE;
RUN;

REPORT CODE 2. A TABLE WITH COMPUTED COLUMNS

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Weight</th>
<th>BMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>69</td>
<td>112.5</td>
<td>16.611531</td>
</tr>
<tr>
<td>Alice</td>
<td>56.5</td>
<td>84</td>
<td>18.498551</td>
</tr>
<tr>
<td>Barbara</td>
<td>65.3</td>
<td>98</td>
<td>16.156788</td>
</tr>
<tr>
<td>Thomas</td>
<td>57.5</td>
<td>85</td>
<td>18.073346</td>
</tr>
<tr>
<td>William</td>
<td>66.5</td>
<td>112</td>
<td>17.804511</td>
</tr>
</tbody>
</table>

PROC REPORT DATA = sashelp.class NOWINDOWS;
COLUMN (Name Height Weight _computed1);
DEFINE Name / ORDER;
DEFINE Height / MEAN;
DEFINE Weight / MEAN;
DEFINE _computed1 / COMPUTED "BMI";
RBREAK AFTER / SUMMARIZE;
COMPUTE _computed1;
    _computed1=weight.MEAN * 703 / (height.MEAN * height.MEAN);
ENDCOMP;
RUN;

REPORT CODE 3. A GROUPED SUMMARY TABLE WITH ACROSS COLUMNS

<table>
<thead>
<tr>
<th></th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td>Height</td>
<td>Weight</td>
<td>Height</td>
<td>Weight</td>
<td>Height</td>
<td>Weight</td>
</tr>
<tr>
<td>F</td>
<td>51.3</td>
<td>50.5</td>
<td>58.05</td>
<td>80.75</td>
<td>60.9</td>
<td>91</td>
</tr>
<tr>
<td>M</td>
<td>57.5</td>
<td>85</td>
<td>60.366667</td>
<td>103.5</td>
<td>62.5</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>54.4</td>
<td>67.75</td>
<td>59.44</td>
<td>94.4</td>
<td>61.433333</td>
<td>88.666667</td>
</tr>
</tbody>
</table>

PROC REPORT DATA = sashelp.class NOWINDOWS;
COLUMN (Sex Age,(Height Weight));
DEFINE Sex / GROUP;
DEFINE Age / ACROSS;
DEFINE Height / MEAN;
DEFINE Weight / MEAN;
RBREAK AFTER / SUMMARIZE;
RUN;

INTRODUCTION TO ODS TABLE TEMPLATES

ODS Table template code is a subset of the PROC TEMPLATE statements used specifically to create the TABLE templates. It has been available in production since SAS 8.2, and is used internally to control the output from many commonly used procedures.

The basic TABLE template is made up of nested structures:

```
DEFINE TABLE name; -- create the template
NMVAR name(s); -- define any macro variable parameters by name (optional)
COLUMN names(s); -- define any SAS variable parameters by name (optional)

DEFINE HEADER name; -- define headers (optional)
| Any definition statements, including text, justify, etc.
END;

DEFINE COLUMN name; -- define any columns

DEFINE HEADER name; -- define column headers (optional)
| Any definition statements, including text, justify, etc.
END;

Any definition statements, including justify, calculation, cellstyle, etc.
END;

END;
```

Definition statements include:
- JUST = justify CENTER, LEFT, RIGHT.
- TEXT – text to be included in a header.
- CELLSTYLE – specify the ODS Style properties for specific columns.
- START, END – used to specify first and last columns for headers.
REPLACING PROC REPORT FUNCTIONALITY

The following code examples use ODS Table templates and commonly used procedures, instead of PROC REPORT, to produce familiar reports.

ODS TABLE CODE 1. A GROUPED TABLE WITH A TOTAL SUMMARY LINE

The resulting report only differs from the same PROC REPORT output in the justification of the headers and the number of decimal places:

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>60.588888889</td>
<td>90.111111111</td>
</tr>
<tr>
<td>M</td>
<td>63.91</td>
<td>108.95</td>
</tr>
</tbody>
</table>

The summary data is generated by PROC SUMMARY, but we are not using the NWAY option, as it would restrict the summary levels available. In this case it is important, as all of the report data is stored in the summary data set and will be selected to generate the nested summary lines:

```sas
PROC SUMMARY DATA = sashelp.class MISSING;
CLASS sex;
VAR height weight;
OUTPUT OUT = _test1 MEAN(height weight) =;
RUN;
```

The separated summary data are recombined, a line counter __n added to each line, and the counter values of the summary lines recorded in __break, before being copied to the &__rbreak and &__break macro variables. These macro variables store the line numbers all of the summary rows. The default value of "0" for &__break is used to prevent errors when creating the __break format:

```sas
DATA _xtest1 (DROP = __:);
SET _test1 (WHERE = (_TYPE_ = 1)) END = eof;
BY sex;
LENGTH __break $200;
RETAIN __n . __break '0';
IF NOT FIRST.sex THEN sex = ' ';
OUTPUT;
__n + 1;
IF eof THEN DO;
SET _test1 (WHERE=(_TYPE_=0));
sex = ' ';
OUTPUT;
__n + 1;
CALL SYMPUT('&__rbreak', STRIP(PUT(__n,8.))); 
CALL SYMPUT('&__break', STRIP(__break));
END;
RUN;
```

This is a standard format to flag the summary lines:

```sas
PROC FORMAT;
VALUE __break
&__break = '1'
OTHER = ' ';
RUN;
```

The ODS Table template is then generated with DATAEMPHASIS ODS Style properties for the summary rows, and DATA properties for the other rows. The report data will be associated with either the order or cell columns, and the headers will be taken from the data labels:

```sas
PROC TEMPLATE;
DEFINE TABLE _test1;
NMVAR __rbreak __break;
CLASSLEVELS = ON;
```
Finally, the report is generated using a DATA _NULL_ step using the TEMPLATE _test1 that we created and assigning the data to the appropriate template columns:

```
DATA _NULL_;  
SET _test1;  
FILE PRINT ODS = (TEMPLATE = '_test1'  
   COLUMNS = (order = name(GENERIC = ON)  
               cell = height(GENERIC = ON)  
               cell = weight(GENERIC = ON)  
               )  
   );  
PUT _ODS_;  
RUN;  
```

ODS TABLE CODE 2. A TABLE WITH COMPUTED COLUMNS

The resulting report only differs from the same PROC REPORT output in the justification of the headers, the number of decimal places displayed, and the "bmi" header, as the calculated bmi variable had no associated label:

<table>
<thead>
<tr>
<th>Name</th>
<th>Height</th>
<th>Weight</th>
<th>bmi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>69</td>
<td>112.5</td>
<td>16.611531191</td>
</tr>
<tr>
<td>Alice</td>
<td>56.5</td>
<td>84</td>
<td>18.498551179</td>
</tr>
<tr>
<td>Barbara</td>
<td>65.3</td>
<td>98</td>
<td>16.156788436</td>
</tr>
<tr>
<td>Thomas</td>
<td>57.5</td>
<td>85</td>
<td>18.073345936</td>
</tr>
<tr>
<td>William</td>
<td>66.5</td>
<td>112</td>
<td>17.804511278</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>62.336842105</td>
<td>100.02631579</td>
<td>18.095892285</td>
</tr>
</tbody>
</table>

This time the report requires sorted data, so this is done first:

```
PROC SORT DATA = sashelp.class OUT = _test2;  
   BY name;  
RUN;  
```
Again the summary data is generated by PROC SUMMARY without the NWAY option, although here it is not really necessary, as the summary rows are not nested:

```sas
PROC SUMMARY DATA = _test2 MISSING;
  VAR height weight;
  OUTPUT OUT = _xtest2 MEAN(height weight) =;
RUN;
```

The sorted and summary data are combined, and a line counter `__n` added to each line, and the counter values of the summary lines recorded in `__break`, before being copied to the `&__rbreak` and `&__break` macro variables:

```sas
DATA _test2 (drop = __:);
  SET _test2 END = eof;
  BY name;
  LENGTH __break $200;
  RETAIN __n . __break '0';
  IF NOT FIRST.name THEN name = ' ';
  OUTPUT;
  __n + 1;
  IF eof THEN DO;
    SET _xtest2 (WHERE = (_TYPE_ = 0));
    name = ' ';
    OUTPUT;
    __n + 1;
    CALL SYMPUT('__rbreak', STRIP(PUT(__n, 8.)));
    CALL SYMPUT('__break', STRIP(__break));
  END;
RUN;
```

This is a standard format to flag the summary lines:

```sas
PROC FORMAT;
  VALUE __break
    &__break = '1'
    OTHER = ' ';
RUN;
```

Finally, the report is generated using a DATA _NULL_ step using the same TEMPLATE _test1 that we created in ODS Table Code 1 above, and assigning the data to the appropriate template columns after calculating the bmi variable:

```sas
DATA _NULL_;  
  SET _test2;
  bmi = weight * 703 / (height * height);
  FILE PRINT ODS = (TEMPLATE = '_test1'
    COLUMNS = (order = name(GENERIC = ON)
      cell = height(GENERIC = ON)
      cell = weight(GENERIC = ON)
      cell = bmi(GENERIC = ON)
    ));
  PUT _ODS_;  
RUN;
```

**ODS TABLE CODE 3. A GROUPED SUMMARY TABLE WITH ACROSS COLUMNS**

The resulting report only differs from the same PROC REPORT output in the justification of the headers and the number of decimal places displayed:

<table>
<thead>
<tr>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
<th>Height</th>
<th>Weight</th>
<th>Height</th>
<th>Weight</th>
<th>Height</th>
<th>Weight</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51.3</td>
<td>50.5</td>
<td>58.05</td>
<td>80.75</td>
<td>60.9</td>
<td>91</td>
<td>63.55</td>
<td>96.25</td>
<td>64.5</td>
<td>112.25</td>
<td>.</td>
</tr>
<tr>
<td>M</td>
<td>57.5</td>
<td>85</td>
<td>60.36</td>
<td>66.66</td>
<td>103.5</td>
<td>62.5</td>
<td>64.25</td>
<td>107.5</td>
<td>66.75</td>
<td>122.5</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>54.4</td>
<td>67.75</td>
<td>59.44</td>
<td>94.4</td>
<td>61.43</td>
<td>33.33</td>
<td>88.66</td>
<td>66.66</td>
<td>84</td>
<td>101.87</td>
<td>72</td>
</tr>
</tbody>
</table>

5
The report includes "across" variables, which will need to be identified before the template is created. This is best done in a macro:

```
%MACRO report;

The report requires sorted data, so this is done first:
PROC SUMMARY DATA = sashelp.class MISSING;
   CLASS sex age;
   VAR height weight;
   OUTPUT OUT = _test3 MEAN(height weight) =;
RUN;

The "across" variable is age, which must be analysed to find all the unique values present:
PROC SQL;
   CREATE TABLE _test3age AS
      SELECT DISTINCT
         age
      FROM   _test3
      ORDER BY
         age
;
QUIT;
DATA _NULL_;
   SET _test3age (WHERE = (age NE .)) END = eof;
   CALL SYMPUT('__num', STRIP(PUT(_N_, 8.)), STRIP(AGE));
   IF eof THEN CALL SYMPUT('__max', STRIP(PUT(_N_, 8.))); RUN;

The "across" variable values are then used to split the data values into groups for summarising into new data values at the different summary levels:
PROC SQL;
   CREATE TABLE _xtest3 AS
      SELECT sex
         %DO i = 1 %TO &__max.;
            MEAN(CASE
                  WHEN age = &&__num&i. THEN height
                  ELSE .
                  END) AS height&i.
            MEAN(CASE
                  WHEN age = &&__num&i. THEN weight
                  ELSE .
                  END) AS weight&i.
         %END;
      FROM   _test3 (WHERE = (_TYPE_ = 3))
      GROUP BY
         sex
;
QUIT;
PROC SQL;
   CREATE TABLE _xxtest3 AS
      SELECT ' ' AS sex
         %DO i = 1 %TO &__max.;
            MEAN(CASE
                  WHEN age = &&__num&i. THEN height
                  ELSE .
                  END) AS height&i.
            MEAN(CASE
                  WHEN age = &&__num&i. THEN weight
                  ELSE .
                  END) AS weight&i.
         %END;
      FROM   _test3 (WHERE = (_TYPE_ = 1))
;
QUIT;
```
The sorted and summary data are combined, and a line counter __n added to each line, and the counter value for the final summary line is recorded in the &__rbreak macro variable:

```
DATA _xxxtest3 (DROP = __:);
SET _xtest3 END = eof;
BY sex;
RETAIN __n .;
LABEL
%DO i = 1 %TO &__max.;
  height&i. = "Height"
  weight&i. = "Weight"
%END;
__n + 1;
IF NOT FIRST.sex THEN sex = ' ';
OUTPUT;
IF eof THEN DO;
  SET _xxt3est3;
  sex = ' ';
  OUTPUT;
  __n + 1;
  CALL SYMPUT('__rbreak', STRIP(PUT(__n, 8.)));
END;
RUN;
```

This ODS Table template is different to the previous example, as the columns have to have explicit data names. The DATAEMPHASIS ODS Style properties are still applied to the summary rows, and DATA properties for the other rows. The headers are generated from the original data names:

```
PROC TEMPLATE;
DEFINE TABLE _test3;
  NMVAR __rbreak;
  CLASSLEVELS = ON;
  COLUMN sex
%DO i = 1 %TO &__max.;
    height&i. weight&i.
%END;
  DEFINE HEADER myheader;
    TEXT "Age";
    JUST = CENTER;
    START = height1;
    END = weight&__max.;
  END;
%DO i = 1 %TO &__max.;
  DEFINE HEADER myheader&i.;
    TEXT "&__num&i.";
    JUST = CENTER;
    START = height&i.;
    END = weight&i.;
  END;
  DEFINE COLUMN height&i.;
    DEFINE HEADER height&i._hd;
    JUST = CENTER;
  END;
  CELLSTYLE _ROW_ = __rbreak AS DATAEMPHASIS
    _ROW_ NOT = __rbreak AS DATA
    ;
  END;
  DEFINE COLUMN weight&i.;
    DEFINE HEADER weight&i._hd;
    JUST = CENTER;
  END;
```

7
Finally, the report is generated using a DATA _NULL_ step using the TEMPLATE _test3_. No columns are specified, as the template is data-specific:

```
DATA _NULL_;  
SET _xxxtest3;  
FILE PRINT ODS = (TEMPLATE = '_test3');  
PUT _ODS_;  
RUN;  
%MEND report;  
%report;  
```

**ODS TABLE CODE 4. STACKED VALUES IN A CELL**

The report is not possible with PROC REPORT in the same way for all ODS destinations, but has been included to demonstrate the other capabilities of ODS Table templates. The resulting simple report shows the 3 values stacked in the right-hand cell of each row:

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfred</td>
<td>M</td>
<td>14</td>
<td>69</td>
<td>112.5</td>
</tr>
<tr>
<td>Alice</td>
<td>F</td>
<td>13</td>
<td>56.5</td>
<td>84</td>
</tr>
<tr>
<td>Barbara</td>
<td>F</td>
<td>13</td>
<td>65.3</td>
<td>98</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Thomas</td>
<td>M</td>
<td>11</td>
<td>57.5</td>
<td>85</td>
</tr>
<tr>
<td>William</td>
<td>M</td>
<td>15</td>
<td>66.5</td>
<td>112</td>
</tr>
</tbody>
</table>
PROC TEMPLATE;
DEFINE TABLE _test4;
COLUMN name sex (age height weight);
DEFINE COLUMN age;
DEFINE HEADER ageheightweight;
TEXT "Age*Height*Weight";
SPLIT = "*
";
END;
END;
END;
RUN;

As above this report is generated using a DATA _NULL_ step, but using the TEMPLATE _test4_. No columns are specified, as the template is data-specific:

DATA _NULL_;  
SET sashelp.class;
FILE PRINT ODS = (TEMPLATE = '_test4');
PUT _ODS_;
RUN;

PROS AND CONS

DISADVANTAGES
• The obvious disadvantage is the volume of programming required for the code using ODS Table templates.
• The code required for grouped reports is more complicated than for ordered reports.
• Data for reports with "across" columns requires pre-processing to determine the unique values present.

ADVANTAGES
• In most cases the ODS Table templates can be re-used unchanged for a wide range of reports.
• Data processing only requires knowledge of PROC SORT, PROC SUMMARY and PROC SQL, in addition to the re-usable template code.
• It is also possible to include the coding of computed columns in the ODS Table templates, but including the calculations in the DATA _NULL_ step gives more flexibility. However, in both locations the computed columns can be located anywhere in an output row, whereas PROC REPORT requires them to appear to the right of any columns used in their calculation.
• The resulting output is 100%-compatible with all ODS destinations, unlike PROC REPORT output.
• ODS Table templates include features not found in PROC REPORT, like the stacking of 2 or more values in a single report cell with stacked headers, which could be useful when reporting statistics.

CONCLUSIONS
• This technique is not intended necessarily to replace the use of PROC REPORT, but to suggest some alternative methods of creating the same output.
• This technique is very complicated to use with "across" variables, as the data must be pre-processed to find all the unique values present, but has advantages over PROC REPORT when the positioning of computed columns needs to be flexible.
• ODS Table template offer new features, like stacked values, which could be useful when reporting data statistics in a more compact, but readable, layout. This would be particularly helpful when the expected width of the report needs reducing.

REFERENCES
• All the PROC REPORT and ODS Table code in this paper was based on code generated using the evaluation Enterprise Guide Add-Ins that can be downloaded from www.hollandnumerics.com/SOFTWARE.HTM
PhUSE 2008

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

Author Name: Philip R Holland
Company: Holland Numerics Ltd
Address: 94 Green Drift
City / Postcode: Royston, Herts SG8 5BT, UK
Work Phone: +44 7714 279085
Fax: +44 1763 244497
Email: phil@hollandnumerics.com
Web: www.hollandnumerics.com

This paper and associated sample SAS code can be downloaded from the Holland Numerics Ltd web site at www.hollandnumerics.com/SASPAPER.HTM

Brand and product names are trademarks of their respective companies.