PharmaSUG 2017 – Paper AD01

Here Comes The Smart Mock Table: A Novel Way of Creating Clinical Summary Tables Without Any Table Programming (No Kidding!)

Joseph Hinson, inVentiv Health, Princeton, NJ, USA

ABSTRACT

The creation of a clinical summary table typically involves two main phases: the statistical analyses of data, and the presentation of the analyses results onto a layout predefined by a mock table. The first phase can be pretty straightforward, simply involving the calling of SAS statistical procedures plus a few DATA steps. The second phase, a relatively time-consuming part, would constitute taking those procedure outputs and programming their placements in specified positions to become the summary table. It turns out that this second phase might not even be necessary. The mock table can be made to populate itself with analyses results! Such a "smart" mock table can be made by embedding macro calls directly in the RTF mock document. These macros would contain SAS procedures or DATA step codes wrapped inside a DOSUBL function and called by a %SYSFUNC to generate single macro variables. Finally, the entire smart RTF mock table, can be placed on the SAS Input Stack as a %INCLUDE for macro processing using the new SAS 9.4 STREAM Procedure. Such an approach could potentially allow programmers to focus entirely on data analyses, significantly shortening turnaround times for deliverables. Table cosmetic changes could be done at the mock table level with no need for reprogramming. Data point repositioning could also be implemented directly on the mock table by simply relocating the macro calls. The technique in fact is applicable to any WORD document, including the CONSORT Flow Diagram (see author's other paper).

INTRODUCTION

We live in interesting times: smart light bulbs, smart baby diapers, smart garbage cans, and even smart coffee-stirrers. So it should come as no surprise that we should now have smart mock tables -- not with sensors and WiFi-chips, but loaded with macro variables and even macro calls. It turns out that when such a macro-embedded mock table is processed by Proc STREAM, the table magically comes to life, displaying real analysis results at the desired positions, with not a single line of table programming code ever written. Move over, Proc REPORT!

The life of a clinical programmer is easily dominated by two phases of activities: writing code for data analyses, and figuring out programmatically how to place the results of those analyses onto a table, exactly as specified by a mock table. What if one could simply let the mock table process itself, populating the data placeholders ("xx.x") with real numbers from the analyses results?

Three SAS tools actually make possible such self-processing by a mock table:

(a) The DOSUBL function,
(b) The %SYSFUNC macro keyword, and
(c) The STREAM procedure.

The idea is to let a macro generate each data point so that the mock table would be populated with macro calls rather than "xx.x" placeholders, then place the macro-embedded mock table onto the SAS Input Stack for macro processing using a %INCLUDE.

However, under normal circumstances, placing an RTF document onto the SAS Input Stack (via %include) would generate plenty of syntax errors (see further down). The RTF codes would not be recognized as valid SAS syntax.

THE STREAM PROCEDURE

The general idea of adding macro elements to documents for macro resolution would be fairly easy if one could engage only the macro processor. As the document is tokenized, macro elements are diverted to the macro processor. The resolved values are returned and the documents are streamed back in original form except with resolved macro values. The SAS 9.4 Proc STREAM was developed for exactly that purpose. With that new procedure, almost anything can be put on the Input Stack (with the exception of binary formats like PDF, JPEG, DOCX). Proc STREAM disables the SAS Compiler thereby permitting the processing of the macro elements in a RTF documents in the presence of the SAS syntax-violating RTF codes. After macro resolutions, the RTF document is streamed to an external file location with the original structure and non-macro contents intact. Without Proc STREAM, the log would display syntax errors from the RTF codes, as shown below for a simple RTF table:

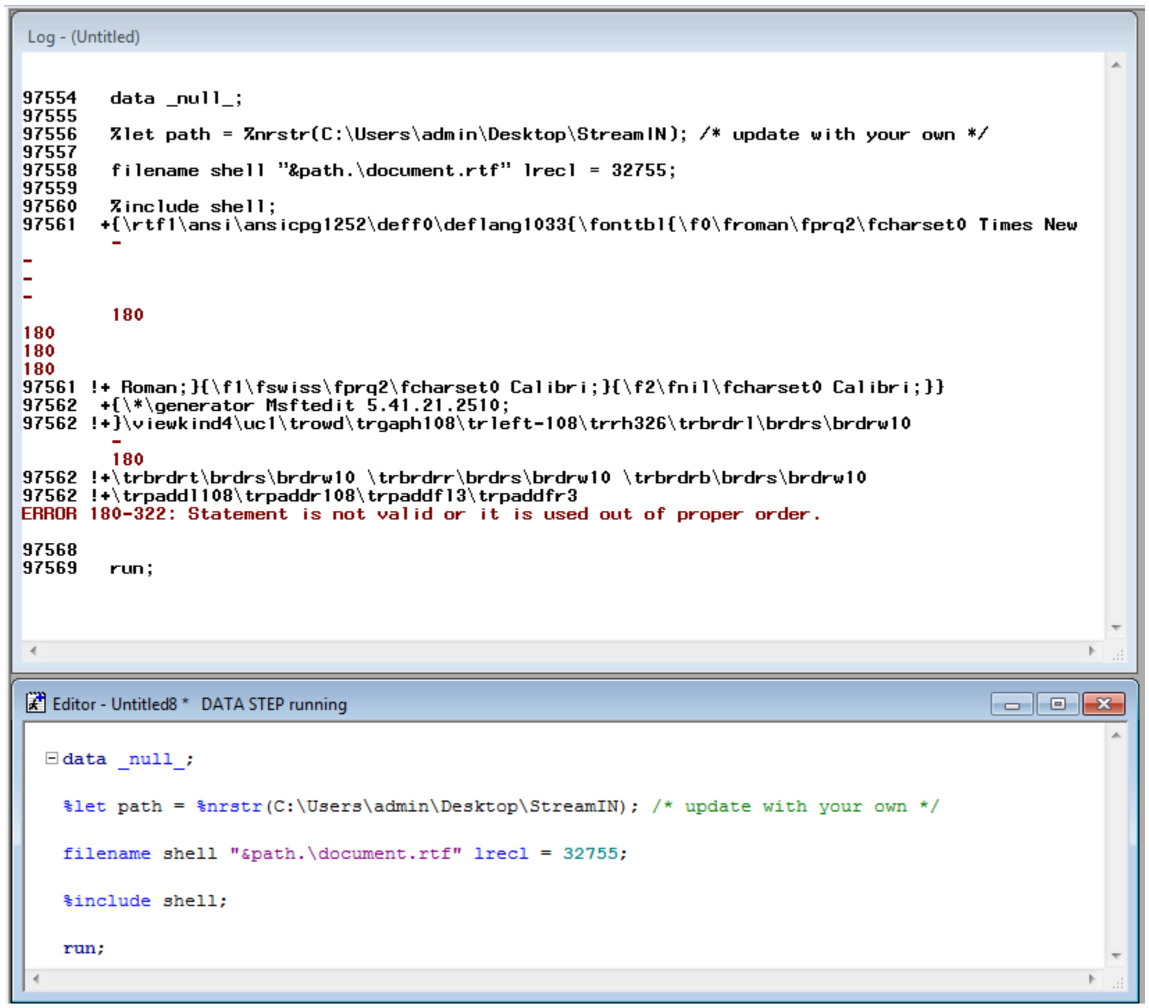
(a) RTF Table (document.rtf):

ACTIVE	PLACEBO
x.xx	x.xx
xx.x	xx.x

(b) Underlying RTF Code:

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\froman\fprq2\fcharset0 Times New Roman;}{\f1\fswiss\fprq2\fcharset0
Calibri;}{\f2\fnil\fcharset0 Calibri;}}

{\*\generator Msftedit 5.41.21.2510;}\viewkind4\uc1\trowd\trgaph108\trleft-108\trrh326\trbrdrl\brdrs\brdrw10 \trbrdrt\brdrs\brdrw10
\trbrdrr\brdrs\brdrw10 \trbrdrb\brdrs\brdrw10 \trpaddl108\trpaddr108\trpaddfl3\trpaddfr3

\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs
\cellx1503\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs \cellx3114\pard\intbl\f1\fs22
ACTIVE\cell PLACEBO\cell\row\trowd\trgaph108\trleft-108\trrh326\trbrdrl\brdrs\brdrw10 \trbrdrt\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10
\trbrdrb\brdrs\brdrw10 \trpaddl108\trpaddr108\trpaddfl3\trpaddfr3

\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs
\cellx1503\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs \cellx3114\pard\intbl x.xx\cell
x.xx\cell\row\trowd\trgaph108\trleft-108\trrh326\trbrdrl\brdrs\brdrw10 \trbrdrt\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10
\trbrdrb\brdrs\brdrw10 \trpaddl108\trpaddr108\trpaddfl3\trpaddfr3

\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs
\cellx1503\clbrdrl\brdrw10\brdrs\clbrdrt\brdrw10\brdrs\clbrdrr\brdrw10\brdrs\clbrdrb\brdrw10\brdrs \cellx3114\pard\intbl xx.x\cell
xx.x\cell\row\pard\sa200\sl276\slmult1\lang9\f2\par

}
```

(a) SAS Code and Log:



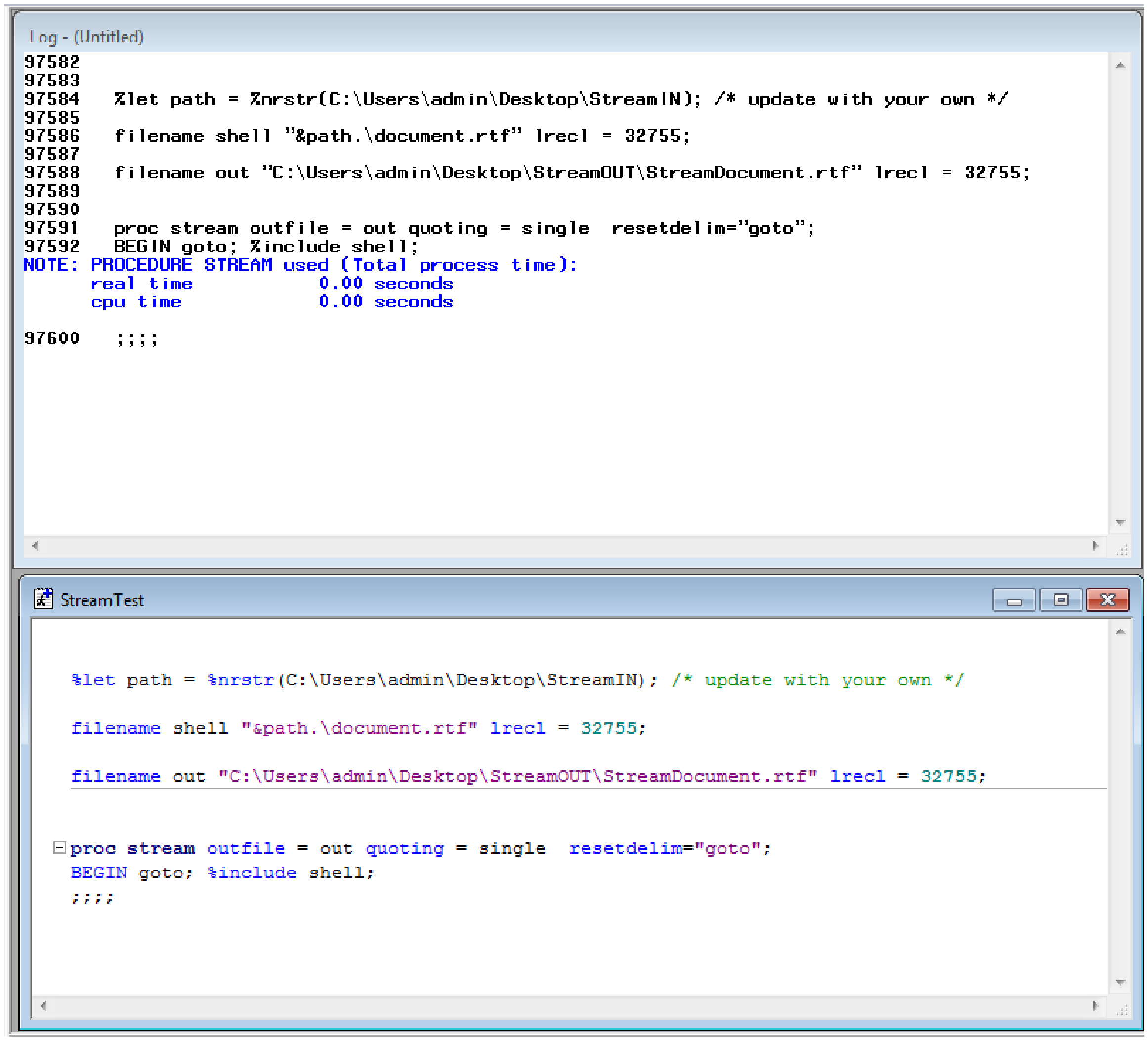
```
97554    data _null_;
97555
97556    %let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */
97557
97558    filename shell "&path.\document.rtf" lrecl = 32755;
97559
97560    %include shell;
97561   +{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\froman\fprq2\fcharset0 Times New
         -
-
-
-
         180
180
180
180
97561  !+ Roman;}{\f1\fswiss\fprq2\fcharset0 Calibri;}{\f2\fnil\fcharset0 Calibri;}}
97562   +{\*\generator Msftedit 5.41.21.2510;
97562  !+}\viewkind4\uc1\trowd\trgaph108\trleft-108\trrh326\trbrdrl\brdrs\brdrw10
         -
         180
97562  !+\trbrdrt\brdrs\brdrw10 \trbrdrr\brdrs\brdrw10 \trbrdrb\brdrs\brdrw10
97562  !+\trpaddl108\trpaddr108\trpaddf13\trpaddfr3
ERROR 180-322: Statement is not valid or it is used out of proper order.

97568
97569    run;
```

```
data _null_;

  %let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */

  filename shell "&path.\document.rtf" lrecl = 32755;

  %include shell;

  run;
```

The RTF codes trigger errors and the DATA step get stuck in an endless loop ("DATA STEP running")

(b) The Proc STREAM Solution:

When the same RTF file is run with Proc STREAM, the log becomes error-free:


Log - (Untitled)

```
97582
97583
97584   %let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */
97585
97586   filename shell "&path.\document.rtf" lrecl = 32755;
97587
97588   filename out "C:\Users\admin\Desktop\StreamOUT\StreamDocument.rtf" lrecl = 32755;
97589
97590
97591   proc stream outfile = out quoting = single  resetdelim="goto";
97592   BEGIN goto; %include shell;
NOTE: PROCEDURE STREAM used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

97600   ;;;;
```

StreamTest

```
%let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */

filename shell "&path.\document.rtf" lrecl = 32755;

filename out "C:\Users\admin\Desktop\StreamOUT\StreamDocument.rtf" lrecl = 32755;


proc stream outfile = out quoting = single  resetdelim="goto";
BEGIN goto; %include shell;
;;;;
```

THE PROC STREAM SYNTAX

The Proc STREAM statement specifies an external file with the “OUTFILE=” keyword, as well as options.

The arbitrary text is wrapped inside a “BEGIN” and a four semi-colon ending “;;;;”.

There is no “RUN” or “QUIT”.

PROC STREAM OUTFILE=*fileref <options>*;

BEGIN

(some text which may contain macro triggers)

;;;;

Two Useful Proc STREAM statement Options:

a. **RESETDELIM=** ”label”

The SAS® Word Scanner expects macro statements like “%LET” and “%INCLUDE” to begin on a statement boundary -- which means, the statements must be preceded by a semicolon and must end with a semicolon. Therefore to use %LET and %INCLUDE in a Proc STREAM input text, one must place a special marker token before the statements and end the statement with a semicolon. This special marker token is defined with the option RESETDELIM=*label*, where *label* can be any arbitrary SAS® name:

```
PROC STREAM  OUTFILE= myfile  RESETDELIM=”goto”;

BEGIN

goto; %INCLUDE myotherfile;

;;;;
```

The marker token specified by RESETDELIM is also required when a carriage return is required in

```
PROC STREAM  OUTFILE= myfile  RESETDELIM=”goto”;

BEGIN

Dear Sir, goto NEWLINE;
The profile below is for patient 12345.

;;;;
```

the input text. The keyword “NEWLINE” is used with the marker token:

b. **QUOTING=SINGLE** *(or* **DOUBLE** *or* **BOTH***)*

This option specifies that the single quotation mark (‘) should be treated like any other character, as expected for: *the patient’s blood pressure*.

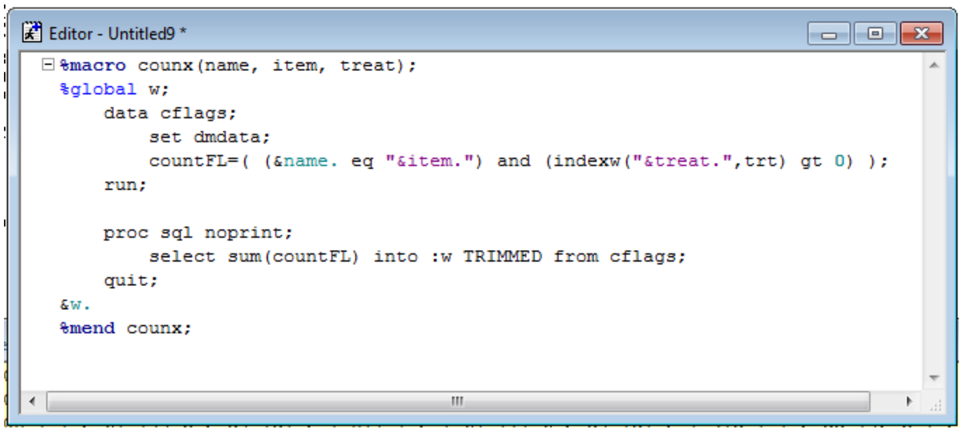
QUOTING=DOUBLE would be required in cases where the text involves, for instance, XML elements in Define-XML documents:

```
<ItemRef ItemOID="ADSL.ARM" OrderNumber="6" Mandatory="No"/>
```

So with Proc STREAM, a macro-embedded RTF document can be placed on the SAS Input Stack without any triggering of errors.

THE PROBLEM WITH MACRO CALLS

Thus, Proc STREAM can allow a macro-embedded RTF document to be processed by the Macro Processor. But there is yet another problem. Processing documents with macro variables is very straight forward and not that different from the everyday text substitution with macro variables done in SAS programming. Putting MACRO CALLS in documents presents another challenge, especially if those macros contain DATA steps and/or procedures. Such macros would be resolved by the Macro Processor into the actual constituent DATA step or procedure codes, and placed back on the Input Stack for streaming out. An example of such macros is shown below:



```
%macro counx(name, item, treat);
%global w;
    data cflags;
        set dmdata;
        countFL=( (&name. eq "&item.") and (indexw("&treat.",trt) gt 0) );
    run;

    proc sql noprint;
        select sum(countFL) into :w TRIMMED from cflags;
    quit;
&w.
%mend counx;
```

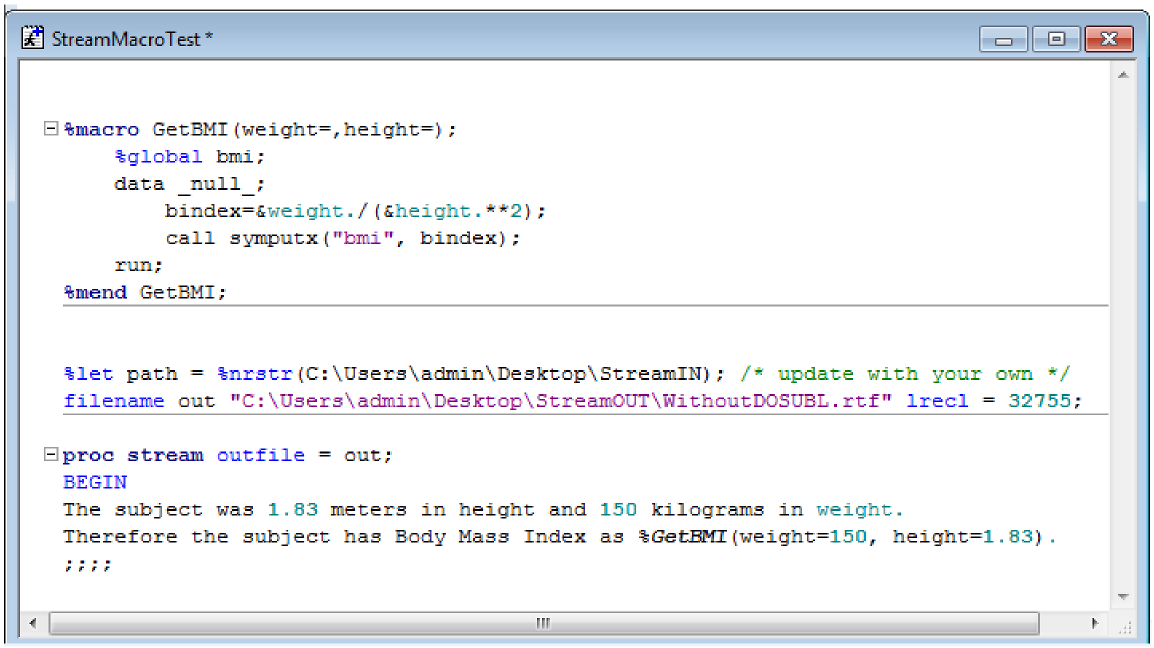
We desire the analysis macro to generate a single value.

However when %counx() is called, the macro processor would return the generated code instead of the computed value.

THE NEED FOR DOSUBL() AND %SYSFUNC

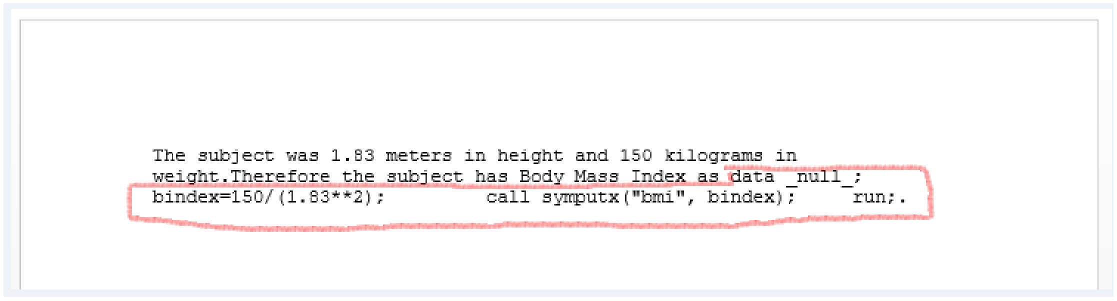
The DOSUBL function enables the immediate execution of SAS code after a text string is passed. If the text is a DATA step or a PROC program, DOSUBL would allow them to execute and wait for them to complete. Also any macro variables that are created or updated during the execution of the submitted code are exported back to the calling environment. DOSUBL does not return until the execution of the SAS code is fully completed.
If DOSUBL() is called by a macro via %SYSFUNC, then the macro is forced to execute the DATA step and Proc SQL codes and any macro variable created inside the DOSUBL becomes available, instead of the macro execution returning the raw lines of DATA step or Proc codes to the input stack. %SYSFUNC forces execution of functions it carries as arguments. Also for proper timing of resolution for macro variables enclosed in quotes, the outermost quotes inside the DOSUBL function should be single ***(this is important!)***. Below is an example with a macro that calculates the Body Mass Index of a subject:

(a) Without DOSUBL:



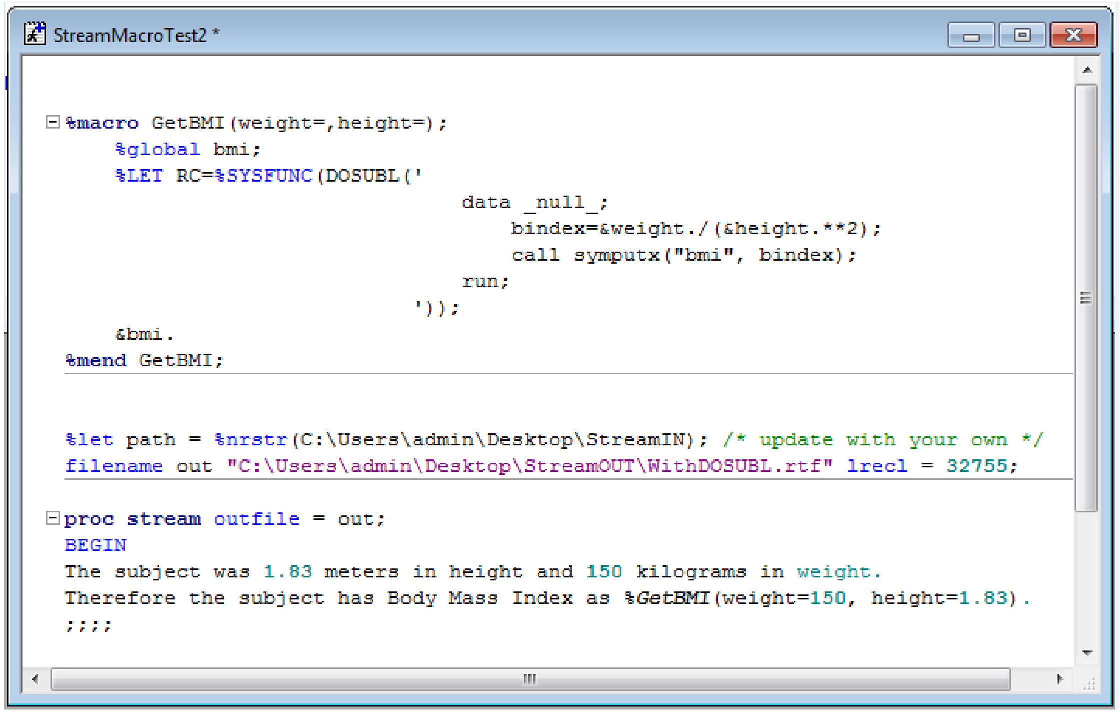
```
%macro GetBMI(weight=,height=);
    %global bmi;
    data _null_;
        bindex=&weight./(&height.**2);
        call symputx("bmi", bindex);
    run;
%mend GetBMI;


%let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */
filename out "C:\Users\admin\Desktop\StreamOUT\WithoutDOSUBL.rtf" lrecl = 32755;

proc stream outfile = out;
BEGIN
The subject was 1.83 meters in height and 150 kilograms in weight.
Therefore the subject has Body Mass Index as %GetBMI(weight=150, height=1.83).
;;;;
```

The streamed output shows the DATA step code instead of the calculated BMI value:



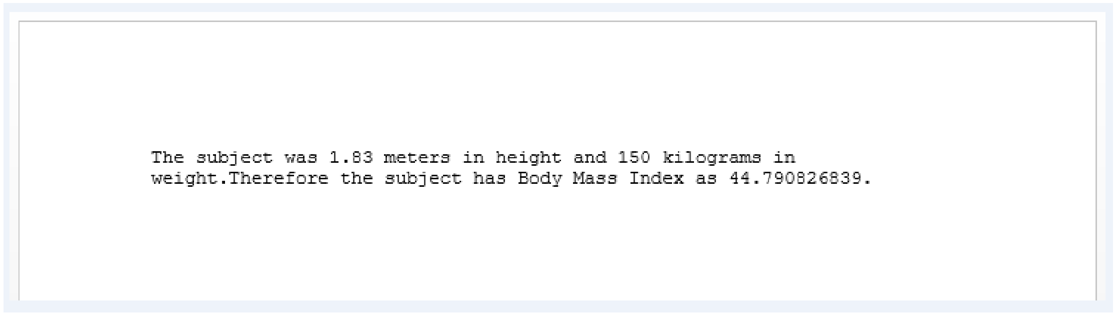
```
The subject was 1.83 meters in height and 150 kilograms in
weight.Therefore the subject has Body Mass Index as data _null_;
bindex=150/(1.83**2);          call symputx("bmi", bindex);     run;.
```

(b) With DOSUBL:



```
%macro GetBMI(weight=,height=);
    %global bmi;
    %LET RC=%SYSFUNC(DOSUBL('
                            data _null_;
                                bindex=&weight./(&height.**2);
                                call symputx("bmi", bindex);
                            run;
                        '));
    &bmi.
%mend GetBMI;


%let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own */
filename out "C:\Users\admin\Desktop\StreamOUT\WithDOSUBL.rtf" lrecl = 32755;

proc stream outfile = out;
BEGIN
The subject was 1.83 meters in height and 150 kilograms in weight.
Therefore the subject has Body Mass Index as %GetBMI(weight=150, height=1.83).
;;;;
```

Now, the DATA step code is fully resolved into a value:



```
The subject was 1.83 meters in height and 150 kilograms in
weight.Therefore the subject has Body Mass Index as 44.790826839.
```

OVERALL STRATEGY

1. Create analysis macros according to the original mock specifications, wrapping any DATA step or procedure inside a DOSUBL and invoking with a %SYSFUNC. Make sure each analysis macro generates a macro variable (“macro call variable”).
2. Create Regular Mock Table:
   (a) Open WORD (eg. Office 2010 or 2013).
   (b) Invoke FILE → NEW.
   (c) Do INSERT TABLE (4 by 20 rows, for example).
   (d) Change layout to LANDSCAPE (for example).
   (e) Enter column headers, row labels, and data placeholders (xx.xx, etc) and save as RTF for step 3.
   (f) Or, you may download a free WORD table template, change title, headers and labels, remove unwanted rows and columns, enter xx.xx’s, and save as RTF for step 3.
   (g) Or, if there is already an original mock table as a WORD document, save as RTF for step 3.
3. Reopen the RTF mock from step 2 with WORD. ***(This step is essential!)***
4. Type macro calls in the expected locations for analysis results (replace the “xxx.xx” on original mock), thereby converting the RTF mock table to a smart document.
5. Save the smart RTF mock (SmartMock.rtf).
6. Write Proc STREAM code with %include SmartMock.rtf and output file as Deliverable.rtf
7. Run main program to define analysis macros and to execute Proc STREAM.

AN EXAMPLE WITH THE DEMOGRAPHICS SUMMARY TABLE (SIMPLE LAYOUT)

1. **CREATE ANALYSIS MACROS ACCORDING TO THE ORIGINAL MOCK SPECIFICATIONS:**

(a) Read Input Dataset:

```
data dmdata (drop=code:);
infile datalines;
input SUBJID codeTRT codeGEND codeRACE Age Weight Height codeRELIGION Waist
Iriscan codeBLOOD codeCOUNTRY @@;
BMI=703*(Weight/((Height*12)**2));*<--if WEIGHT in pounds and HEIGHT in feet;
*----DECODE VALUES FOR TREATMENT, RACE, GENDER, etc. -------;
trt=choosec(((codeTRT)+1), "ACTIVE","PLACEBO");
Gender=choosec(codeGEND,"Male","Female");
Race=choosec(codeRACE, "White", "Black", "Other");
Blood=choosec(codeBLOOD,"A","B","AB","O");
Religion=choosec(codeRELIGION,"Buddhist","Catholic","Muslim","Jewish");
```

```
Country=choosec(codeCOUNTRY,"American","Jamaican","Chinese","British");
datalines;
101 0 1 3 37 112 6 2 34 131 2 1 301 0 1 1 70 131 6 3 34 131 4 1 501 0 1 2 33 111 6 2 34 131 2 1
601 0 1 1 50 124 5 1 34 131 3 2 701 1 1 1 60 121 6 1 34 131 3 2 102 1 2 1 65 123 5 3 40 188 4 2
302 0 1 2 55 143 6 2 34 131 4 3 502 1 2 1 44 181 6 2 34 131 4 3 602 0 2 2 30 152 7 2 33 129 3 4
702 0 1 1 28 133 4 3 34 131 1 4 103 1 1 2 32 132 6 4 34 131 2 4 303 1 1 1 65 122 7 1 38 200 3 3
503 1 1 1 64 133 6 4 34 131 2 2 603 1 2 1 33 133 6 2 34 131 1 1 703 1 1 2 44 126 4 4 43 117 1 2
104 0 2 3 23 122 5 1 34 131 3 1 304 0 1 1 45 134 6 3 34 131 2 2 504 0 1 3 56 122 6 4 37 166 1 3
604 0 1 3 65 125 4 2 34 131 4 4 704 0 2 1 66 171 6 4 34 131 3 3 105 1 1 3 44 121 5 3 31 212 2 2
305 1 1 1 36 113 5 3 34 131 1 2 505 1 1 2 73 143 6 2 34 131 4 4 605 1 2 1 57 122 4 1 41 154 2 2
705 1 1 2 46 124 5 4 34 131 3 4 106 0 2 1 49 144 6 4 34 131 3 1 306 0 1 2 46 143 5 3 36 123 1 2
506 0 1 1 46 122 5 4 34 131 2 3 606 0 1 2 56 133 6 4 34 131 3 3 706 1 1 1 75 143 5 2 33 158 3 2
201 1 1 3 35 134 6 3 34 131 4 1 401 1 2 1 44 111 6 2 34 131 2 3 507 1 1 2 44 112 5 4 38 144 4 4
607 1 1 1 67 143 5 2 34 131 2 3 707 1 1 1 46 115 6 1 34 131 4 4 202 0 2 1 50 154 6 2 31 155 3 1
402 0 2 2 77 122 4 1 34 131 4 4 508 0 2 1 53 141 6 2 34 131 2 3 608 0 2 2 46 142 5 4 40 100 2 2
708 0 2 1 55 131 4 2 34 131 1 1 203 1 1 2 49 122 6 4 34 131 3 2 403 1 1 1 45 133 5 2 33 188 3 4
509 0 1 3 45 132 5 2 34 131 2 2 609 1 2 1 72 116 6 2 34 131 4 1 709 0 2 2 57 132 6 4 32 163 4 4
204 0 2 3 60 113 4 2 34 131 3 3 404 1 1 1 59 131 6 2 34 131 1 4 510 0 1 3 65 113 5 4 39 132 1 4
610 0 1 1 29 118 6 4 34 131 4 4 710 0 1 1 63 165 6 4 34 131 3 1 205 1 1 3 39 132 5 1 31 155 2 4
405 0 2 1 49 124 5 1 34 131 1 1 511 1 2 2 43 123 6 4 34 131 2 4 611 1 2 1 65 125 6 3 33 121 3 4
711 1 1 2 61 144 5 4 34 131 2 1 206 1 2 1 67 111 6 2 34 131 2 1 406 1 1 2 33 142 5 4 29 153 4 2
512 1 1 1 39 111 4 2 34 131 2 1 612 1 1 2 46 132 6 2 34 131 2 1 712 0 1 1 49 121 5 2 42 193 1 2
801 1 1 1 67 143 5 4 34 131 2 1 810 1 1 1 46 115 6 1 34 131 2 1 902 0 2 1 50 154 6 4 31 155 3 1
802 0 2 2 77 122 4 3 34 131 2 1 811 0 2 1 53 141 6 2 34 131 2 1 903 0 2 2 46 142 5 4 40 100 2 2
803 0 2 1 55 131 4 4 34 131 2 1 812 1 1 2 49 122 6 4 34 131 2 1 904 1 1 1 45 133 5 1 33 188 3 4
804 0 1 3 45 132 5 4 34 131 2 1 813 1 2 1 72 116 6 2 34 131 2 1 905 0 2 2 57 132 6 4 32 163 4 4
805 0 2 3 60 113 4 2 34 131 2 1 814 1 1 1 59 131 6 1 34 131 2 1 910 0 1 3 65 113 5 3 39 132 1 4
806 0 1 1 29 118 6 4 34 131 2 1 815 0 1 1 63 165 6 2 34 131 2 1 911 1 1 3 39 132 5 2 31 155 2 4
807 0 2 1 49 124 4 4 34 131 2 1 816 1 2 2 43 123 6 3 34 131 2 1 912 1 2 1 65 125 6 3 33 121 3 4
808 1 1 2 61 144 5 2 34 131 2 1 817 1 2 1 67 111 6 2 34 131 2 1 913 1 1 2 33 142 5 1 29 153 4 2
809 1 1 1 39 111 5 2 34 131 2 1 818 1 1 2 46 132 6 4 34 131 2 1 917 0 1 1 49 121 5 2 42 193 1 2
;
run;
```

(b) Define Macros for Big N's:

```
%macro bn(gp);
%let t=%sysfunc(dosubl('proc sql noprint;select count(distinct subjid) into
:bgn TRIMMED from dmdata where indexw("&gp.",trt) gt 0;quit;'));
&bgn.
%mend bn;
```

(c) Define Macros for Counts:

```
%macro counx(name, item, treat);
%global w;
%let u1=%sysfunc(dosubl('data cflags;set dmdata;
countFL=( (&name. eq "&item.") and (indexw("&treat.",trt) gt 0) );run;'));
%let u2=%sysfunc(dosubl('proc sql noprint;select sum(countFL)
      into :w TRIMMED from cflags;quit;'));
&w.
%mend counx;
```

(d) Define Macros for Descriptive Statistics:

```
%macro stax (name, test, form, treat);
%global res;
%let st1=%sysfunc(dosubl('proc sql noprint;select &test.(&name.)
format=&form.
      into :res TRIMMED from dmdata where indexw("&treat.",trt) gt
0;quit;'));
&res.
%mend stax;
```

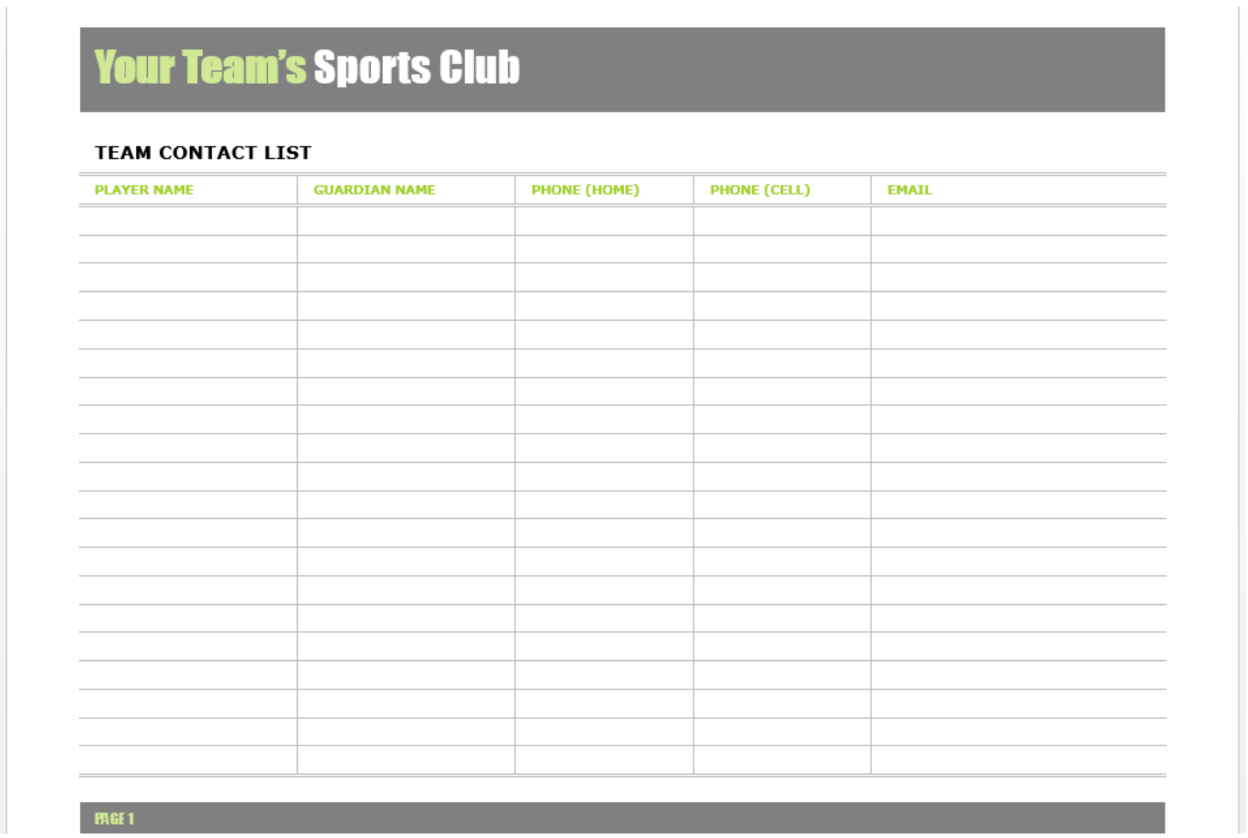
2. CREATE AN ORIGINAL RTF MOCK TABLE BY ONE OF THE FOLLOWING RELIABLE METHODS:

i. Save a pre-existing MS WORD mock table as RTF, or

ii. Insert a new table into MS WORD and fill in with the titles, footnotes, headings, row-labels, and data placeholders (xx.xx), then save as RTF, or

iii. Download a free MS WORD table template, and edit out the row and column labels and fill in the mock table info, then save as RTF.

Using method iii, a free template called "Contact list for youth sports" (WORD 2013) was downloaded as shown below:



Your Team's Sports Club

TEAM CONTACT LIST

PLAYER NAME	GUARDIAN NAME	PHONE (HOME)	PHONE (CELL)	EMAIL

PAGE 1

The above template then had unwanted rows and columns removed and mock table information typed in as shown below:

TABLE 1.2.3.4 SUMMARY OF DEMOGRAPHIC CHARACTERISTICS

	ACTIVE	PLACEBO	TOTAL
Number of Subjects	xxx	xxx	xxx
Age (Years)			
N	xxx	xxx	xxx
Mean (SD)	xx.x (x.xx)	xx.x (x.xx)	xx.x (x.xx)
Range	xx	xx	xx
Median	xx	xx	xx
Min – Max	xx - xx	xx - xx	xx - xx
Gender			
Female	xx	xx	xx
Male	xx	xx	xx
Race			
White	xx	xx	xx
Black	xx	xx	xx
Asian	xx	xx	xx
Other	xx	xx	xx

The modified WORD template above was ***then saved as RTF, reopened with WORD***, and converted to a "SmartMock" by typing in the macro calls, as shown below (SmartMock.rtf):

TABLE 1.2.3.4 SUMMARY OF DEMOGRAPHIC CHARACTERISTICS

	ACTIVE	PLACEBO	TOTAL
Number of Subjects	%bn(ACTIVE)	%bn(PLACEBO)	%bn(ACTIVE PLACEBO)

Age (Years)			
N	%stax(age,N,3.,ACTIVE)	%stax(age,N,3.,PLACEBO)	%stax(age,N,3.,ACTIVE PLACEBO)
Mean (SD)	%stax(age,Mean,5.1,ACTIVE) (%stax(age,STD,5.2,ACTIVE))	%stax(age,Mean,5.1,PLACEBO) (%stax(age,STD,5.2,PLACEBO))	%stax(age,Mean,5.1,ACTIVE PLACEBO) (%stax(age,STD,5.2,ACTIVE PLACEBO))
Range	%sysevalf(%stax(age,Max,3.,ACTIVE)- %stax(age,Min,3.,ACTIVE))	%sysevalf(%stax(age,Max,3.,PLACEBO)- %stax(age,Min,3.,PLACEBO))	%sysevalf(%stax(age,Max,3.,ACTIVE PLACEBO)- %stax(age,Min,3.,ACTIVE PLACEBO))
Median	%stax(age,Median,4.,ACTIVE)	%stax(age,Median,4.,PLACEBO)	%stax(age,Median,4.,PLACEBO)
Min - Max	%stax(age,MIN,3.,ACTIVE) - %stax(age,Max,3.,ACTIVE)	%stax(age,MIN,3.,PLACEBO) - %stax(age,Max,3.,PLACEBO)	%stax(age,MIN,3.,ACTIVE PLACEBO) - %stax(age,Max,3.,ACTIVE PLACEBO)
Gender			
Female	%counx(Gender,Female, ACTIVE)	%counx(Gender,Female, PLACEBO)	%counx(Gender,Female, ACTIVE PLACEBO)
Male	%counx(Gender,Male, ACTIVE)	%counx(Gender,Male, PLACEBO)	%counx(Gender,Male, ACTIVE PLACEBO)
Race			
White	%counx(Race,White, ACTIVE)	%counx(Race,White,PLACEBO)	%counx(Race,White,ACTIVE PLACEBO)
Black	%counx(Race,Black, ACTIVE)	%counx(Race,Black,PLACEBO)	%counx(Race,Black,ACTIVE PLACEBO)
Asian	%counx(Race,Asian, ACTIVE)	%counx(Race,Asian,PLACEBO)	%counx(Race,Asian,ACTIVE PLACEBO)
Other	%counx(Race,Other, ACTIVE)	%counx(Race,Other,PLACEBO)	%counx(Race,Other,ACTIVE PLACEBO)

3. WRITE PROC STREAM CODE WITH %INCLUDE SMARTMOCK.RTF AND OUTPUT FILE AS DELIVERABLE.RTF

```
%let path = %nrstr(C:\Users\admin\Desktop\StreamIN); /* update with your own
*/

filename shell "&path.\SmartMock.rtf" lrecl = 32755;
filename out "C:\Users\admin\Desktop\StreamOUT\Deliverable.rtf" lrecl =
32755;

proc stream outfile = out quoting = single  resetdelim="goto";
BEGIN goto; %include shell;
       ;;;;
```

4. RUN PROC STREAM
5. OUTPUT (Open Deliverable.rtf with Office):

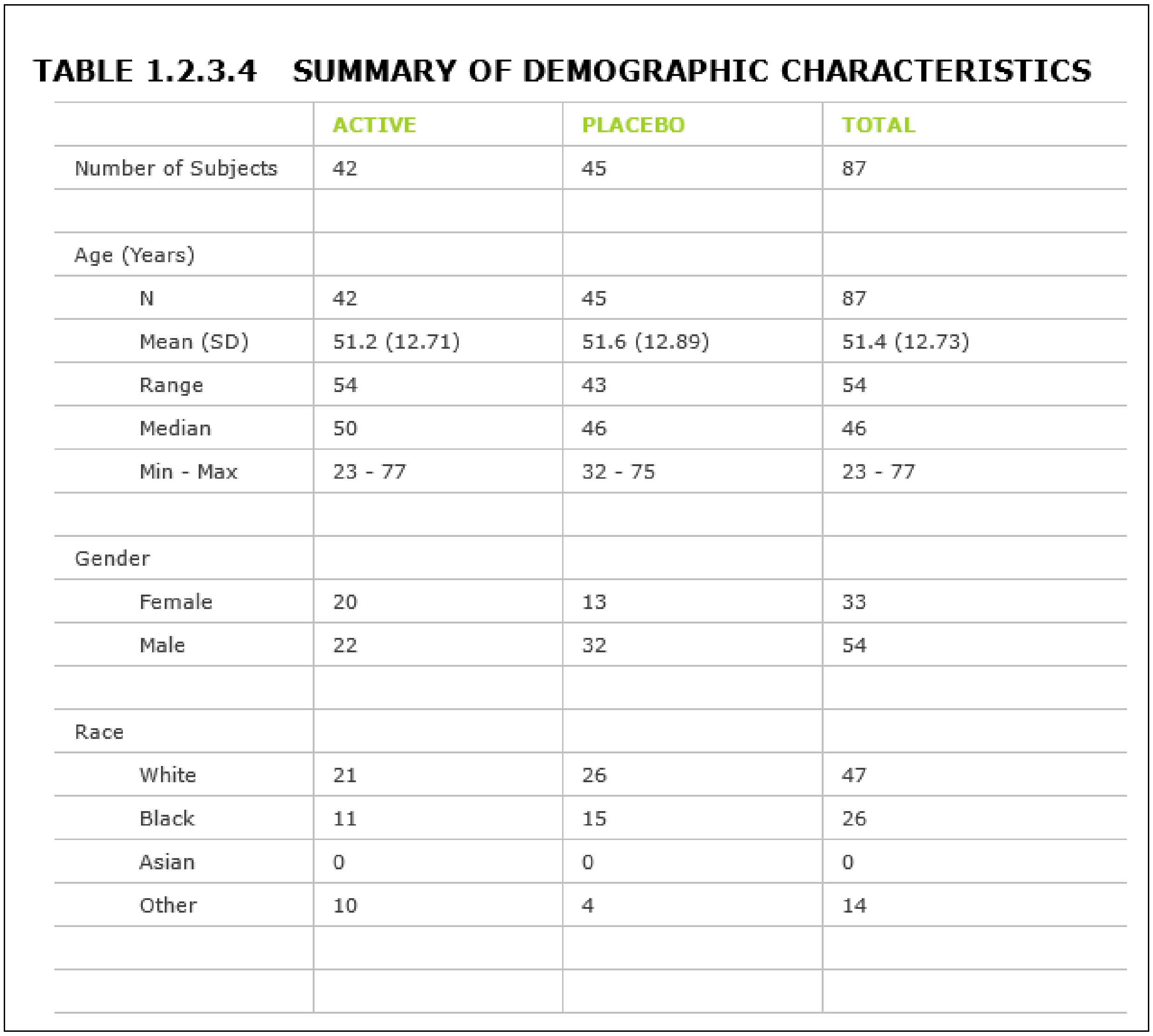


TABLE 1.2.3.4 SUMMARY OF DEMOGRAPHIC CHARACTERISTICS

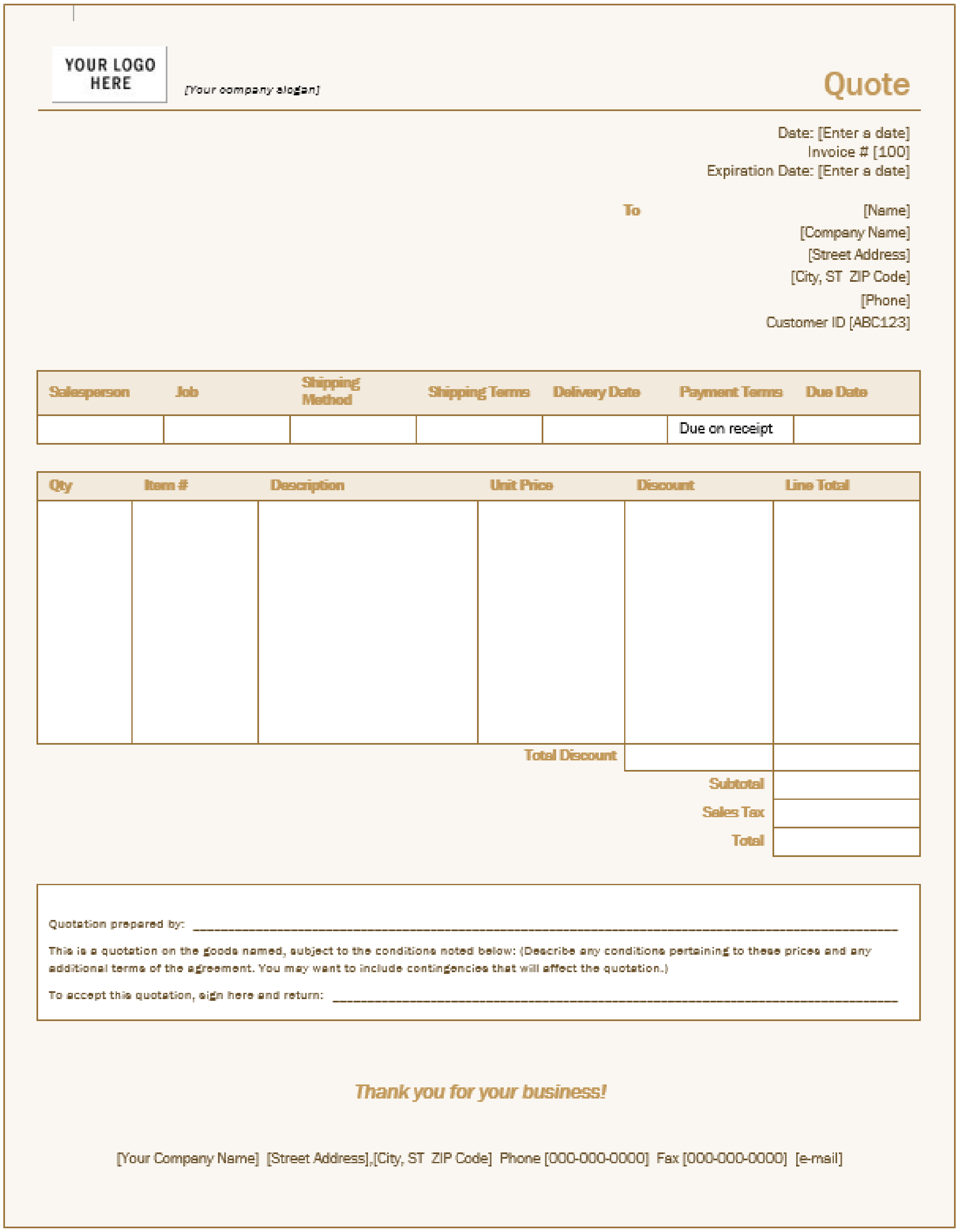
	ACTIVE	PLACEBO	TOTAL
Number of Subjects	42	45	87
Age (Years)			
N	42	45	87
Mean (SD)	51.2 (12.71)	51.6 (12.89)	51.4 (12.73)
Range	54	43	54
Median	50	46	46
Min - Max	23 - 77	32 - 75	23 - 77
Gender			
Female	20	13	33
Male	22	32	54
Race			
White	21	26	47
Black	11	15	26
Asian	0	0	0
Other	10	4	14

Output 1. Output from Proc STREAM

A SECOND EXAMPLE USING A MORE ELABORATE TABLE LAYOUT

This example will show how it matters little if the mock has a complex layout. Needless to say, clinical reports do not require fancy tables, and the typical bare-bones layout is the norm. The example below is just to illustrate how versatile the technique is. The template used is more suited for financial reports, where "looks" might matter. In the fancy template below, the same macro calls will be utilized and no reprogramming would be needed.

Again, a free OFFICE 2013 WORD template is being used: a commercial invoice table ("Sales Quote – Sienna Design"), as shown below:



YOUR LOGO HERE

[Your company slogan]

Quote

Date: [Enter a date]
Invoice # [100]
Expiration Date: [Enter a date]

To

[Name]
[Company Name]
[Street Address]
[City, ST ZIP Code]
[Phone]
Customer ID [ABC123]

Salesperson	Job	Shipping Method	Shipping Terms	Delivery Date	Payment Terms	Due Date
					Due on receipt	

Qty	Item #	Description	Unit Price	Discount	Line Total
			Total Discount		
				Subtotal	
				Sales Tax	
				Total	

Quotation prepared by: ____

This is a quotation on the goods named, subject to the conditions noted below: (Describe any conditions pertaining to these prices and any additional terms of the agreement. You may want to include contingencies that will affect the quotation.)

To accept this quotation, sign here and return: ____

Thank you for your business!

[Your Company Name] [Street Address],[City, ST ZIP Code] Phone [000-000-0000] Fax [000-000-0000] [e-mail]

Display 2. A Free OFFICE 2013 WORD Template ("Sales Quote – Sienna design")

The template is edited to change labels and converted to an RTF document then transformed into a Clinical Smart Mock Table (SmartMock.rtf) as shown below:

YOUR LOGO HERE *CLINICAL SUMMARY TABLE*

14.1.2.2 Subject Demographics

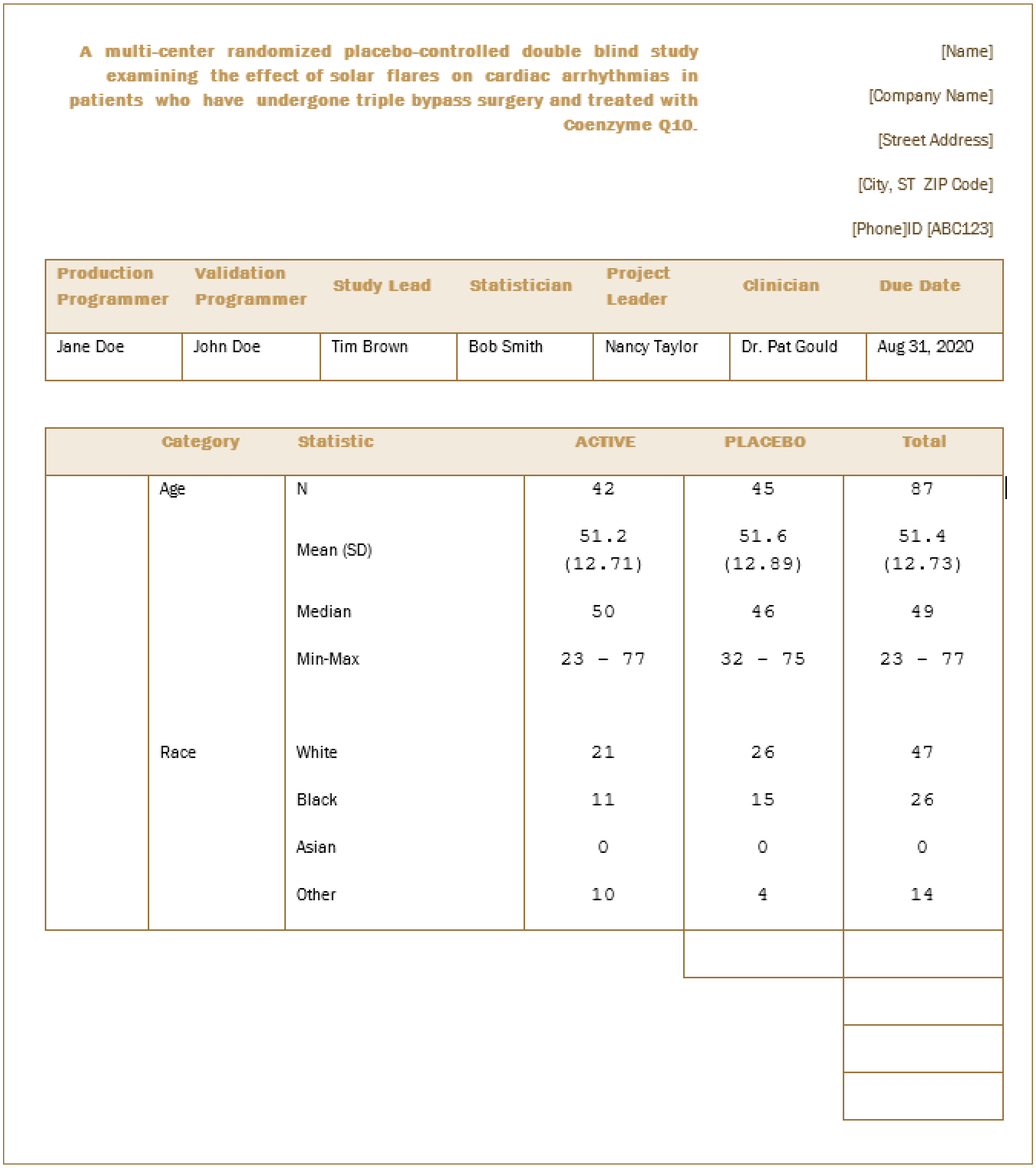
A multi-center randomized placebo-controlled double blind study examining the effect of solar flares on cardiac arrhythmias in patients who have undergone triple bypass surgery and treated with Coenzyme Q10.

[Name]
[Company Name]
[Street Address]
[City, ST ZIP Code]
[Phone]
Customer ID [ABC123]

Production Programmer	Validation Programmer	Study Lead	Statistician	Project Leader	Clinician	Due Date
Jane Doe	John Doe	Tim Brown	Bob Smith	Nancy Taylor	Dr. Pat Gould	Aug 31, 2020

	Category	Statistic	ACTIVE	PLACEBO	Total
	Age	N	%stax(age,N,3.,ACTIVE)	%stax(age,N,3.,PLACEBO)	%stax(age,N,3.,ACTIVE PLACEBO)
		Mean (SD)	%stax(age,mean,5.1,ACTIVE) (%stax(age,STD,5.2,ACTIVE))	%stax(age,mean,5.1,PLACEBO) (%stax(age,STD,5.2,PLACEBO))	%stax(age,mean,5.1,ACTIVE PLACEBO) (%stax(age,STD,5.2,ACTIVE PLACEBO))
		Median	%stax(age,median,4.,ACTIVE)	%stax(age,median,4.,PLACEBO)	%stax(age,median,4.,ACTIVE PLACEBO)
		Min-Max	%stax(age,min,3.,ACTIVE) - %stax(age,max,3.,ACTIVE)	%stax(age,min,3.,PLACEBO) - %stax(age,max,3.,PLACEBO)	%stax(age,min,3.,ACTIVE PLACEBO) - %stax(age,max,3.,ACTIVE PLACEBO)
	Race	White	%counx(Race,White, ACTIVE)	%counx(Race,White, PLACEBO)	%counx(Race,White, ACTIVE PLACEBO)
		Black	%counx(Race,Black, ACTIVE)	%counx(Race,Black, PLACEBO)	%counx(Race,Black, ACTIVE PLACEBO)
		Asian	%counx(Race,Asian, ACTIVE)	%counx(Race,Asian, PLACEBO)	%counx(Race,Asian, ACTIVE PLACEBO)
		Other	%counx(Race,Other, ACTIVE)	%counx(Race,Other, PLACEBO)	%counx(Race,Other, ACTIVE PLACEBO)

When processed with Proc Stream using the same code as in Example 1, the output below is obtained:



A multi-center randomized placebo-controlled double blind study examining the effect of solar flares on cardiac arrhythmias in patients who have undergone triple bypass surgery and treated with Coenzyme Q10.

[Name]

[Company Name]

[Street Address]

[City, ST ZIP Code]

[Phone]ID [ABC123]

Production Programmer	Validation Programmer	Study Lead	Statistician	Project Leader	Clinician	Due Date
Jane Doe	John Doe	Tim Brown	Bob Smith	Nancy Taylor	Dr. Pat Gould	Aug 31, 2020

	Category	Statistic	ACTIVE	PLACEBO	Total
	Age	N	42	45	87
		Mean (SD)	51.2 (12.71)	51.6 (12.89)	51.4 (12.73)
		Median	50	46	49
		Min-Max	23 - 77	32 - 75	23 - 77
	Race	White	21	26	47
		Black	11	15	26
		Asian	0	0	0
		Other	10	4	14

CONCLUSION

The use of Proc STREAM can enable the placement of an entire RTF mock table onto the SAS Input Stack. If such a table contains macro elements, the Macro Facility can resolve them, thereby allowing the table to update itself with new data. The mock table can even contain macro calls with DATA steps and procedures, so long as those steps are wrapped inside a DOSUBL and invoked with a %SYSFUNC. The ability to contain macro calls is what would make an RTF mock table “smart”. A smart mock table for a clinical summary can therefore generate a deliverable, after handling by Proc STREAM, without any table programming. Such an approach could potentially free the clinical programmer to focus entirely on statistical analyses of data. This technique is applicable to any RTF document that needs to contain computed data. The author has provided another example, the Consort Flow Diagram, in a second paper.

SOME CAVEATS

(1) Macro Calls must be put into the RTF version of the original mock and not on the WORD (.doc) version.
(2) The mock should not be the annotated version.
(3) Macro Calls MUST be typed, and not copied from an external source, to ensure the macro syntax is not contaminated with extraneous RTF codes.
(4) It’s okay to copy Macro Calls from one part of the RTF mock to another part of the same RTF document.
(5) The Macro Calls MUST be plain font (no color or bold, or italic), to avoid syntax contamination with extraneous RTF codes.
(6) The outermost quotes of the DOSUBL argument MUST be single.
(7) The macro variable statement inside the macros should NOT end with a semicolon.

Example:
```
%macro bn(gp);
%let t=%sysfunc(dosubl('proc sql noprint;select
count(distinct subjid) into :bgn TRIMMED from dmdata where
indexw("&gp.",trt) gt 0;quit;'));
&bgn.
%mend bn;
```

POTENTIAL ADVANTAGES

- Table cosmetic changes can quickly be done on the smart mock without any SAS reprogramming.
- Rearrangement of data can be achieved by simply relocating the macro calls.
- Similarly, analyses can be updated only at the macro level without touching the table layout.
- It matters not how complex or elaborate the mock table – only the macro calls need to be inserted in a document. This is of tremendous benefit to financial documents in particular, as they tend to have complex layouts (for example invoices).
- Outputs styles can quickly be changed since no table programming is involved.
- Titles and footnotes would be as provided by the statistician on the mock, with no programmer involvement, minimizing display errors and the need to update.
- Any table restructuring would be effected at the mock table level with absolutely no reprogramming.
- Table validation by double-programming would only occur at the analysis macro level, with only visual inspection utilized for the smart mocks.

- By directly using the mock table to create a deliverable, all outputs using the same smart mock would appear identical, even if analyses done by different programmers with different algorithms.

CONTACT INFORMATION

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

Joseph W. Hinson, PhD
inVentiv Health
202 Carnegie Center, Suite 200
Princeton, NJ, 08540
1-609-282-1615
joehinson@outlook.com

sas® Certified Advanced Programmer *for* SAS®9

sas® Certified Clinical Trials Programmer Using SAS®9