Background
Over the years I have automated reporting of clinical data
- used data definition for ADAM dataset to generate SAS code
- expanded data display plan to meta data usable for generation of SAS code to generate tables, listings and graphs (TLG)
- used a simple command language to execute mainly pre-specified reports using different not pre-specified sub groups

Much of the work consist of making datasets and output like the previous with a slight change. With “Specification like this” (SLT) the data sets and outputs are specified using plain text, processed by SAS programs to generate programs and documentation for reporting as exemplified below.

Experience
- It is time consuming to make the initial full specification, changes are fast
- desirable side-effect are high-quality meta-data for data sets and outputs
- documentation and programs are by design synchronized
- consistent naming is easy attained
- nice layout in generated programs is difficult (indentation and line breaks)
- writing programs for code generation is not more difficult than making SAS macros
- code generation can be used to make “executable code” as well as calling standard scripts

Ongoing development
- code generation targeting other languages, such as R
- automated checks of the specifications, such as naming conventions and consistency
- use more direct approach for code generation (StringTemplate)

Conclusion
After using the framework for reporting of two studies and other tasks, I still find it is worthwhile approach supporting “make it like table 14 fast only changing ...”.

Programs available – send me an e-mail.

---

**1. Create specification**

```SAS
!SLTDEF TLG
TLGId: E1
Title: Descriptive statistics
OutputId: @{$TLGId$}
Tableinputdataset: ADAM.ADBL
Tableoutname: Result.Table @{$OutputId$}
Analysevars: sex/cl age/co weight/co
Colvar: trtr01a
Tbyvarvalue: Placebo Active
Tbyvarnamelist: @{$Colvar$}
TLGprogramname: Table @{$OutputId$}
!SLTEND
```

**2. Expand specification into datasets**

```SAS
/* Code snippet showing code generation */
codeline=cats(“”, “define”, variable_name, “/”,”
            ifc(missing(report_label), “”, quote(strip(report_label))),
            report_usage, report_printwidth, report_just,
            ifc(missing(report_format), “”, cats(“format”,”=”,report_format) ),”,”) ;
output;
  if report_computebefore=rowabove then do;
    codeline= cats(“”, “compute before”, variable_name,
        ”/ style={just=left font_weight=bold}” ),”;”;
  output;
  codeline= cats(“”, “line #2”, variable_name, report_format),”;”);
  output;
  codeline=endcomp;
end;
```

**3. Create documentation and programs from the datasets**

```SAS
!--- Code snippet showing code generation ---
```

**4. Generated code**

```SAS
/* Code snippet showing the generated code */
proc report data=expdn missing nobs spacing=1 split=’#’ spanrows;
  column Variable_sequ Variable_variable_label statesequ statname
  ResColumn1_Active ResColumn1_Placebo;
  define Variable_sequ / order noprnt right;
  define Variable / order noprnt left;
  define variable_label / order noprnt left;
  compute before Variable_label / style={just=left font_weight=bold};
  line #2 variable_label $200.;
endcomp;
```

**5. Output**

```SAS
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Active</th>
<th>Placebo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>5 (55.6%)</td>
<td>4 (40.0%)</td>
</tr>
<tr>
<td>M</td>
<td>4 (44.4%)</td>
<td>6 (60.0%)</td>
</tr>
<tr>
<td>Total (N)</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Age (years)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 (Hmns)</td>
<td>9 (0)</td>
<td>10 (0)</td>
</tr>
<tr>
<td>Mean (sd)</td>
<td>13.1 (1.5)</td>
<td>13.5 (1.6)</td>
</tr>
<tr>
<td>Weight (pounds)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 (Hmns)</td>
<td>9 (0)</td>
<td>10 (0)</td>
</tr>
<tr>
<td>Mean (sd)</td>
<td>100 (16)</td>
<td>100 (29)</td>
</tr>
</tbody>
</table>
```