Statistics without DATA _NULLS_

Michael C. Palmer and Cecilia A. Hale, Ph.D.
Zurich Biostatistics, Inc.

The recent release of a new software standard can substantially ease the integration of human, document, and computer resources. The techniques reported here demonstrate how the reporting of statistical analyses can benefit.

Abstract

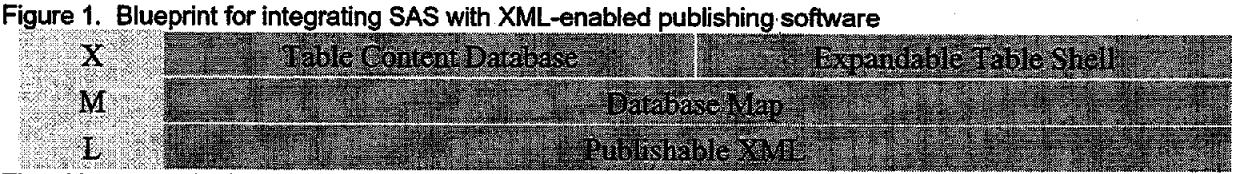
Analysts attracted to SAS® for its statistical capabilities no longer have to settle for its clumsy publishing capabilities. Instead, they can take advantage of the Output Delivery System (ODS) in version 7.0, SAS's data warehouse and data processing capabilities, and a new open-source, non-proprietary software technology, and literally never use a DATA _NULL_, PROC REPORT, or PROC TABULATE again. An analyst working with publishing software and exploiting its layout and styling flexibility creates a terse sample table. The sample, via SAS programs, directs the retrieval of data from a warehouse and the creation of publishable markup. The publishing software renders the markup publication ready without re-typing, cutting-and-pasting, or format conversion of SAS output. The new, platform-independent software technology is eXtensible Markup Language (XML). Statistical analysts and their organizations will better exploit the strengths of SAS and publishing software by coupling the two via XML.

Introduction

Statisticians rarely resort to custom programming of statistical methods when they work with SAS. It is equally rare that clients and colleagues are satisfied with SAS output before any custom table programming is done. This paper presents a blueprint and some useful techniques for integrating SAS with document publishing software that automate the programming of statistical tables and apply the styling and composition capabilities of the publishing software directly to the output. As a result, SAS can do what it does best, statistical analysis, and the publishing software can do what it does best, compose and style documents.

The blueprint and techniques presented here were developed during ongoing research at Zurich Biostatistics (ZBI) into ways to automate the statistical analysis of clinical trials. They have been demonstrated in a system that integrates statistical table content stored in SAS data sets with table shells in an electronic format. Figure 1 is a schematic of the components of the integration and their relationships.

Figure 1. Blueprint for integrating SAS with XML-enabled publishing software



The table content database contains analysis results and table text. The expandable table shell contains the table style and composition. The database map associates shell entries with data in the content database. Combining the table content with the shell yields publishable XML. XML is also the medium of data exchange throughout this process.

The technology that makes this integration work is XML. XML (eXtensible Markup Language) is a standard for computerized document publishing and exchange developed by the World Wide Web Consortium (W3C) (http://www.w3.org/TR/REC-xml). The W3C released version 1.0 of XML in February 1998. The W3C is the standards body for the World Wide Web on the Internet and it designated XML as the successor standard to HTML, the current web page standard. XML, like HTML, is non-proprietary and platform-independent. As the successor to HTML, XML is optimized for web usage, but it also has broader applicability as a medium for data exchange and publishing in paper and electronic formats.

The example that follows illustrates the blueprint. The table content database consists of SAS version 7 data sets. The document publishing software is Adept™ version 8.0 from Arbortext, Inc. Adept is XML-enabled. Code used

to process XML was written with SAS version 7. This integration of commercial, off-the-shelf software and non-commercial software developed at ZBI has successfully scaled to larger projects. The example is very simple, but instructive, rather than complicated and real. The table views below are from PDF files produced with Adobe® Acrobat® 4.0.

Example

The example table content database has two data sets, PLdb00 and PLdb01 (Figure 2). PLdb00 has one observation, a code for the table title. PLdb01 has two observations that are analysis results. The CONTENT0 field has the table content. FORMAT0 has the format for CONTENT0, and the other fields are key variables.

Figure 2. Table content database: two data sets for the example table

Z U R I C H B I O S T A T I S T I C S
SAS Users Group 1999 Presentation

PLdb00

STATISTIC0	MEASURE0	FORMAT0	CONTENT0
LABEL	TITLE	TEXT.	0

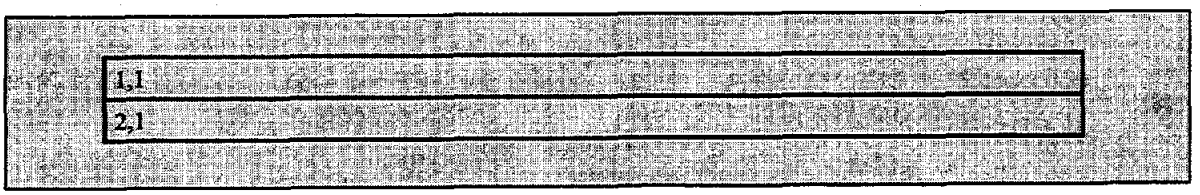
PLdb01

STATISTIC0	MEASURE0	FORMAT0	CONTENT0
MEAN	DBP	DEMO.	120
MEAN	SBP	DEMO.	80

All content for the table, text and numeric, is stored in the database. No content appears in the shell. This facilitates table revision and reuse of the shells. The data sets have a highly vertical structure with one element of content per record. All content is stored in the numeric CONTENT0 variable. The other variables are keys. This facilitates generic programming. SAS formats are used to decode CONTENT0 to text for titles, footnotes, labels, and headings.

The expandable table shell has two rows and one column. Figure 3 show the starting point for developing the shell, a table with Adept's default appearance and each entry indexed with its row and column numbers.

Figure 3. Example table and underlying XML



This table was created in Adept's menu-driven editor. Adept produced the underlying XML statements shown below.

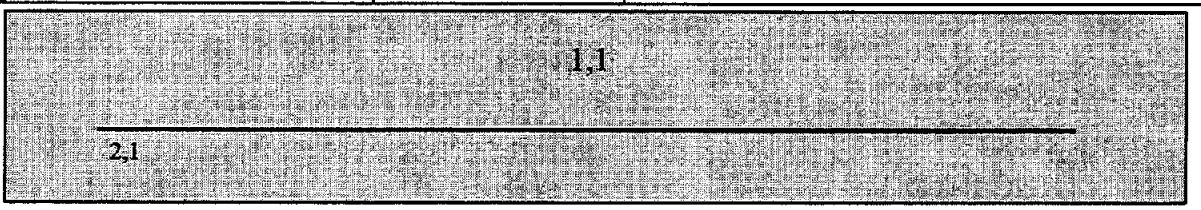
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE cals02 PUBLIC "ZBI//DTD CALS Table Exchange Test 2//EN" "">
<!--Zurich Biostatistics, Inc. www.zbi.net-->
<cals02>
<table>
<tgroup cols="1">
<colspec colname="col1"/>
<tbody>
<row>
<entry colname="col1">1,1</entry>
</row>
<row>
<entry colname="col1">2,1</entry>
</row>
</tbody>
</tgroup>
</table>
</cals02>
```

Table content is delimited by <entry> and </entry> tags. Table rows are delimited by <row> and </row> tags.

An objective of integrating SAS with document publishing software is to apply the publishing software's style and composition capabilities directly to statistics computed in SAS. Figure 4 illustrates the modification of the example

table shell to do that. A word-processor style interface was used and the underlying XML statements were created.

Figure 4. Modification of the example table – the final expandable table shell



The table in Figure 3 was edited in Adept's editor to turn off the default table frame, center-align the content in entry 1,1, increase the height of the first row, and to set a 16 point font size for the content in the first row. Below is the modified XML.

```
...
<table frame="none">
...
<row>
<?PubTbl row rht="0.67in"?>
<entry colname="col1" align="center">
<?Pub _cellfont TypeSize="16pt"?>1,1</entry>
</row>
<row>
<entry colname="col1">2,1</entry>
</row>
...
```

Adept added the frame="none" attribute to the <table> tag in Figure 1 and the align="center" attribute to the <entry> tag in the first row. It also added the <?Pub> tags for row height and font size. This XML table shell will be expanded into the finished table.

The appearance of the table in Figure 3 was determined by the OASIS Exchange CALS table standard, which is supported by a number of XML-enabled packages, and by the default settings of Adept. The on-screen modifications to the table in Figure 3 had several effects on the underlying XML code. Style attributes were added to <table> and <entry> tags as a result of removing the table frame and centering the content of the first row. These are non-default values for attributes in the table standard. The other on-screen changes resulted in the addition of <?Pub> tags for row height and font size. These are Adept-specific options. The W3C has released a draft style standard called XSL (eXtensible Style Language) and presumably when that is finalized, XML-enabled publishing software will add XSL capabilities that will not be software product-specific.

Figure 5 contains the database map that associates each entry in the table shell with data in the table content database. In the example, the row 1 entry will display data from the PLdb00 data set. Since PLdb00 has only one observation, no rule is necessary to select data. The row 2 entry will select data from the PLdb01 data set where the key variable STATISTIC0 has the value of MEAN and will sort the selected data by MEASURE0 before putting it into the table shell.

Figure 5. Database map for the example table shell

Tekoa Database Map

DEMO00

Location on Table		DATASET	CONDITION	
Row	Column	Dataset	Keys	Rules
1	1	PLdb00	STATIS-TIC0	
2	0		MEA-SURE0	
2	1	PLdb01		IF STATISTIC0='MEAN'

A database query is defined for each entry in the table shell. The "Dataset" column is the location of data for the entry. The "Keys" column specifies key variables in "Dataset" to use in sorting the table data. "Rules" are applied to "Dataset" to select the data for the table. Each entry inherits from the "0" column for its row.

The table shell in Figure 4 and the map in Figure 5 contain all of the instructions necessary to create the sample table.

Figure 6 shows an XML file that integrates the composition and style instructions in Figure 4 with the table content instructions in Figure 5. To build the XML shown below, the "Key" and "Rule" instructions were extracted from the XML version of the database map and placed into the shell, replacing the corresponding row and column indices. Since XML is extensible, that is, tag names can be added and their functions defined, it is possible to use a family of <ZBI:> tags to add the key and rule instructions. All of the instructions necessary to build the sample table are now in one place, the XML file shown in Figure 6.

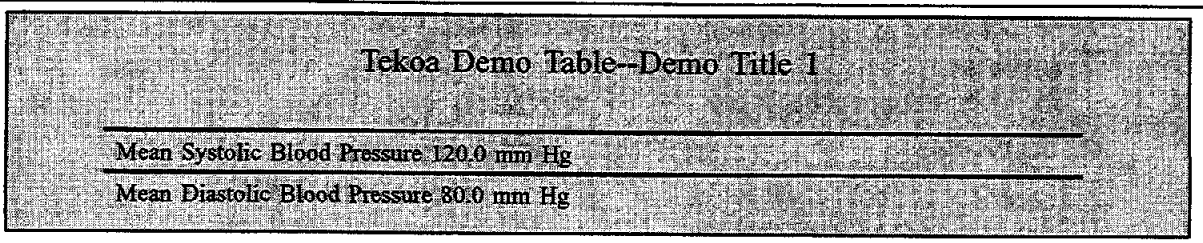
Figure 6. Table shell XML combined with XML statements that define database queries

```
...
<table frame="none">
...
<row>
<?PubTbl row rht="0.67in"?>
<entry colname="col1" align="center">
<?Pub _cellfont TypeSize="16pt"?>
<ZBI:DATASET DS="PLdb00" />
<ZBI:CONDITION KEYS="STATISTIC0" />
</entry>
</row>
<row>
<ZBI:CONDITION KEYS="MEASURE0" />
<entry colname="col1">
<ZBI:DATASET DS="PLdb01" />
<ZBI:CONDITION RULE="IF STATISTIC0='MEAN';" />
</entry>
</row>
...
```

The placeholder entry content in Figure 5 has been replaced by <ZBI:> tags that define database queries.

Figure 7 shows the finished table after processing the XML in Figure 6. The XML in Figure 6 differs from that in Figure 7 in two ways. First, the <ZBI:> tags have been replaced by the table content that they pointed to in the table content database and, second, the second row of the table has been repeated because the data query for that row retrieved two records. This illustrates why the table shell is termed "expandable." It is designed with knowledge of the structure of the table content database but without knowledge of how many values or how many observations there are for variables that define rows. The shell expands row-wise to fit the data.

Figure 7. The finished table



To produce this table from Figure 6, the table content database was queried, the table shell was expanded because the table content database contained two records meeting the query specification in the second row in the shell, and the <ZBI:> XML statements were replaced by the retrieved data. Below is an excerpt of the XML for this table.

```
...
<table frame="none">
...
<row>
<?PubTbl row rht="0.67in"?>
<entry colname="col1" align="center">
<?Pub _cellfont TypeSize="16pt"?>Tekoa Demo Table--Demo Title 1</entry>
</row>
<row>
<entry colname="col1">Mean Systolic Blood Pressure 120.0 mm Hg</entry>
</row>
<row>
<entry colname="col1">Mean Diastolic Blood Pressure 80.0 mm Hg</entry>
</row>
...
```

The XML in Figure 7 is identical to that in Figure 6 except that table content has replaced the <ZBI:> tags.

Techniques

Several of the techniques noted or alluded to above bear further explanation.

All table content, statistical results, titles, footnotes, row headings, and column headings, are stored in the table content database and associated format library. The table shells contain no content, just row and column indices. With this restriction, revisions to table text and content are a matter of edits to the table content database and format library, not program changes or table shell changes. Table shell reuse is also facilitated since the shells contain only style and composition instructions.

The table content database has a uniform, vertical structure. Table content on every record in every data set is in one and only one place, a numeric variable called CONTENT0 with an associated format in FORMAT0. Key variables to identify the content may vary from data set to data set and administrative variables to track the content may vary, but the content variable does not. This rigidity in database design facilitates generic programming to process queries extracted from Figure 6-type XML.

XML is extensible. Tags and tag families can be designed for specific functions in the document to be published or in software that processes the document at some point during its development.

Version 7 of SAS has regular expression functions that make it much easier to manipulate text than it was in earlier SAS versions. Figure 8 contains a DATA step that demonstrates how regular expressions can be used to process XML statements. In the step, regular expression functions are used to identify XML tags in the <ZBI:> family and to extract SAS statements that are embedded in the XML as attributes. In version 7, the length of variables and format decodes has been increased to 32k. The combination of regular expressions and 32k text strings makes SAS a workable platform for developing applications that process XML.

Figure 8. Example of a data step for parsing XML statements

```
01   DATA QUERY_INGREDIENTS(KEEP=QUERY_RULE);
02      LENGTH RULE_FOUND QUERY_RULE XML_STATEMENT $ 256;
03      INFILE 'c:\ZBI\PRESENTATIONS\SUG\SHELL_3.TXT' END=EOF LENGTH=STATEMENT_LENGTH;
04      INPUT XML_STATEMENT $VARYING256. STATEMENT_LENGTH;
05      RETAIN RX_ZBI RX_QUERY_RULE RX_QUOTED_STRING;
06      IF _N_=1 THEN DO;
07        RX_ZBI=RXPARSE("ZBI:");
08        RX_QUERY_RULE=RXPARSE("RULE$'='$Q");
09        RX_QUOTED_STRING=RXPARSE("$Q");
10      END;
11      IF RXMATCH(RX_ZBI,TRIM(LEFT(XML_STATEMENT))) THEN DO;
12        CALL RXSUBSTR(RX_QUERY_RULE,TRIM(LEFT(XML_STATEMENT)),POSITION,LENGTH);
13        IF POSITION GT 0 THEN DO;
14          RULE_FOUND=SUBSTR(TRIM(LEFT(XML_STATEMENT)),POSITION,LENGTH);
15          CALL RXSUBSTR(RX_QUOTED_STRING,RULE_FOUND,POSITION,LENGTH);
16          QUERY_RULE=SUBSTR(RULE_FOUND,POSITION+1,LENGTH-2);
17          OUTPUT QUERY_INGREDIENTS;
18        END;
19      END;
20      IF EOF THEN DO;
21         CALL RXFREE(RX_ZBI);
22         CALL RXFREE(RX_QUERY_RULE);
23         CALL RXFREE(RX_QUOTED_STRING);
24      END;
25   RUN;
```

SAS version 7 includes regular expression functions such as RXMATCH and RXSUBSTR that are useful in extracting text from XML statements. In the example above, XML statements that include the string ZBI: are selected. Each selected statement is scanned for the string RULE=a quoted string and, if found, the contents of the quoted string are extracted and output to the QUERY_INGREDIENTS data set.

Discussion

Any result from SAS that can be placed in a table content database can be published with the approach outlined here. That is virtually all of the statistics produced by ODS-enabled procedures in version 7. Regular expression functions and 32k length variables and formats, new in version 7, facilitate the processing of XML.

The approach in this paper relies on XML-enabled, commercial off-the-shelf software to create table shells and database maps and for publishing finished tables. The XML standard, as of this writing, is barely 18 months old and XML-enabled publishing software is not widely known. Arbortext's Adept version 8 was used for the work reported here. It was chosen because it adheres closely to the published XML standard, has a standard table model that is adequate for statistical tables, and has styling and composition capabilities suitable for clinical trial reports and drug regulatory submissions. Adept has, so far, integrated well with SAS through the medium of XML. A way to integrate it with Microsoft Word® has been more problematic. Microsoft participated in the development of the XML standard and is developing XML-based products, but Word is XML-illiterate. XML-to-RTF conversion programs are available but ZBI has not yet evaluated them. RTF is Microsoft's relatively portable file format for Word. The solution adopted at ZBI has been to publish material from different software, Adept, Word, WordPerfect®, or SAS, to PDF files using Adobe Acrobat and then to assemble a final PDF format document from these files. PDF is a proprietary format owned by Adobe Systems. In ZBI's experience, PDF files viewed on-screen with Adobe's free Acrobat Reader® usually look just like the printed version of a file, as long as the file and Reader are compatible versions.

Conclusion

Statisticians working with SAS, publishing software, and XML can assemble electronic documents that replace the traditional table programs and SAS-to-word processor conversion tasks. These documents direct the retrieval of statistical results and table text from a database and the production of report-quality statistical tables. No table programming is necessary. Tables produced this way are easily revised and reused and can be published in print or electronic formats.

Contact the authors at: Zurich Biostatistics, Inc., 45 Park Place South, PMB 178, Morristown, NJ 07960; (973)727-0025; or mcpalmer@zbi.net, cahale@zbi.net.