

Custom Outcomes Benchmarking Reports Using Dynamic Data Exchange (DDE), the Only Way to Go.

Rebecca Ottesen, Julie Kilburn, Ann Vanderplas, Anna Ter Veer, Rizvan Mamet, Joyce Niland, City of Hope, Duarte, CA

ABSTRACT

Often in prospective health care research studies, it is necessary to provide reports to determine how well a study is progressing. The National Comprehensive Cancer Network (NCCN) Breast Cancer (BCA) Outcomes Database produces highly customized annual benchmarking reports for its twelve NCCN institutions and fifteen community sites. These Microsoft® Excel® reports are designed to present statistics and accompanying graphics in such a way that each of the 27 center specific reports shows aggregate center information as well as center ranges and individual center specific data. Exporting data and statistics from SAS® to an Excel report can be accomplished many ways. However, when dealing with a highly customized Excel reporting structure the optimal way is to use DDE. Some may argue that DDE is obsolete, however in this case we have found the best method to be DDE with additional report formatting techniques using Excel version 4 Macro Language (X4ML). We will discuss our experience in using DDE to output statistics for tables and data for graphics across multiple worksheets to create benchmarking reports for the NCCN BCA Outcomes Database Project. In addition we will discuss incorporating SAS macros to create standardized code that allows for flexibility in how the data are output and formatted, as well as opening, saving and creating multiple versions of the report for each NCCN center.

INTRODUCTION

THE NATIONAL COMPREHENSIVE CANCER NETWORK

The NCCN is an alliance of 21 of the nation's leading cancer centers across the United States. The NCCN is responsible for development and maintenance of several major contributions to the clinical community. The most widely used product of the NCCN are its NCCN cancer care guidelines, which are evidence-based pathways used by clinicians and patients to follow treatment recommendations based on the clinical characteristics of their cancer. The guidelines include detailed information about treatment for 31 main types of cancer sites. The NCCN also has created and maintains the NCCN Drug Compendium, and peer-reviewed publication, the Journal of the National Comprehensive Cancer Network. The NCCN also is the main sponsor of the NCCN Outcomes Database. (For more information about the NCCN go to www.nccn.org.)

The NCCN Outcomes Database Project provides a central repository of data collected on patients with breast, Non-Hodgkin's Lymphoma, colon/rectal (CRC), lung, and ovarian cancers. The first database created by the Data Coordinating Center (DCC) located at City of Hope (COH) was for breast cancer data collection, which began in 1997. Currently the BCA database contains data from 12 different NCCN institutions, and from 15 community sites affiliated with an NCCN institution. This ever growing database currently contains approximately 42,900 BCA patients, and will soon expand to 16 NCCN institutions and 16 community sites. The system is structured as a relational database with some tables being one-to-one records, such as demographic information, and other tables being one-to-many, such as treatment information. The data is stored in a SQL database which is centrally located and maintained by the COH DCC. The study data can be entered into the database real time via direct web entry or, for institutions with pre-existing compatible databases, via FTP transmission on a quarterly basis.

REPORTING

The purpose of the NCCN BCA Outcomes Database Project is primarily to provide reporting and analysis on the patterns and outcomes of care, including performance measurements based on the NCCN guidelines. To achieve these goals, we have four major types of reports that are generated with SAS programming.

The first type of report is based on live data. An example of this is the patient summary report that a clinician can use to see a quick summary of all the data entered on a patient across 19 data tables. This summary is generated through the database, however it also includes information on which guidelines the patient falls, and his/her adherence to these guidelines based on the clinical data. All of the guideline algorithm programming is done with SAS; to provide the live report with these results the SAS programming is batch automated. The program outputs its results to a text file each night, which then feeds directly into the patient summary report.

Quarterly reporting is the second type of report, and is set to coincide with the FTP submissions. These reports also are programmed in SAS and include information routinely used for quality assurance (QA) visits to the sites. One such report is the missing/unknown report which displays frequency and percentage of missing and unknown values across all variables in the database. This report has built-in conditions for relevant variables, such as only reporting missing death date if the variable is missing and a cause of death is entered. The missing/unknown report is output to Excel for ease of use with the data management staff. Another example of quarterly reporting programmed in SAS are the QA reports which are more detailed queries based on reasonable clinical assumptions such as diagnosis date and procedure data conflicts. These reports are reviewed by the data management staff, and if in fact the data are correct then a flag built into the report is set so that the query will not flag the patient again for that specific data and query. Programming techniques used in our quarterly reports will be discussed at WUSS by Kilburn (2009).

The third category of database reporting represents analyses in the form of presentations for professional conferences and publications in peer reviewed journals. Each disease-specific database is governed by a Disease Specific Executive Committee (DSEC) and one of their many functions is to oversee analytic concepts that result from the database. Analyses are run centrally using SAS as a collaboration of the first author, DSEC, Scientific Office (SO) staff and DCC staff. However, the database also has the capability of producing standard datamarts. Instead of a centrally run analysis by the DCC, an investigator from a participating institution can submit their concept to the DSEC for approval as a datamart. If approved, a standard set of SAS data sets are created that contain raw data as well as derived data using standard algorithms programmed in SAS. The investigator at the institution is responsible for supporting the staff to analyze the datamart with close oversight from the SO and DCC.

The last type of report is the annual reporting used to provide feedback to the project Principal Investigators and NCCN Board Members. These packets are referred to as "benchmarking packets" and include two main sections: Patterns of Care (POC) and Concordance to Guidelines. The BCA POC portion of the report is divided into seven areas of interest: Patient Characteristics, Surgery, Radiation, Chemo and Hormone Therapy, Testing, Therapy for Metastatic Disease, and Surveillance. The concordance portion of the packet is divided up by guideline based on stage and contains results for DCIS, Stage I/II Axillary Staging, Stage I/II Radiation, Stage I/II Adjuvant Therapy, Stage III, Stage IV, and Metastatic Recurrence. In addition, there are also five summary sheets in the concordance section. A working example of the concordance section for the CRC packet will be discussed at WUSS by ter Veer (2009). The entire BCA packet is presented in Excel, and is a huge file with a total of 19 worksheets and 94 pages. The report format and SAS code used to generate the POC section of this report are the focus of this paper and will be expanded on in the following sections.

NCCN BENCHMARKING PACKETS

LAYOUT

As previously described, the benchmarking packets are just one of the mechanisms used to track data in the NCCN BCA Outcomes database. Specifically they are used to provide feedback and pattern tracking of the data to the NCCN Board members. Each center/institution considers its data highly confidential and as a result each institution has its own packet for a total of 27 packets to be produced on an annual basis. This coming year we are adding more centers, such that the total number of packets to be run will be 32. The entire packet is run using SAS and data are output to Excel in graphic and tabular form. Each variable in the report is reported as aggregate (all centers combined) and center specific. Additionally the numeric table contains center ranges (low to high percent) so that any center can compare itself not only to the aggregate rate, but also to the center ranges (Figure 1).

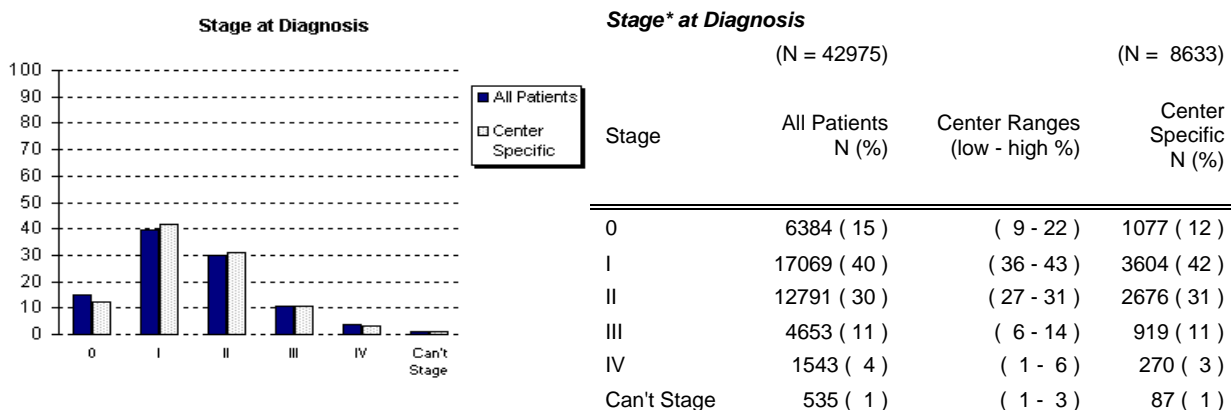


Figure 1: An example (fictitious data) of the layout for the variable stage

*Patients diagnosed prior to January 1, 2003 were staged according to the 5th edition of AJCC, and those diagnosed on or after January 1, 2003 according to the 6th edition of AJCC.

CODING

GENERAL VARIABLE CODING

The SAS coding for this the packet is complicated, but can be broken down into four main parts. First there is the code that reads the raw data from the database and manipulates it into relevant variables using algorithms developed by the SO and DCC. This programming calls on many macros such as the staging macro which combines T, N, and M stage that are collected clinically, by MD and pathologically to derive an overall stage for the breast cancer. This portion of packet programming is beyond the scope of this paper. The coding for the report itself is described in the following sections.

TABLE TOTALS CODING

The next portion of the packet coding takes the derived variables of interest and calculates the various statistics for the graph and tables. As this is a highly customized report there is no SAS procedure (PROC) that lends itself easily to the data that are required for the layout. Therefore the calculations for data in the graphs and tables, while not mathematically challenging, are a combination of PROC and DATA steps that can produce a data set with the desired formatting and relevant information. Additionally we've incorporated code to dump this output into Excel according to our layout. Because of the highly customized packet layout some of the techniques for getting data to excel such as ODS HTML or ODS XML with tagsets will not export the data to Excel in the layout that is required. Therefore the method we've chosen for export is DDE to Excel.

We start by "waking up" Excel using the simplest method suggest by Delwiche & Slaughter (2008) and opening a particular template file. Others have suggested failsafe methods if issues arise where SAS tries to move on before waiting for Excel to finish opening (Vyverman, 2001).

```
options noxsync noxwait; ❶  
X 'C:\files\WUSS2009\Example.xls'; ❷
```

❶ The NOXSYNC and NOXWAIT options allow us to return control to SAS without having to close the Excel process.

❷ The X command wakes up Excel and opens the specified file. Note that if you have embedded spaces in your filename or path you should include a set of double quotes *inside* the single quotes around the file and path name.

Next we calculate our statistics group by group, and output the information to the worksheet using DDE. The first set of statistics to be calculated are simple, the overall sample sizes that are found in the columns above 'All Patients' and 'Center Specific' headers on the table. Our programming here needs to calculate the sample sizes, format them as required for the report and then output this information to the specific location on the worksheet.

```
** Find Aggregate Count;  
proc means data=ex noprint; ❶  
var stage;  
output out=AggCnt N=AggCount; ❷  
run;  
  
filename AggN DDE  
"Excel|C:\files\WUSS2009\[Example.xls]Packet!R6C8:R6C8"; ❸  
  
** Makes N pretty and outputs to Excel;  
data _null_; set AggCnt; ❹  
countlabel=(N = '||put(AggCount,5.0)||'); ❺  
file AggN notab; ❻  
put countlabel '09'x; ❼  
run;  
  
** Find Center Count;  
proc means data=ex noprint;  
where cid=3; ❽  
var stage;  
output out=CenterCnt N=CenterCount;  
run;
```

```

** Filename ref to point to location in Excel file;
filename CenterN DDE
  "Excel|C:\files\WUSS2009\[Example.xls]Packet!R6C10:R6C10"; 9

** Makes N pretty and outputs to Excel;
data _null_; set CenterCnt;
countlabel='(N = '||put(CenterCount,5.0)||)';
file CenterN notab;
put countlabel '09'x;
run;

```

- 1 The NOPRINT option is used to keep the PROC MEANS from outputting to the listing window.
- 2 The purpose of this PROC is to calculate a number and output it to a data set as a variable called AggCount.
- 3 The FILENAME statement invokes the DDE 'triplet'. This consists of three important parts: which application SAS will be talking to, which file and worksheet, and what range of cells to write data to. Note that the DDE triplet can also be used to read data from Excel, and also to communicate with other applications such as Word®. The DDE triplet is very sensitive and if specified incorrectly will lead to an error: ERROR: Physical file does not exist. In our case we would like to communicate with the Excel file called Example.xls. We will output to the packet worksheet in one cell which is listed here as the range row 6 column 8 (use column numbers not letters as show on Excel) to row 6 column 8.
- 4 A DATA _null_ step is used as the purpose here is to communicate with Excel not to create another data set.
- 5 We modify the value of the aggregate sample size according to the format specifications of our layout. This involved adding parentheses and N = text to the value.
- 6 The FILE statement will call the FILENAME ref to link to Excel. Note that Excel uses tabs as a delimiter for each column, so to prevent Excel from translating data with embedded blanks into more than one column we need to add the notab option.
- 7 The PUT statement tells SAS what variable to write to the Excel file. Now we need to add the tabs back in between each variable to be output so we use the '09'x which is hexadecimal for tab.
- 8 We repeat a similar process for the center specific sample size, but this time we use a WHERE statement in the PROC means to only calculate N for that specific center.
- 9 A new FILENAME ref is created for the center specific data. While we are still pointing to the same file and worksheet we have changed the cell reference to output into a different location on the worksheet.

MAIN TABLE AND GRAPH CODING

The coding for the graphic and table are similar so we calculate them together. First we find the count and percentage of each stage category for the aggregate data, and then apply a little formatting to the data for the table. Then we use similar programming for the center specific data for stage. At this point we are ready to output the data for the plot which has been created ahead of time but remains empty waiting for data from the worksheet. Finally we calculate the center ranges and combine all the data for the table to be output.

```

** Find Aggregate Frequencies;
proc freq data=ex noprint;
table stage / out=AggFreq;
format stage stage.; 1
run;

** For Table;
data AggFreqClean; set AggFreq; 2
AggLabel=count||' ('||right(put(percent,3.))||)';
run;

```

```

** Find Center Frequencies;
proc freq data=ex noprint;
table cid*stage / out=CenterFreq outpct; ❸
format stage stage.;
run;

** For Table;
data CenterFreqClean; set CenterFreq;
where cid=3;
CenterLabel=count||' ('||right(put(pct_row,3.))||' )';
run;

** For output to plot;
filename Plot DDE
  "Excel|C:\files\WUSS2009\[Example.xls]Packet!R4C14:R9C16"; ❹

data plot; ❺
merge AggFreq (keep=stage percent)
      CenterFreq (keep=cid stage pct_row where=(cid=3));
by stage;
file Plot notab;
put stage '09'x percent '09'x pct_row '09'x; ❻
run;

** find ranges for each category;
proc means data=CenterFreq noprint;
var pct_row;
class stage;
output out=Ranges min=min max=max;
run;

data RangeClean; set Ranges;
where _type_=1; ❼
Range='('||put(min,3.))||' -'||right(put(max,3.))||' )';
run;

** Output to Excel;

filename Table DDE
  "Excel|C:\files\WUSS2009\[Example.xls]Packet!R8C7:R13C10";

** Put it all together;
data AllClean;
merge AggFreqClean (keep=stage AggLabel)
      CenterFreqClean (keep=stage CenterLabel)
      RangeClean (keep=stage Range); ❽
by stage;
file Table notab;
put stage '09'x AggLabel '09'x Range '09'x CenterLabel '09'x;
run;

```

- ❶ To calculate frequencies we use a PROC FREQ on the variable of interest and include user defined formats to give the appropriate labels for each category of our numeric variable stage.
- ❷ This time we will create a data set for the aggregate N (%) to be merged later with the center specific and ranges. Combining the data first this way and then using DDE to output all of it to the table at once saves processing time and generates a cleaner log file.
- ❸ The OUTPCT option is used to save the row and column percentages as well as the two-way table frequencies in the output data set. This way we have the N (%) for each center by stage combination.

- 4 We take a break from the table data for a moment to output the data for the plot that is ready to go. In the DDE triplet we have modified it to output to a range of cells. Notice that this range is to the far right of the plot and table. This allows us to output the data for the plot and then hide it by turning the text to white in these cells and specifying the print layout to the first 12 columns. Alternatives to this approach are to 'hide' the plot data under the plot itself in the first 3 columns of the worksheet, or the output can be dumped to a different 'working' worksheet and this worksheet can then be hidden.
- 5 The plot data set is a combination of the aggregate percentages and center specific percentages linked by stage category.
- 6 The FILE and PUT statements are combined directly into the data step with the merge. In the PUT statement we must include '09'x after each variable name to output variables to their own individual columns.
- 7 When using the OUTPUT statement in PROC MEANS to create a data set we get a variable included in the output data set called _TYPE_. When there is a class statement involved with this PROC we will get a different value of _TYPE_ for the overall min and max (_TYPE_=0), versus the min and max broken down by stage (_TYPE_=1). For this report we would like the min and max for each category of stage.
- 8 We merge the formatted table statistics into one data set in order to use only one DDE output call. The data from each data set are linked by the variable of interest, in this case stage.

X4ML CODING

The final portion of our code are the X4ML commands to modify the header and footers in Excel through SAS. Each header contains the center specific name and needs to be changed for each center's packet. Rather than type this in by hand across 19 worksheets and 27 packets (as we did for many years) we use SAS to communicate via the X4ML language. We also use this language to assist with the saving formatting and closing of files. To work with X4ML we need to invoke the DDE doublet (Vyverman, 2001) which links SAS to a certain application, in the case Excel, and allows us to send commands to Excel as if we were actually in that application clicking and pointing away.

```

filename TTEExcel dde 'excel|system'; 1

data _null_;
file TTEExcel; 2

** Activate Packet worksheet;
put '[workbook.activate("Packet")]'; 3

** Header;
put '[page.setup("&L&B&I&10 CONFIDENTIAL 4

&C&B&I&10 NCCN Breast Cancer Outcomes Project Benchmarking Packet:' '0d'x '
Patterns of Care Analyses' '0d'x '
For All Patients and Center Specific for City of Hope National Medical Center 5

&R&10 For Internal Use Only")]'; 6

** Footer;
put '[page.setup(,"&L&8 Patient Characteristics ' '0d'x ' ' 'A9'x' 2009
National Comprehensive Cancer Network

&C&8 (July 2004 - June 2008)

&R&8 &P")]'; 7

** change color of cells;
put '[select("R3C14:R9C16")]'; 8
put '[font.properties(,,,,,,2)]'; 9
put '[select("R1C1")]'; 10

put '[save.as("C:\files\WUSS2009\Example Center3.xls")]'; 11

put '[file.close(false)]'; 12

run;

```

- 1 The DDE doublet only requires an application name and does not point to a particular worksheet or cell range.
- 2 We invoke the DDE doublet in the data step and we then can pass Excel commands through SAS using the X4ML coding through PUT statements enclosing the X4ML functions in single quotes.
- 3 Workbook.Activate allows us to active the workbook specified just to be sure that we are on the worksheet that we intend to change. This can be added again at later points in the data step to move to other worksheets if further formatting is desired.
- 4 Page.Setup is a complicated function and has many parameters. It is the equivalent of choosing page setup from the file menu. Fortunately for us the header is the first parameter and footer is the second. Just typing directly into the first parameter to change the header and using format coding for formatting font and location of the text is all we need to deal with the header. The spacing in the PUT statement breaks the header up into left, center and right sections for easier digestion. &L&B&I&10 tells Excel in the left side of the header, bold/italic, size 10 font to put the word CONFIDENTIAL.
- 5 The next portion of the header is in the center section with bold/italic and size 10 font. The '0d'x is hexadecimal for a carriage return and must be enclosed in quotes.
- 6 The last portion of the header is in the right section with size 10 font.
- 7 The footer is the second parameter of the Page.Setup function so we delimit this with a leading comma. The first portion of the footer is in the left section in size 8 font. The 'A9'x gives us a © symbol. The second part of the footer is located in the center section with size 8 font. And finally the last part of the footer is in the right section in size 8 font and includes the automatic page numbers by denoting &P.
- 8 Recall that we stored the data for the plot off to the right side of the worksheet. The select function allows us to select the range of cells associated with this plot data.
- 9 Now that the plot range is selected we will use the Font.Properties function to change the color of the text to white. The parameter for color is in the 10th position and the code for white is 2. Note that all subsequent parameters can be ignored as they are not changing.
- 10 Just adding a quick move of the active cell back to the first row first column.
- 11 The Save.As function allows us to now save the template with filled in and formatted data as a new filename. This will not overwrite our original template so that we can rerun the programming for the next NCCN center and format it appropriately.
- 12 Finally the File.Close function closes the file and passing the false parameter tells Excel not to try saving again.

RUNNING THE PACKET

WHY EXCEL?

We've seen the layout of the packet in Excel - what are some other reasons that we chose to go with the Excel layout? First there are the strict layout specifications with graphs and tables showing aggregate versus center specific data. Next there is the issue of the flexibility of manipulation of the template by staff who are not always SAS users. The report changes year- to-year with addition or deletion of variables in the POC section. Additionally the guidelines are updated several times during the year therefore the layout for the concordance section is constantly changing in the packet. Using Excel allows our clinical support staff to make any changes to the layout of the report and then the statistical staff will fill in the relevant data via SAS. Additionally after the report is run it can be further manipulated, if necessary, and finally printed by administrative staff to pdf for final distribution.

The idea behind using an Excel template for the reports is that the content changes each year; therefore it will be easiest to make the changes in one place and then run the SAS programming. This structure allows for the data manipulation to be run by a SAS program and output to the template for each center and then save the Excel file as a different name as to not overwrite the original template file. Because we have 27 packets to run (growing to 32 packets in the coming year) with 94 pages each, we have found that it is easiest to output the aggregate and institution range data to the template first, then save this as a replacement to the template as it is the same across all packets. Then we output each institutions data and save these as the final packets.

INCORPORATION OF MACROS

Much of this programming is achieved through SAS macros. The macro specifies what variable is being output and its placement on the worksheet. This is a lot of typing, especially with the annual changes that take place each year, so running it all as a one large macro makes the most sense to cut down on all that programming. Without repeating all of the parts in the code above here we simply wrap the code into a macro and replace some of the SAS variable names and references into macro variables. The final macro and macro call are represented here:

```
options noxsync noxwait;
X 'C:\files\WUSS2009\Example.xls';

%macro VarLevel(dsn,var,fmt,N,PlotS,PlotE,TabS,TabE); ❶

** Repeat of code discussed in previous sections with replacements for macro
variables listed in tables below;

%mend;

%VarLevel(ex,stage,stage.,6,4,9,8,13);
```

❶ The replacements in the code from the previous section are as following:

Code	Macro Variable Replacement
ex	&dsn
stage	&var
stage.	&fmt

Additionally the cells locations have to be reset for each additional variable to the report. Because the columns remain consistent we only need to replace the rows. The sample size columns point to only one cell so they can be shared for aggregate and center specific. The other cell ranges need a start row and end row. Also note that SAS may get confused by these macro variable names, because it does not know where the macro variable name ends. Therefore we use the . at the end of the macro variable name to tell SAS where this name ends.

FILENAME Ref	Cell Coding	Macro Variable Replacement
AggN and CenterN	6	&N.
Plot	4	&PlotS.
	9	&PlotE.
Table	8	&TableS.
	13	&TableE.

Eventually we also worked in the flexibility to call in a macro variable for which center is being run in the 27 packets. The final feature of the macro is that we can share it across disease sites. Each of the five disease sites has its own packet, so using a macro for the dumping portion of the programming allows it to be shared.

PROS AND CONS OF OUR APPROACH

DDE

The argument between SAS programmers over DDE and X4ML compared to other output to Excel methods rages on (Derby 2008). DDE advocates have created modifiable SAS macros that assist in exporting data to Excel via DDE (Vyverman 2001, Derby 2008). We've found that using DDE with our own macro to output to Excel has several advantages and disadvantages for our packets and programming. On the plus side it can handle the most customized version of any report and accommodate changes in the layout as requested by investigators. Additionally we can tell SAS exactly where to output the results for each variable, and we can incorporate specific headers/footers and footnotes within the packet. Once the Excel template is established it is easy to run the programming just by changing the macro variables values as appropriate. However using DDE is not fool-proof and there are also some significant disadvantages. The programming is very complicated and took many years to develop. The worst part of using the DDE method is that when there is a change in the packet, say an addition or deletion of a table and graphic, all of the subsequent cell locations and therefore programming has to be changed. As our macro points to each cell,

once rows are removed or added all others have to be updated. We have found some ways to work around this, such as hiding rows or columns for items that are temporarily deleted. However permanent deletions or new additions require extra programming time just to tell SAS where to output to the report. As a result, our macro and template method become that much more important for the production of this packet.

X4ML

As the “icing on the cake”, the X4ML programming calls also have advantages (Watts2005) and disadvantages. The X4ML language is very handy in that it allows you to break down coding into specific purposes such as formatting cells, saving and closing files, etc...It also allows us to work around some things that we would be unable to achieve with DDE alone, such as changing the header information in the report. The X4ML calls can be employed interactively through a DATA step, whereas using ODS directives are communicated all at once. Additionally if you enjoy decrypting programming syntax, you’ll have a blast figuring out X4ML! Watts (2005) discusses much of this as well as the unsupported help menu for X4ML macro functions macrofun.hlp which at this time can be found at <http://support.microsoft.com/kb/128185>. Disadvantages include the fact that X4ML was discontinued and replaced by VBA, and therefore is currently unsupported and could eventually disappear. It can also seem very complicated to use until one becomes familiar with its syntax.

CONCLUSIONS

Overall because of the customized formatting of this benchmarking packet the other techniques such as ODS HTML, ODS XML with tagsets, PROC EXPORT, and csv files are simply not possible. With these methods the flexibility for project staff to manage the template directly as they would like would be lost. The additional programming in getting DDE to work with our Excel layout, plus the macros that we’ve developed, are well worth the effort in order to achieve our desired reports.

REFERENCES

- Delwiche, L. and Slaughter, S. (2008), The Little SAS Book: A Primer, Fourth Edition, SAS Institute Inc., Cary, NC
- Derby, N. (2008), Revisiting DDE: An Updated Macro for Exporting SAS Data into Custom-Formatted Excel Spreadsheets, Proceedings of the SAS Global Forum Conference, paper 259-2008.
- Kilburn, J. et al (2009), An Example of Using Differing Methods to Create Excel Output from SAS, Proceedings of the Western Users of SAS Software Conference
- ter Veer, A. et al (2009), Reporting Quality of Cancer Care for the National Comprehensive Cancer Network – A SAS Bridge to Excel, Proceedings of the Western Users of SAS Software Conference
- Vyverman, K. (2001), Using dynamic data exchange to export your SAS data to MS Excel - Against all ODS, Part I, Proceedings of the Twenty-Sixth SAS Users Group International Conference, paper 011-26.
- Watts, P. (2005), Using single-purpose SAS macros to format Excel spreadsheets with DDE, Proceedings of the Thirtieth SAS Users Group International Conference, paper 089-30.

CONTACT INFORMATION

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

Rebecca Ottesen
City of Hope
Department of Biostatistics
1500 East Duarte Rd.
Duarte, Ca, 91010
(805) 594-0441
rottesen@coh.org

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.