PharmaSUG 2013 - Paper CC13

Creating a Clinical Summary Table within a Single DATA Step with the Dynamic Trio: DOSUBL(), Hash Objects, and ODS Objects

Joseph Hinson, Princeton, NJ

ABSTRACT

SAS® 9.3 provides new features like the DOSUBL function, which allows PROCs to be called within a DATA step, and makes the procedure output immediately available to the same DATA step upon procedure exit. This property of DOSUBL makes it somewhat superior to Call Execute. SAS 9.3 also features an updated ODS Report Writing Interface (RWI), an object-oriented pre-production tool that provides great flexibility in controlling how data appear in tables and figures. The Hash object interface, first introduced with SAS 9, is offered again with a few new methods. These three tools can interact within the DATA step: clinical data can be processed by calling statistical PROCs with DOSUBL, the results organized with hash objects, and the hash data transformed into a final report with the ODS RWI object. This novel approach very much streamlines the intricate task of managing different procedure outputs to generate a clinical summary report. The present paper is about the production of a demographic summary table, as an example.

INTRODUCTION

Clinical summary tables are often at the heart of clinical trial reporting[1]. Prominent in the generation of summary tables are SAS statistical procedures such as UNIVARIATE, FREQ, MEANS, TTEST, NPAR1WAY, GLM, LOGISTIC, and LIFETEST, to mention a few. Usually the outputs of these procedures need to be easily incorporated into the summary table. Since the DATA step cannot include PROCs, the only way to get data from the various PROCs has been making them generate output data sets and later combining them into a single entity compatible with the summary table layout. This becomes quite a challenge when different procedures yield different tabular structures, requiring transpositions and complex restructuring in many cases.

If a way could be found to run PROCs in the DATA step and dynamically collect their output data directly, manipulating such outputs could be simpler and creating summary tables would be much straightforward.

In this paper, we demonstrate that the use of the new SAS function, DOSUBL, is one such way. Using DOSUBL allows PROCs to be run and the results made available within the same DATA step. Two object systems are then employed for consolidating data: the hash object for collecting and organizing PROC output data, and the ODS object for creating a tabular display. Hash objects, which are just DATA step memory tables, allow flexible manipulation of data structures, by their inherent matrix-like data access. The ODS Report Writing Interface (RWI) first introduced with SAS 9.2, and much enhanced in SAS 9.3 is still pre-production, but provides incredible ease and flexibility in graphical data output. Until the RWI, the use of ODS was largely restricted to PROCs. The DATA step can now attain new prowess with the hash and ODS objects.

In the present paper, a clinical demographics summary table is generated with a single DATA step, by using the DOSUBL function to run statistical PROCs, organizing the data from PROC outputs with hash objects, then using the ODS RWI object to read the hash rows to create a summary table.

THE DOSUBL FUNCTION

DOSUBL is an experimental function introduced in SAS version 9.3. This novel function is unique in that it can be called directly within a DATA step and executes SAS code directly[2].

The function is called with a return code variable "rc" provided:

```
rc=dosubl (" the text of the entire SAS code to execute ");
```

Using the function, one can execute a PROC inside a DATA step and immediately use the PROC output, unlike CALL EXECUTE, which just queues SAS code to be executed only after the DATA step completes. When DOSUBL is called, the function does not return until the particular SAS code completes. Data access functions, like OPEN, FETCH, and GETVARN can then be used to access the results made available by DOSUBL. In this paper, DOSUBL was used to run statistical PROCs.

HASH OBJECTS

Hash objects made their debut with SAS 9 and has since gained some popularity, mainly for fast look-up tables and sortless merges. They are memory-resident tables that operate through the DATA step component object interface. Many introductory papers have been written about hash objects and a great tutorial can be found in the reference[3].

In this paper, hash objects were used in four ways:

1. to create a shell ("mock") table
2. to enable parameter-by-parameter processing of clinical demographic data. Using the hash iterator object *hiter*, names of demographic parameters stored in the hash table were recalled one at a time for processing of associated data
3. to collect PROC output data into the same table as different PROCs were executed.
4. as a hash-of-hashes, by splitting the summary table into separate hash tables, one for each demographic parameter, within the main hash object. Then a read-out of the hash-of-hashes allowed parameter-by-parameter creation of the summary table, with easy captioning and indentation.

ODS REPORT WRITING INTERFACE

This is an object-oriented language for SAS version 9, that has been updated for SAS version 9.3, but is still pre-production[4]. It is designed to provide ODS technology to the DATA step. Until now, ODS-generated outputs have been the mainstay of PROCs. With RWI, the DATA step can directly communicate with the ODS system and features, including proportional fonts, traffic, and Unicode characters. Likewise, the ODS system, through RWI, can take advantage of the lighting, colors, images powerful programming features of the DATA step, including conditional logic, by-group processing, arrays, and hash object programming. Here are key features of the ODS RWI object system:

Like hash objects, the dot syntax is used to write ODS object methods, following the object declaration and instantiation.

1. The ODS object class is called "***odsout***".
2. The "declare" statement creates an instance of the ***odsout*** class, and calls it "***obj***", as shown below:

   ```
   declare odsout obj();
   ```

   Like hash objects, the declaration must be issued just once (when _N_=1).
3. Various object methods can then be invoked using the created object "***obj***", as shown below (full method details are available in the Dan O'Connor paper) [4]:

   ```
   obj.title(), obj.region(), obj.table_start(), obj.row_start(),
   ```

   ```
   obj.format_cell(),
   ```

   etc.
4. To create a table, the ODS object methods are invoked in a hierarchical manner:

   title---> layout--->region--->table--->row--->cell
5. A "start" method always needs to terminate with a corresponding "end" method, and in a nested manner, as shown below:

   ```
   obj.table_start()
   obj.row_start()
   obj.row_end()
   obj.table_end()
   ```
6. An ODS object method for populating a table cell with data is:

   ```
   obj.format_cell()
   ```
   which carries a variety of arguments: *inhibit, overrrides, text, span,* etc.

Using DATA step iteration, any ODS object method or group of methods can then be repeated any number of times to create the desired report layout. In this paper, hash iterator objects were used to control ODS method iteration as well as to provide data.

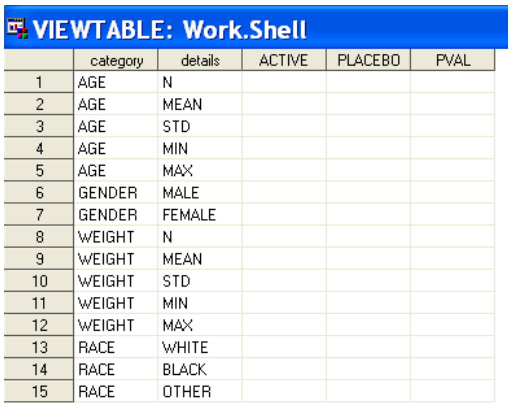
PROGRAMMING STRATEGY

Overall, this is a three-part approach (after first creating the input data set):

1. analyze data by using DOSUBL() to run a statistical PROC, then
2. use hash objects to capture and restructure the PROC output data, and finally
3. transform the hash data into a summary table with ODS RWI.

DATA STEP-I: Input Data Preparation:

Clinical demographic data can be organized into *category* and *details*. In this paper, the *category* items are AGE, GENDER, RACE, or WEIGHT. For the continuous *category* items like Age and Weight, the *details* are simply the descriptive statistics: N, MEAN, MIN, MAX, and STD. For the discrete *category* items, Gender has "MALE" and "FEMALE" as *details*, whereas Race has "WHITE", "BLACK", and "OTHER". So, a shell table (Figure 1) is first created containing the desired list of *category* items and their corresponding *details* items. This table would be loaded into hash objects and used to perform analysis one *category* item at a time, as well as for proper layout of the final summary table:


VIEWTABLE: Work.Shell

	category	details	ACTIVE	PLACEBO	PVAL
1	AGE	N			
2	AGE	MEAN			
3	AGE	STD			
4	AGE	MIN			
5	AGE	MAX			
6	GENDER	MALE			
7	GENDER	FEMALE			
8	WEIGHT	N			
9	WEIGHT	MEAN			
10	WEIGHT	STD			
11	WEIGHT	MIN			
12	WEIGHT	MAX			
13	RACE	WHITE			
14	RACE	BLACK			
15	RACE	OTHER			

Figure 1. A shell table loaded into hash to control analysis

An input data set is then created from raw demographics data while decoding the numeric values for Treatment, Gender, and Race into their text names using the CHOOSEC function:

```
GENDER=choosec(GENDcode,"MALE","FEMALE");
```

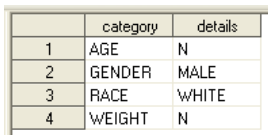
DATA STEP-II: Data Analyses and Reporting:

Section-A: CREATION OF HASH TABLES

Two hash tables are created from the shell table:

1. The *category* hash table "***ct***"

   This object ***ct*** contains unique values of *category* and is used to iterate the analysis category-by-category through its iterator object, ***hict***:



	category	details
1	AGE	N
2	GENDER	MALE
3	RACE	WHITE
4	WEIGHT	N

Figure 2. A category hash table

All continuous *category* items would have "N" as *details*, and anything else would be considered discrete. This is used to select the appropriate statistical PROC for DOSUBL function. For example:

```
do while (hict.next() eq 0);
if details ne "N" then
do;
*------CATEGORICAL/FREQUENCIES-------------------;
rc=dosubl('proc freq data=dmdata noprint;tables
  trt*'||category||'/list out=freqset;run;');
```

2. Hash table "***tab***"

   This object is also derived from the shell table and is used to collect summary data from PROC output data sets as analyses take place. Results are picked up by hash object ***tab*** as analyses are conducted, using the hash object methods as shown below:

```
rc=tab.find();
  -
  -
rc=tab.replace();
```

Section-B: STATISTICAL ANALYSES

Step-B1/B2:
The hash iterator object ***hict*** for hash object ***ct***, is stepped through one row at a time, to pick up *category/details* pairs, deciding whether to select a PROC for a continuous or discrete variable.

Step-B3/B4:
1. For discrete types of *category*, the DOSUBL function is invoked to execute PROC Freq.
2. Data access functions: open(), attrn(), fetch(), varnum(), getvarc(), and getvarn(), are then employed. to retrieve data from the PROC Freq output data set ("freqset").
3. The hash object ***tab*** is then used to store the retrieved values.

To preserve existing data in the hash object, the hash method ***tab.find()*** is first used to retrieve existing data (if any) from the hash table. New data becomes available from (b) and the method ***tab.replace()*** is then invoked to update the hash table with both old and new data.

Because all processing stay in the same DATA step, the hash object persists, and get updated with new data whenever a PROC is executed by DOSUBL.

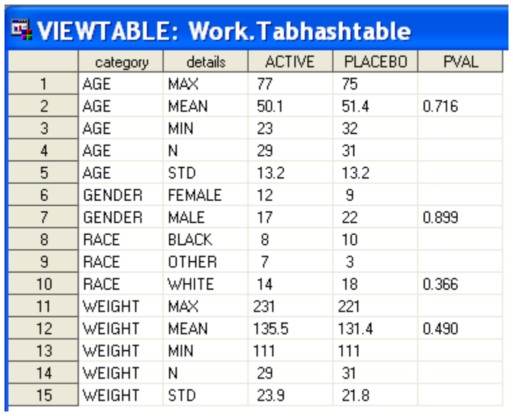
Step-B5:
1. DOSUBL next computes the P-Value for discrete variables with PROC Freq/FISHER EXACT.
2. PROC output data are accessed and values sent to hash object via ***tab.find()*** and ***tab.replace().***

Step-B6/B7/B8:

For continuous *category* items, DOSUBL runs,

1. PROC Summary for descriptive statistics, and
2. PROC GLM/Anova for P-Value.

When all category items have been analyzed, object ***tab*** looks like the table below:



VIEWTABLE: Work.Tabhashtable

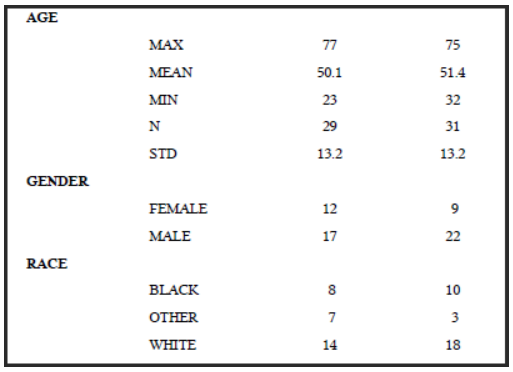
	category	details	ACTIVE	PLACEBO	PVAL
1	AGE	MAX	77	75	
2	AGE	MEAN	50.1	51.4	0.716
3	AGE	MIN	23	32	
4	AGE	N	29	31	
5	AGE	STD	13.2	13.2	
6	GENDER	FEMALE	12	9	
7	GENDER	MALE	17	22	0.899
8	RACE	BLACK	8	10	
9	RACE	OTHER	7	3	
10	RACE	WHITE	14	18	0.366
11	WEIGHT	MAX	231	221	
12	WEIGHT	MEAN	135.5	131.4	0.490
13	WEIGHT	MIN	111	111	
14	WEIGHT	N	29	31	
15	WEIGHT	STD	23.9	21.8	

Figure 3. Contents of hash object "tab" after analysis

This is the data eventually used by the ODS Report Writing Interface to create the clinical demographics summary report.

Section-C: TABLE RESTRUCTURING WITH HASH-OF-HASHES

Before the hash object ***tab*** is presented to the ODS object, it is first transformed into a hash-of-hashes. In other words, rather than having a flat hash table with all the data presented as rows and columns, the table is transformed into one main hash table containing "sub-hash" tables, one for each *category*. Thus, when a *category* is read out, it presents its sub-hash table of data as an entity for the ODS object. This facilitates the generation of the summary table layout, which usually has side indented labels and laid out as isolated sections, as shown below:



AGE			
	MAX	77	75
	MEAN	50.1	51.4
	MIN	23	32
	N	29	31
	STD	13.2	13.2
GENDER			
	FEMALE	12	9
	MALE	17	22
RACE			
	BLACK	8	10
	OTHER	7	3
	WHITE	14	18

Figure 4. Summary table layout with side indented labels

Creating the Hash-of-Hashes:
Hash object ***tab*** is transformed into a main hash object "***c***" containing hash objects "***d***" and its iterator object "***hid***".

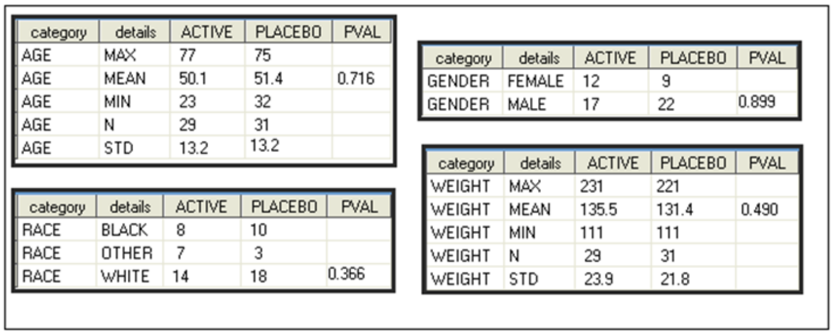
Step-C1:
Main object ***c*** and iterator ***hic*** are created and sub-hash ***d***, and its iterator ***hid***, are declared as shown below:

```
declare hash c (ordered:"a");
c.defineKey("category");
c.defineData("category","d","hid");
c.defineDone();
declare hiter hic("c");
declare hash d;
declare hiter hid;
```

Step-C2:
The objects ***d*** and ***hid*** are then instantiated and ***d*** filled with data using object ***tab***'s iterator object, ***hitab***, to step through its rows. The whole idea is to partition data into small tables ("sub-hashes") based on *category*. For a particular *category*, the code would test to see if it already has a sub-hash object created (rc1=0). If not, it creates it, using the _**new**_ operator, and populating the sub-hash with data related only to that particular *category*:

```
do while (hitab.next() eq 0);
rc1=c.find(key:category); *<--[rc1=0 means the search was successful];
if rc1 ne 0 then
do;
d = _new_ hash(ordered:"a");
d.defineKey("details");
d.defineData("category","details","active","placebo","pval");
hid = _new_ hiter("d");
rcc=c.replace();
end;
rcd=d.replace();
end;
```

Hash table c now containing four sub-hash tables:



category	details	ACTIVE	PLACEBO	PVAL
AGE	MAX	77	75	
AGE	MEAN	50.1	51.4	0.716
AGE	MIN	23	32	
AGE	N	29	31	
AGE	STD	13.2	13.2	

category	details	ACTIVE	PLACEBO	PVAL
GENDER	FEMALE	12	9	
GENDER	MALE	17	22	0.899

category	details	ACTIVE	PLACEBO	PVAL
RACE	BLACK	8	10	
RACE	OTHER	7	3	
RACE	WHITE	14	18	0.366

category	details	ACTIVE	PLACEBO	PVAL
WEIGHT	MAX	231	221	
WEIGHT	MEAN	135.5	131.4	0.490
WEIGHT	MIN	111	111	
WEIGHT	N	29	31	
WEIGHT	STD	23.9	21.8	

Figure 5. Four separate hash tables within main hash table "c"

Section-D: ASSEMBLE SUMMARY REPORT WITH ODS RWI

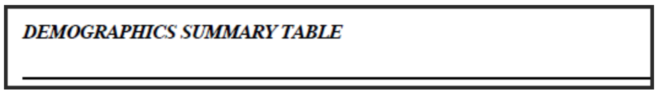
Step-D1/D2**:**
Declare and instantiate ODS RWI object odsout and call the new object summ:

```
  declare odsout summ();
```

Step-D3**:**
1. Begin creating title and separator line, using object methods:

```
     summ.title(text:"DEMOGRAPHICS SUMMARY TABLE");
     summ.layout_gridded(columns:1);
     summ.region();
     summ.line();
```

which produces:



DEMOGRAPHICS SUMMARY TABLE

Figure 6. Title and separator line generated by ODS Report Writing Interface

2. Generate headings including population totals for treatment groups (contained in the variables active and placebo). The "split:" method argument allows splitting of headings into two rows. The "inhibit: TLR" method argument suppresses the Top, Left, and Right border lines for a cell in the row. The "overrides:" argument imposes custom formatting over default settings.
Object methods are invoked to initiate table formation:

```
  summ.table_start();
  summ.row_start();
```

3. The following hash method is invoked to retrieve the values for treatment population totals,

using the AGE *category*. Strings are then created with the information:

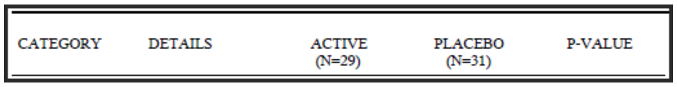
```
    rc=tab.find(key:"AGE",key:"N");
     nact="(N="||compress(active)||")";
     nplac="(N="||compress(PLACEBO)||")";
```

4. Object methods are now invoked to initiate table formation:

```
    summ.format_cell(inhibit:"TLR", text: "CATEGORY", overrides:"cellwidth=3cm");
     summ.format_cell (inhibit:"TLR", text: "DETAILS", overrides:"cellwidth=3cm");
     summ.format_cell (inhibit:"TLR", text: cat("ACTIVE#",nact),
     overrides:"cellwidth=3cm just=center" , split:"#");
     summ.format_cell (inhibit:"TLR", text:
```

```
    cat("PLACEBO#",nplac),overrides:"cellwidth=3cm
    just=center" , split:"#");
    summ.format_cell (inhibit:"TLR", text: "P-VALUE", overrides:"cellwidth=3cm
    just=center ", split:"#");
    summ.row_end();
```

which produces:



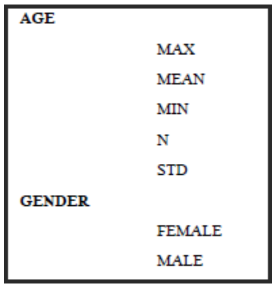
CATEGORY	DETAILS	ACTIVE (N=29)	PLACEBO (N=31)	P-VALUE

Figure 7. Table headers generated by ODS Report Writing Interface

Step-D4/D5:
The iterator object, ***hic***, is then used to step through the *category* sub-hash objects. The ***summ*** object method picks up *category* data exposed by the hash iterator and populates the row with it. The method argument, "column_span: 5", makes sure the five rows are merged to contain the *category* text:

```
 do while (hic.next() eq 0);
 summ.row_start();
 summ.format_cell(inhibit:"BLR", text: category, column_span:5,
 overrides:"font_weight=bold");
 summ.row_end();
```

This enables indentation of the side captions as shown below:



AGE	
	MAX
	MEAN
	MIN
	N
	STD
GENDER	
	FEMALE
	MALE

Figure 8. Side captions abd indented row labels created by ODS Report Writing Interface

Step-D6/D7:
Now the rest of the data for the table can be assigned. For each *category* , the iterator object, ***hid***, is used to pick a row from the *details* data for that particular *category*:

```
 do while(hid.next() eq 0);
 summ.row_start();
 pvalue=pval;
 summ.format_cell(inhibit:"BTLR", text: blank2, overrides:"cellwidth=3cm");
 summ.format_cell(inhibit:"BTLR", text: details, overrides:"cellwidth=3cm");
 summ.format_cell(inhibit:"BTLR", text: active, overrides:"cellwidth=3cm
 just=center");
 summ.format_cell(inhibit:"BTLR", text: placebo, overrides:"cellwidth=3cm
 just=center");
 summ.format_cell(inhibit:"BTLR", text: pvalue, overrides:"cellwidth=3cm

 just=center");
```

The ODS RWI system requires that "end" methods are invoked in reverse order, to complement "start" methods, following execution of methods:

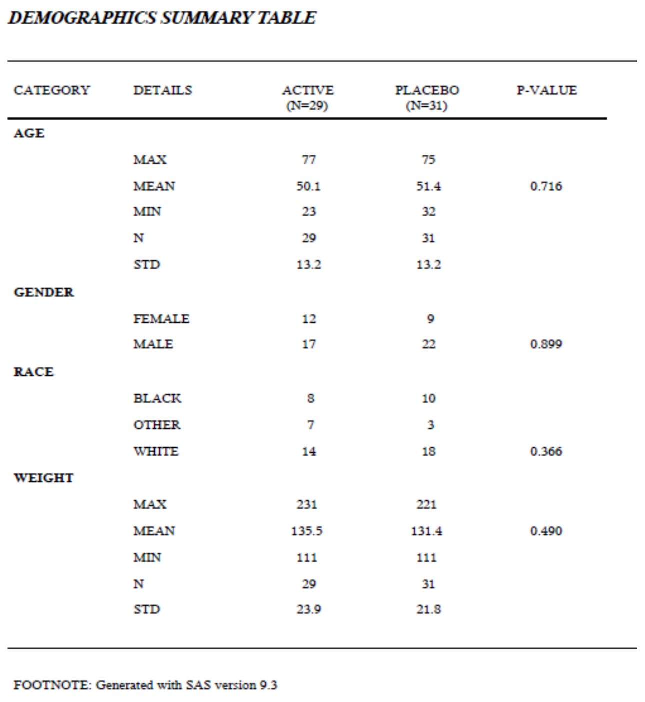
```
 summ.row_end();
 summ.table_end();
 summ.line();
 summ.region();
```

Step-D6/D7:
A footnote can be inserted at the bottom as follows:

```
 summ.table_start();
```

```
summ.row_start();
summ.format_cell(inhibit:"BTLR", text:"FOOTNOTE: Generated with SAS version
9.3",column_span:5,overrides:"cellwidth=15cm");
summ.row_end();
summ.table_end();
summ.layout_end();
```

THE COMPLETED DEMOGRAPHICS REPORT



DEMOGRAPHICS SUMMARY TABLE

CATEGORY	DETAILS	ACTIVE (N=29)	PLACEBO (N=31)	P-VALUE
AGE				
	MAX	77	75	
	MEAN	50.1	51.4	0.716
	MIN	23	32	
	N	29	31	
	STD	13.2	13.2	
GENDER				
	FEMALE	12	9	
	MALE	17	22	0.899
RACE				
	BLACK	8	10	
	OTHER	7	3	
	WHITE	14	18	0.366
WEIGHT				
	MAX	231	221	
	MEAN	135.5	131.4	0.490
	MIN	111	111	
	N	29	31	
	STD	23.9	21.8	

FOOTNOTE: Generated with SAS version 9.3

Table 2. The final demographics summary table generated by ODS Report Writing Interface

CONCLUSION

The present paper has demonstrated that using the newer SAS tools, DOSUBL, Hash Objects, and ODS Report Writing Interface, a clinical summary report can be generated in a much simplified and straightforward approach. Many papers have been presented about summary table creation with SAS, a task that is often complicated by the need to run separate statistical PROCs, with each procedure producing an entirely different table of results. Usually, the task involves a variety of intricate table transpositions, merges, concatenations, and restructuring strategies. As shown in this paper, by executing PROCs within the DATA step as afforded by DOSUBL, and by using object programming as provided by Hash and ODS RWI interfaces, the process of conducting statistical analyses and collecting results for table creation can seamlessly occur within a single DATA step with much greater simplicity.

REFERENCES

1. John Henry King, 2012**.** The Ubiquitous Clinical Trials Data Summary Table “Summary Statistics in Rows” PharmaSUG2010 - Paper TT05
http://www.lexjansen.com/pharmasug/2010/tt/tt05.pdf

2. Jason Sekosky, 2012. Executing a PROC from a DATA step.
Proceedings of the SAS Global Forum 2012 - Paper 2012-227.
http://support.sas.com/resources/papers/proceedings12/227-2012.pdf

3. Dorfman, Paul and Vyverman, Koen, 2005. Data Step Hash Objects as Programming Tools. Proceedings of the Thirtieth Annual SAS Users Group International Conference - Paper 236-30

http://www2.sas.com/proceedings/sugi30/236-30.pdf

4. Daniel O'Connor, 2009. The Power to Show: Ad Hoc Reporting, Custom Invoices, and Form Letters. Proceedings of the SAS Global Forum 2009 - Paper 313. http://support.sas.com/rnd/base/datastep/dsobject/Power_to_show_paper.pdf

ACKNOWLEDGMENTS

The author is very grateful to his manager and mentor Margaret Coughlin and colleagues while at Merck, especially Changhong Shi, for guidance and providing a very intellectually-enriching environment for creative work.

CONTACT INFORMATION

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

Name:	Joseph Hinson, PhD
City, State ZIP:	Princeton, NJ, 08542
Work Phone:	1-609-540-1309
E-mail:	jwhinson@gmail.com

APPENDIX

The Complete Code (Requires SAS version 9.3 to run:

```
*[REPLACE THIS WITH OWN LOCATION];
%let outputlocation= C:\Documents and Settings\hinsonj\Desktop\SASoutputs\newsummary.pdf;


 *        D A T A   S T E P - I  [PREPARATION OF INPUT DATA SETS]

=================================================================
               C R E A T E    S H E L L    T A B L E
=================================================================;
data shell;
       infile datalines missover;
       input category $ details $ ACTIVE $ PLACEBO $ PVAL $;
       datalines;
AGE N
AGE MEAN
AGE STD
AGE MIN
AGE MAX
GENDER MALE
GENDER FEMALE
WEIGHT N
WEIGHT MEAN
WEIGHT STD
WEIGHT MIN
WEIGHT MAX
RACE WHITE
RACE BLACK
RACE OTHER
;
run;

*================================================================
        C R E A T E   I N P U T   D A T A  S E T
=================================================================;
data dmdata (drop=TRTcode GENDcode RACEcode);
       infile datalines;
       input SUBJID TRTcode GENDcode RACEcode AGE WEIGHT @@;
       *----DECODE VALUES FOR TREATMENT, RACE, and GENDER-------;
       trt=choosec(((TRTcode)+1), "ACTIVE","PLACEBO");
       GENDER=choosec(GENDcode,"MALE","FEMALE");
       RACE=choosec(RACEcode, "WHITE", "BLACK", "OTHER");
datalines;
101 0 1 3 37 112      301 0 1 1 70 131      501 0 1 2 33 111
601 0 1 1 50 124      701 1 1 1 60 121      102 1 2 1 65 123
302 0 1 2 55 143      502 1 2 1 44 181      602 0 2 2 30 152
702 0 1 1 28 133      103 1 1 2 32 132      303 1 1 1 65 122
503 1 1 1 64 133      603 1 2 1 33 133      703 1 1 2 44 126
104 0 2 3 23 122      304 0 1 1 45 134      504 0 1 3 56 122
604 0 1 3 65 125      704 0 2 1 66 171      105 1 1 3 44 221
305 1 1 1 36 113      505 1 1 2 73 143      605 1 2 1 57 122
705 1 1 2 46 124      106 0 2 1 49 144      306 0 1 2 46 143
506 0 1 1 46 122      606 0 1 2 56 133      706 1 1 1 75 143
201 1 1 3 35 134      401 1 2 1 44 111      507 1 1 2 44 112
607 1 1 1 67 143      707 1 1 1 46 115      202 0 2 1 50 154
402 0 2 2 77 122      508 0 2 1 53 141      608 0 2 2 46 142
708 0 2 1 55 231      203 1 1 2 49 122      403 1 1 1 45 133
509 0 1 3 45 132      609 1 2 1 72 116      709 0 2 2 57 132
204 0 2 3 60 113      404 1 1 1 59 131      510 0 1 3 65 113
610 0 1 1 29 118      710 0 1 1 63 165      205 1 1 3 39 132
405 0 2 1 49 124      511 1 2 2 43 123      611 1 2 1 65 125
711 1 1 2 61 144      206 1 2 1 67 111      406 1 1 2 33 142
512 1 1 1 39 111      612 1 1 2 46 132      712 0 1 1 49 121
```

```
;
run;

*     D A T A   S T E P - II  [DATA ANALYSES AND SUMMARY TABLE CREATION];
data _null_;
       *==================================================================
              Section-A:   C R E A T E     H A S H     O B J E C T S
       ==================================================================;
       *-----STEP A1----a false execution just to put variables in PDV*;
       if (1=2) then
          do;
              set Shell;
              set dmdata;
          end;
       *-----STEP A2----*;
       declare hash ct(dataset:"shell", ordered:"a");
              rcx=ct.defineKey("category");
              rcx=ct.defineData("category","details");
              rcx=ct.defineDone();
       declare hiter hict("ct");
       ct.output(dataset:"cattable");

       trtnum=2;
       *-----STEP A3----*;
       declare hash tab (dataset:"shell",ordered:"a");
              tab.defineKey("category","details");
              tab.defineData( "category","details","ACTIVE","PLACEBO","PVAL");
              tab.defineDone();
       declare hiter hitab ("tab");
       call missing (ACTIVE, PLACEBO, PVAL);

       *===================================================================
       Section-B:     D O    S T A T I S T I C A L     A N A L Y S E S
       ===================================================================;
       *-----STEP B1----*;
       do while (hict.next() eq 0);
       *-----STEP B2----*;
          if details ne "N" then do;
       *-----STEP B3------------CATEGORICAL/FREQUENCIES-------------------;
rc2=dosubl('proc freq data=dmdata noprint;tables trt*'||category||'/list out=freqset;run;');

              array frq(3,3) $ _temporary_;
              f=0;
              *-----STEP B3----*;
              dsid=open("freqset");
              obsnum=attrn(dsid,"NOBS");
              nx=obsnum/trtnum;
              rc1=fetch(dsid);
              do while (rc1=0);
                 f=f+1;
                 xcat=varnum(dsid,"trt");
                 cat=getvarc(dsid,1);
                 det=getvarc(dsid,2);
                 fval=put(getvarn(dsid,3),3.);
                 *-----STEP B4----*;
                 xx=whichc(compress(cat),"ACTIVE","PLACEBO","PVAL");
                 ff=f-(nx*(f>nx));
                 rc3=tab.find(key:category,key:det);
                 frq(ff,xx)=fval;
                 rc=tab.replace(key:category,key:det,
                 data:category,data:det,data:frq(ff,1),data:frq(ff,2),data:frq(ff,3));
                 rc1=fetch(dsid);
              end;
              rcc=close(dsid);

       *-----STEP B5---------CATEGORICAL/PVALUE via FISHER EXACT -------;
rc4=dosubl('proc freq data=dmdata noprint;tables trt*'||category||'/FISHER;EXACT FISHER;output
out=chitable FISHER;run;');
              dsid=open("chitable");
              rc1=fetch(dsid);
              rc3=tab.find();
```

```
                PVAL=put(getvarn(dsid,1),5.3); *<---[OUTPUT HAS FISHER P-VALUE
                                                     IN THE 1st COLUMN];
                rc=tab.replace();
                rcc=close(dsid);
           end;*[if details ne "N"];

           else if details="N" then do;
        *-----STEP B6--------CONTINUOUS/DESCRIPTIVE STATISTICS-----------;
rc1=dosubl('proc summary data=dmdata noprint;var '||category||';class trt;output
out=summtable(where=(_TYPE_ ne 0));run;');

                array smm(5,3) $ _temporary_;
                f=0;
                *-----STEP B7----*;
                dsid=open("summtable");
                obsnum=attrn(dsid,"NOBS");
                nx=obsnum/trtnum;
                rc1=fetch(dsid);
                do while (rc1=0);
                        f=f+1;
                        xcat=varnum(dsid,"trt");
                        cat=getvarc(dsid,1);   *<-------[COLUMN HAS THE CATEGORY NAME NAME];
                        xdet=varnum(dsid,"_STAT_");
                        det=getvarc(dsid,4);   *<--------[COLUMN 4 HAS STATISTIC NAME];
                        cval=getvarn(dsid,5);  *<-------[COLUMN 5 HAS THE VALUE];
                        *-----STEP B8----*;
                        if det in ("MEAN", "STD") then dval=put(cval,5.1);
                        else dval=put(cval, 3.);
                        xx=whichc(compress(cat),"ACTIVE","PLACEBO","PVAL");
                        ff=f-(nx*(f>nx));
                        rc3=tab.find(key:"category",key:det);
                        smm(ff,xx)=dval;
                        rc=tab.replace(key:category,key:det,data:category,
                                       data:det,data:smm(ff,1),data:smm(ff,2),data:smm(ff,3));
                        rc1=fetch(dsid);
                end;
                rcc=close(dsid);

        *-----STEP B9---------CONTINUOUS/PVALUE via ANOVA--------------;

rc2=dosubl('proc glm data=dmdata noprint outstat=pvaltable;class trt;model
'||category||'=trt;run;');
                *STEP B10--------*;
                dsid=open("pvaltable");
                rc1=fetch(dsid);  *<--[DO FETCH TWICE TO SKIP THE FIRST
                                       ROW OF GLM OUTPUT WHICH HAS MISSING PROB VALUE];
                rc2=fetch(dsid);
                details="MEAN";   *<---[WE WANT PVALUE JUST FOR MEAN];
                rc3=tab.find();
                PVAL=put(getvarn(dsid,7),5.3); *<----[GLM OUTPUT HAS PROB IN THE 7th COLUMN];
                *STEP B11--------*;
                rc=tab.replace();
                end;*[if details="N"];
                rcc=close(dsid);

           end;*[do while hict];

        *=========================================================================
        Section-C:     O R G A N I Z E    I N T O    H A S H - O F - H A S H E S
        ========================================================================;
        *STEP C1--------INITIALIZE HASH-OF-HASHES--------------------;
        declare hash c (ordered:"a");
                        c.defineKey("category");
                        c.defineData("category","d","hid");
                        c.defineDone();
        declare hiter hic("c");
        declare hash d;
        declare hiter hid;

        *STEP C2--------POPULATE HASH-OF-HASHES--------------------;
        do while (hitab.next() eq 0);
```

```
rc1=c.find(key:category); *<--[rc1=0 means the search was successful];
if rc1 ne 0 then
        do;
                d = _new_ hash(ordered:"a");
                d.defineKey("details");
                d.defineData("category","details","active","placebo","pval");
                d.defineDone();
                hid = _new_ hiter("d");
                rcc=c.replace();
        end;
        rcd=d.replace();
end;*[do while hitab];

*======================================================================
Section-D:         A S S E M B L E    S U M M A R Y    T A B L E
======================================================================;
*STEP D1--INITIALIZE ODS REPORT WRITING INTERFACE------;
options nonumber nodate nocenter;
ods listing close;
ods pdf file="&outputlocation." notoc;

length blank1 $10 blank2 $30;
blank1=repeat(byte(32),10);
blank2=repeat(byte(32),30);

*STEP D2---CREATE ODS OBJECT------------------------------;
declare odsout summ();
*STEP D3---BEGIN ODS RWI/ CREATE TITLE AND HEADINGS-------;
summ.title(text:"DEMOGRAPHICS SUMMARY TABLE");
summ.layout_gridded(columns:1);
  summ.region();
  summ.line();
  summ.table_start();
        summ.row_start();
          rc=tab.find(key:"AGE",key:"N");
          nact="(N="||compress(active)||")";
          nplac="(N="||compress(PLACEBO)||")";
          summ.format_cell(inhibit:"TLR", text: "CATEGORY", overrides:"cellwidth=3cm");
          summ.format_cell(inhibit:"TLR", text: "DETAILS", overrides:"cellwidth=3cm");
          summ.format_cell(inhibit:"TLR", text:
          cat("ACTIVE#",nact),overrides:"cellwidth=3cm just=center" , split:"#");
          summ.format_cell(inhibit:"TLR", text:
        cat("PLACEBO#",nplac),overrides:"cellwidth=3cm just=center" , split:"#");
          summ.format_cell(inhibit:"TLR", text:
        "P-VALUE", overrides:"cellwidth=3cm just=center ", split:"#");
        summ.row_end();
*STEP D4-----START READING MAIN HASH DATA ROW BY ROW------;
        do while (hic.next() eq 0);
*STEP D5-----CREATE ROW SUB-HEADINGS FOR LEFT COLUMN------;
        summ.row_start();
          summ.format_cell(inhibit:"BLR", text:
                category, column_span:5, overrides:"font_weight=bold");
        summ.row_end();
        *STEP D6-----START READING SUB HASH DATA ROW BY ROW-------;
        do while(hid.next() eq 0);
        *STEP D7-----CREATE ROWS OF SUMMARY DATA------------------;
        summ.row_start();
          pvalue=pval;
          summ.format_cell(inhibit:"BTLR", text: blank2, overrides:"cellwidth=3cm");
          summ.format_cell(inhibit:"BTLR", text: details, overrides:"cellwidth=3cm");
          summ.format_cell(inhibit:"BTLR", text:
                                      active, overrides:"cellwidth=3cm just=center");
          summ.format_cell(inhibit:"BTLR", text:
                                      placebo, overrides:"cellwidth=3cm just=center");
          summ.format_cell(inhibit:"BTLR", text:
                                      pvalue, overrides:"cellwidth=3cm just=center");
        summ.row_end();
        end;*[do while hid];
        end;*[do while hic];
summ.table_end();
summ.line();
```

```
        summ.region();
        *STEP D8-----CREATE TABLE FOOTNOTE------------------------;

        summ.table_start();
                summ.row_start();
                  summ.format_cell(inhibit:"BTLR", text:"FOOTNOTE:
                        Generated with SAS version 9.3",column_span:5,overrides:"cellwidth=15cm");
                summ.row_end();
        summ.table_end();
        summ.layout_end();
        stop;
run;
ods pdf close;
********************************* END OF PROGRAM ********************************************;
```