PharmaSUG 2024 - Paper DV- 356

Standardizing Validation Data Sets (VALDS) as matrices indexed by Page, Section, Row, and Columns (PSRC) to improve Validation and output creation and revisions.

Kevin R. Viel, Ph.D., Navitas Data Sciences and Histonis, Inc.

ABSTRACT

Table and listing (TL) shells are blueprints specifying the dimensions of the output: PAGES, SECTIONS, ROWS, and COLUMNS (PSRC), their orders, and the precision of the values in the cells indexed by those dimensions. Values may be a composite such as Mean (SD). Viel introduced the Validation Data Set (VALDS), a standardization data set derived and used as input to create output (RTF/PDF). Importantly, the VALDS formalizes the structure and format of such data sets, standardizing 1) TL programs of various types and projects, 2) their Validation, and 3) the manual (spot) checking of the output against the VALDS. The programmer (independently) derives a value and inserts or "pigeon holes" it into its respective cell in the VALDS. For instance, a derived value may be assigned to the cell indexed as COLUMN 3 of PAGE 2, SECTION 3, and ROW 1 of the TL. The goal of this paper is to 1) introduce the creation of an analytic data set (ADS) so that at most only one SAS System® FREQ and one MEANS procedure is typically required for the entire program, to index the (composite) value using its PAGE, SECTION, ROW, and COLUMN labels to place it in its appropriate cell of the VALDS matrix, and to demonstrate an efficient way to populate default values, including for dimensions specified by the shell, but absent in the data. Lastly, this paper demonstrates the ease of making major revisions to the output with extremely minimal changes to the TL programs.

INTRODUCTION

When deriving the datum of composite of data to produce a table or listing, the programmer places the data based on variables, perhaps guided by a shell. Technically, the term "derivation" should apply only to tables since listing should be a presentation of a data set subset by observations and/or variables. For instance, a flag variable such as SAFFL might define a page. Another example of a page might be a PARAM (LBTEST) in ADLB. A section might be a characteristic such as AGE or RACE, but it may be a VISIT. A row might be a value in that sections such as "ASIAN" or "Mean (SD)", the latter being a composite of two derived values.

Whether or not the shell is used as blueprint, the table (listing) can be viewed as a matrix with numerous cells. The indices of the cells related to the dimensions: PAGE, SECTION, ROW, and COLUMN, which have associated numbers. Such a structure can be used to the advantage of a programmer to standardize programs. The burden of the programmer is usually to derive the datum or composite data that appears in the cell. Using the "labels" and data sets that assign the indices (numbers) to a cell, reduces the burden on the programmer to place or "pigeon hole" the derived values into their appropriate cells. For example, one might say PAGE = "AST", SECTION = "VISIT 24", ROW = "Mean (SD)" and COLUMN = "Group 2 (N = 47)". Using the shell, one might associate those labels with numbers such as PAGE_ORDER_1 = 3, SECTION_ORDER_1 = 12, ROW_ORDER_1 = 3, and COLUMN = 2. While the first might be intuitive to the anyone, the second allows the programmer to use tools such as Hashes (associative arrays) and Arrays, while focusing on other issues like deriving the data.

This paper demonstrates this approach, which dovetails well into the **VAL**idation **D**ata **S**et (VALDS) approach presented previously[1,2]. In the near future, aspects of the shell, like the number of columns, order of pages, sections, and rows, and groups into columns as well as font-family and size and precision should be abstracted directly from the shell. Further automation would include the dimensions of the cells (height and width), which should allow the shell author to visualize these aspects at creation, allowing adjustments, such as the reduction of the number of columns or rows per page before the final output arrives.

SIMULATED ADSL

The following code generates a simulated ADSL with a seed that creates a repeatable data set (**Table 1**):

```
%let macvar = usubjid = catx( "-" %str(,) studyid %str(,) siteid %str(,) put( subjidn %str(,) z4. )) %str(;)
              age     = rand( "INTEGER" %str(,) 18 %str(,) 85 ) %str(;)
              bmibl   = rand( "INTEGER" %str(,) 18 %str(,) 85 ) + round( rand( "UNIFORM" ) %str(,) 0.1 )%str(;)
              race    = __r( rand( "INTEGER" %str(,) 1 %str(,) 3 )) %str(;)
              saffl   = ifc( mod( subjidn %str(,) 10 ) = 0 %str(,) "N" %str(,) "Y" ) %str(;)
              ittfl   = ifc( mod( subjidn %str(,) 5  ) = 0 %str(,) "N" %str(,) "Y" ) %str(;)
              agegr1  = ifc( 18 <= age <= 59 %str(,) "18-59" %str(,) "60+" ) %str(;)
              output %str(;)
              ;

data adsl
       ( drop   = subjidn ) ;
  call streaminit( 2024 ) ;
  array __r ( 3 ) $ 20 _temporary_
        ( "Asian", "Black", "White" ) ;
  retain studyid "001"
         siteid  "01"
         ;
  length usubjid $ 11 ;
  arm = "Group 1" ;
  do subjidn = 1 to 10 ;
    &macvar.
  end ;
  arm = "Group 2" ;
  do subjidn = subjidn to subjidn + 7 ;
    &macvar.
  end ;
  arm = "Group 3" ;
  &macvar.
run ;
```

Table 1. The simulated data set (ADSL) used in the paper.

studyid	siteid	usubjid	arm	age	bmibl	race	saffl	ittfl	agegr1
001	01	001-01-0001	Group 1	42	67.2	Black	Y	Y	18-59
001	01	001-01-0002	Group 1	66	61.3	Black	Y	Y	60+
001	01	001-01-0003	Group 1	66	63.9	Black	Y	Y	60+
001	01	001-01-0004	Group 1	64	30.6	Asian	Y	Y	60+
001	01	001-01-0005	Group 1	65	38.1	White	Y	N	60+
001	01	001-01-0006	Group 1	41	47.9	White	Y	Y	18-59
001	01	001-01-0007	Group 1	47	64.2	Black	Y	Y	18-59
001	01	001-01-0008	Group 1	55	75.3	Black	Y	Y	18-59
001	01	001-01-0009	Group 1	58	40.4	White	Y	Y	18-59
001	01	001-01-0010	Group 1	59	34.8	Black	N	N	18-59
001	01	001-01-0011	Group 2	38	32.1	White	Y	Y	18-59
001	01	001-01-0012	Group 2	70	22.7	Asian	Y	Y	60+
001	01	001-01-0013	Group 2	30	52.7	Black	Y	Y	18-59
001	01	001-01-0014	Group 2	22	28.5	White	Y	Y	18-59
001	01	001-01-0015	Group 2	72	68.1	Black	Y	N	60+
001	01	001-01-0016	Group 2	20	73.7	Black	Y	Y	18-59
001	01	001-01-0017	Group 2	60	45.8	White	Y	Y	60+
001	01	001-01-0018	Group 2	71	38.9	Asian	Y	Y	60+
001	01	001-01-0019	Group 3	66	27.5	White	Y	Y	60+

The summary and frequency listing of these data are in **Table 2**, **Table 3** (Group 1 only), and **Table 4**, (Group 2 only), respectively. These tables can verify the results in **Table 9**.

Table 2. The summary of continuous variables in ADSL for SAFFL = "Y" produced manually by the MEANS procedure.

arm	N Obs	Variable	N	Mean	Std Dev	Median	Lower Quartile	Upper Quartile	Minimum	Maximum
Group 1	9	age	9	56.00	10.32	58.00	47.00	65.00	41.00	66.00
		bmibl	9	54.32	15.43	61.30	40.40	64.20	30.60	75.30
Group 2	8	age	8	47.88	22.73	49.00	26.00	70.50	20.00	72.00
		bmibl	8	45.31	18.48	42.35	30.30	60.40	22.70	73.70
Group 3	1	age	1	66.00	.	66.00	66.00	66.00	66.00	66.00
		bmibl	1	27.50	.	27.50	27.50	27.50	27.50	27.50

Table 3. The frequency of AGEGR1 (a categorical variable) in ADSL for SAFFL = "Y" produced manually by the FREQ procedure for GROUP 1.

agegr1	Frequency	Percent	Cumulative Frequency	Cumulative Percent
18-59	5	55.56	5	55.56
60+	4	44.44	9	100.00

Table 4. The frequency of RACE (a categorical variable) in ADSL for SAFFL = "Y" produced manually by the FREQ procedure for GROUP 2.

race	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Asian	2	25.00	2	25.00
Black	3	37.50	5	62.50
White	3	37.50	8	100.00

DIMENSION DATA SETS

The dimensions of the "matrix" are PAGE, SECTION, ROW, and COLUMN. The programmer does not design the shells but follows it as a blueprint. Even when the order of rows is dynamic and determined by the data, for instance, in a table that presents adverse events (AEs) by descending frequency, the programmer is still following the shell (blueprint). The code in **Appendix 1** provides a working example of the data sets that associate an index with a label for the respective dimension.

Nesting occurs in the VALDS: Columns are nested in Rows, Rows are nested in Sections, and Sections are nested in Pages. The following code creates the __LEVELS data set, the collection of all labels, indices, and nesting (partially displayed in **Table 5**) using the MAC_TFL_DIMENSIONS macro (**Appendix 2**):

```
%mac_tfl_dimensions
   ( pages_ds              = __pages
   , sections_ds           = __sections
   , rows_ds               = __rows
   , columns_ds            = __columns
   ) ;
```

Table 5. The __LEVEL data sets produced as a combination of the __PAGES, __SECTIONS, __ROWS, and __COLUMNS data sets.

page_1	page_order_1	section_1	section_order_1	row_1	row_order_1	default	column_label	column
Safety Population	1	Race, n(%)	1	Race, n(%)	0		Group 1	1
Safety Population	1	Race, n(%)	1	Race, n(%)	0		Group 2	2
Safety Population	1	Race, n(%)	1	Race, n(%)	0		Group 3	3
Safety Population	1	Race, n(%)	1	Race, n(%)	0		Total	4
Safety Population	1	Race, n(%)	1	Asian	1	0	Group 1	1
Safety Population	1	Race, n(%)	1	Asian	1	0	Group 2	2
Safety Population	1	Race, n(%)	1	Asian	1	0	Group 3	3
Safety Population	1	Race, n(%)	1	Asian	1	0	Total	4
Safety Population	1	Race, n(%)	1	Black	2	0	Group 1	1
Safety Population	1	Race, n(%)	1	Black	2	0	Group 2	2
Safety Population	1	Race, n(%)	1	Black	2	0	Group 3	3
Safety Population	1	Race, n(%)	1	Black	2	0	Total	4
SNIPPED								
Safety Population	1	Age (years)	2	Age (years)	0		Group 1	1
Safety Population	1	Age (years)	2	Age (years)	0		Group 2	2
Safety Population	1	Age (years)	2	Age (years)	0		Group 3	3
Safety Population	1	Age (years)	2	Age (years)	0		Total	4
Safety Population	1	Age (years)	2	n	1	0	Group 1	1
Safety Population	1	Age (years)	2	n	1	0	Group 2	2
Safety Population	1	Age (years)	2	n	1	0	Group 3	3
Safety Population	1	Age (years)	2	n	1	0	Total	4
Safety Population	1	Age (years)	2	Mean (SD)	2		Group 1	1
Safety Population	1	Age (years)	2	Mean (SD)	2		Group 2	2
Safety Population	1	Age (years)	2	Mean (SD)	2		Group 3	3
Safety Population	1	Age (years)	2	Mean (SD)	2		Total	4
Safety Population	1	Age (years)	2	Median	3		Group 1	1
Safety Population	1	Age (years)	2	Median	3		Group 2	2
Safety Population	1	Age (years)	2	Median	3		Group 3	3
Safety Population	1	Age (years)	2	Median	3		Total	4
SNIPPED								

BIG N DATA SETS

Column header totals, that is “Big N” may vary by page. In many cases, column Big N values are the denominator for frequency listings. In the case of summaries (mean, for instance), the difference between the N of the section and the Big N indicate that subjects have missing values. The following code generates the Big N totals for the Safety Population:

```
%mac_tfl_bign
   ( in_ds                = adsl
   , treat_group_var      = arm
   , total                = Total
   , dimension_ds         = __columns
   , big_n_vars           = saffl
   , big_n_where          = saffl = "Y"
   ) ;
```

A similar macro call created the Big N values for the Intent-to-Treat population and the data sets were concatenated via a SET statement to create the data set BIG_N (**Table 6**, code not shown, but available upon request). **Appendix 3** presents the code of MAC_U_BIGN.

Table 6. The BIG N data set.

page_1	column_label	column	big_n
Intent-To-Treat Population	Group 1	1	8
Intent-To-Treat Population	Group 2	2	7
Intent-To-Treat Population	Group 3	3	1
Intent-To-Treat Population	Total	4	16
Safety Population	Group 1	1	9
Safety Population	Group 2	2	8
Safety Population	Group 3	3	1
Safety Population	Total	4	18

VARIABLE NAMES TO SECTION LABELS

Both macros to create the summary (**CON**tinuous data) and the frequency (**CAT**egorical data) have the options to create the section labels using a SAS System Format based on the variable name. The following is and example of a FORMAT procedure to achieve this:

```
proc format ;
   value $ section_format
         "agegr1"   = "Age Group, n(%)"
         "race"     = "Race, n(%)"
         "age"      = "Age (years)"
         "bmibl"    = "Baseline BMI (kg/m2)"
         ;
run ;
```

"TURNING" THE DATA: CATEGORICAL DATA

Instead of calling a macro for each section (variable), the entire data set is "turned", that is transposed. This appears to be acceptable manipulation (programming) of analysis ready data. In this case each datum or composite of datum, such as Mean and Standard Deviation, "Mean (SD)", or frequency and percent, "n (%)" requires only one (analytic) procedure. The exception is the use of ADSL to obtain the dominator data, an acceptable derivation. The following is an example, though in the experience of the author, unusual since one TF program typically creates one table. For example, even though one program can easily create an output for each population for which ADSL (or another ADAM data set) has a flag such as SAFFL and ITTFL, separate programs typically produce output for the separate tables. The PAGE dimension is usually limited to one value and achieved by subsetting, not turning.

```
data ads_cat
        ( drop   = saffl
                   ittfl
        )
     ;

  set adsl
         ( keep   = usubjid
                    agegr1
                    race
                    arm
                    saffl
                    ittfl
         )
      ;

  length page_1 $ 200 ;

  if saffl = "Y"
  then
    do ;
       page_1 = "Safety Population" ;
       output ;
    end ;

  if ittfl = "Y"
  then
    do ;
       page_1 = "Intent-To-Treat Population" ;
       output ;
    end ;

run ;
```

THE MAC_TFL_ADS_CAT MACRO

Appendix 4 presents the MAC_TFL_ADS_CAT macro (ADS is **A**nalytical **D**ata **S**et). For the first example of this paper, the following generates the frequency listing for the categorical data:

```
%mac_tfl_ads_cat
   ( in_ds           = ads_cat
   , variables       = agegr1
                       race
   , in_keep         = arm
                       page_1
   , section_format  = $section_format.
   , total           = Total
   , denom_on        =     a.page_1       = b.page_1
                       and a.column_label = b.column_label
   ) ;
```

PRECISION

The number of decimal places required usually does not reflect the number of significant digits in the original measurement in the field of clinical trials. Typically, this might rankle those with a background in physical sciences, but it is not likely to affect biostatistical outcomes or interpretation (inferences). The data set in **Table 7** is an example of the precision (number of decimal places) a shell might demand. This data set is used in the creation of the summary data (MAC_TFL_ADS_CON).

Table 7. The precision of the respective variables and their summary measures.

section_1	row_1	format
Age (years)	n	8.0
Age (years)	mean	8.0
Age (years)	median	8.0
Age (years)	stddev	8.1
Age (years)	min	8.0
Age (years)	max	8.0
Age (years)	q1	8.0
Age (years)	q3	8.0
Baseline BMI (kg/m2)	n	8.0
Baseline BMI (kg/m2)	mean	8.1
Baseline BMI (kg/m2)	median	8.1
Baseline BMI (kg/m2)	stddev	8.2
Baseline BMI (kg/m2)	min	8.1
Baseline BMI (kg/m2)	max	8.1
Baseline BMI (kg/m2)	q1	8.1
Baseline BMI (kg/m2)	q3	8.1

THE MAC_TFL_ADS_CON MACRO

Appendix 5 presents the MAC_TFL_ADS_CON macro. An example call for this paper is:

```
%mac_tfl_ads_con
   ( in_ds                = ads_con
   , in_keep              = arm
                            page_1
   , variables            = age
                            bmibl
   , section_format       = $section_format.
   , summary_stats        = n
                            mean
                            stddev
                            median
                            q1
                            q3
                            min
                            max
   , summary_rows         = section
                            n
                            mean_p_sd_p
                            median
                            q1_q3
                            min_max
   , precision_ds         = __precision
   , precision_keys       = section_1
                            row_1
   ) ;
```

THE MAC_R_PSRC MACRO

Once the datum or composite of data are generated with, or rather for, associated labels for the dimensions, for instance PAGE_1 = “Safety Population”, SECTION_1 = "Race, n(%)", and ROW_1 =

"Asian", the macro MAC_R_PSRC (PSRC is **P**age, **S**ection, **R**ow, **C**olumn; **Appendix 6**) associates those labels with their respective indices in the (structural) data sets presented earlier.

Importantly, this macro can be used to create the matrix of all values defined in the data set and establish their default value (presentation), in any. For instance, a level such as "Other" for race, might be in the shell (in the Case Report Form), but not found in the data. The default value might be "0". The macro circumvents the cumbersome IF-THEN-ELSE approach that might handle this issues.

The following "pigeon holes" the data into their respective cells, then UPDATEs the Default data set with the data (TFL_OUTSAS), replacing missing values with data, then UPDATEs the data with updated Default data set:

```
%mac r psrc
  ( out ds      = default
  , in ds       =   levels
  , in by       = page order 1
                  section order_1
                  row order_1
  , val_var     = default
  , hash_lookup = N
  ) ;


%mac r psrc
  ( out ds      = tfl outsas
  , in ds       = ctf summary
  , levels ds   =   levels
  , val_var     = value
  ) ;
```

```
data default ;
  update default
         tfl_outsas
         ;
  by page order 1
     section order_1
     row_order_1
     ;
run ;

data tfl outsas ;
  update tfl outsas
         default
         ;
  by page order 1
     section_order_1
     row_order_1
     ;
run ;
```

Table 8 shows a few rows of the DEFAULT data set (upper panel), the DEFAULT data set UPDATEd with the TFL_OUTSAS data (middle panel), and the TFL_OUTSAS data set (bottom panel) UPDATEd with the intermediate data set in the middle panel:

Table 8. The results of careful updates, including interations, of the DEFAULT and TFL_OUTSAS to obtain levels and defaults for data not present in the ADSL.

col1	col2	col3	col4	page_1	page_order_1	section_1	section_order_1	row_1	row_order_1
				Safety Population	1	Race, n(%)	1	Race, n(%)	0
0	0	0	0	Safety Population	1	Race, n(%)	1	Asian	1
0	0	0	0	Safety Population	1	Race, n(%)	1	Black	2
0	0	0	0	Safety Population	1	Race, n(%)	1	White	3
0	0	0	0	Safety Population	1	Race, n(%)	1	Other	4
0	0	0	0	Safety Population	1	Race, n(%)	1	Missing	5
				Safety Population	1	Race, n(%)	1	Race, n(%)	0
1 (11.1%)	2 (25.0%)	0	3 (16.7%)	Safety Population	1	Race, n(%)	1	Asian	1
5 (55.6%)	3 (37.5%)	0	8 (44.4%)	Safety Population	1	Race, n(%)	1	Black	2
3 (33.3%)	3 (37.5%)	1 (100.0%)	7 (38.9%)	Safety Population	1	Race, n(%)	1	White	3
0	0	0	0	Safety Population	1	Race, n(%)	1	Other	4
0	0	0	0	Safety Population	1	Race, n(%)	1	Missing	5
				Safety Population	1	Race, n(%)	1	Race, n(%)	0
				Safety Population	1	Race, n(%)	1	Race, n(%)	0
1 (11.1%)	2 (25.0%)	0	3 (16.7%)	Safety Population	1	Race, n(%)	1	Asian	1
5 (55.6%)	3 (37.5%)	0	8 (44.4%)	Safety Population	1	Race, n(%)	1	Black	2
3 (33.3%)	3 (37.5%)	1 (100.0%)	7 (38.9%)	Safety Population	1	Race, n(%)	1	White	3
0	0	0	0	Safety Population	1	Race, n(%)	1	Other	4
0	0	0	0	Safety Population	1	Race, n(%)	1	Missing	5

Whether a level is not present in the data or not present in a given group, it appears in the final table with its appropriate default value.

THE RESULTING TABLES

Table 9 presents the first table in the output. Both the derivations and the cells (order) can be checked against the data presented above. The code in **Appendix 7** generates this table using the VALDS model and associated macro presented by Viel[1,2]. Notably, adding or deleting columns (such as Group 1 + 2) or rearranging the table is a simple as updating the structural data sets (such as __COLUMNS) and adding the group to the input data set (with care needed for the correct total values.

Table 9. The resulting table.

	Safety Population			
	Group 1 (N = 9)	Group 2 (N = 8)	Group 3 (N = 1)	Total (N = 18)
Race, n(%)				
Asian	1 (11.1%)	2 (25.0%)	0	3 (16.7%)
Black	5 (55.6%)	3 (37.5%)	0	8 (44.4%)
White	3 (33.3%)	3 (37.5%)	1 (100.0%)	7 (38.9%)
Other	0	0	0	0
Missing	0	0	0	0
Age (years)				
n	9	8	1	18
Mean (SD)	56 (10.3)	48 (22.7)	66 (N/A)	53 (17.0)
Median	58	49	66	59
Q1, Q3	47, 65	26, 71	66, 66	41, 66
Min, Max	41, 66	20, 72	66, 66	20, 72
Age Group, n(%)				
18-59	5 (55.6%)	4 (50.0%)	0	9 (50.0%)
60+	4 (44.4%)	4 (50.0%)	1 (100.0%)	9 (50.0%)
Baseline BMI (kg/m2)				
n	9	8	1	18
Mean (SD)	54.3 (15.43)	45.3 (18.48)	27.5 (N/A)	48.8 (17.35)
Median	61.3	42.3	27.5	46.8
Q1, Q3	40.4, 64.2	30.3, 60.4	27.5, 27.5	32.1, 64.2
Min, Max	30.6, 75.3	22.7, 73.7	27.5, 27.5	22.7, 75.3

CAPTURING THE SAS CODE GENERATED BY THE MACRO OR DEBUGGING

The macro MAC_DEBUG_MFILE[3] can be used to capture the (unformatted) SAS code generated by a macro. This may help study the macro or investigate errors or other issues. Some regulatory agencies may not want externally defined SAS macros called in programs.

```
%mac_debug_mfile
   ( mcall = mac_tfl_ads_cat
             ( in_ds           = ads_cat
             , variables       = agegr1
                                 race
             , in_keep         = arm
                                 page_1
             , section_format  = $section_format.
             , total           = Total
             , denom_on        =     a.page_1       = b.page_1
                                 and a.column_label = b.column_label
             )
   ) ;
```

If run interactively in Windows, then the macro copies the generated SAS code to the clipboard. Typing Ctrl-V in a text editor will produce the following (unformatted) code:

CONCLUSION

This paper contributes to the automation and standardization of table and listings programs by providing macros that reduce the logic requirements of arranging derived data into pages, sections, rows, and columns. This allows the programmer to concentrate on the derivations, including accuracy and efficiency. The data sets that assign "structural" aspects, such as the column number into which a given group is assigned can be provided to a programming group, once validated. Updates, changes, and corrections, especially re-arrangements are reduced to changing the numbers associated with labels without many coding changes, especially when used in conjunction with the MAC_R_REPORT macro.

REFERENCES

[1] Viel, KR. "A Standard Data Set Format for the Validation of Tables and Listings in Regulatory Submissions for Clinical Trials." Conference Proceedings: PHUSE US Connect 2023. PP17. Available at https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/US/Florida/PAP_PP17.pdf and https://phuse.s3.eu-central-1.amazonaws.com/Archive/2023/Connect/US/Florida/POS_PP17.pdf.

[2] Viel, KR. "A Standard Dataset Format in Regulatory Submissions for Clinical Trials: Automatic Creation of Headers, Headers of Headers, and Footnotes." Conference Proceedings: PHUSE US Connect 2024. SM04. Available at https://phuse.s3.eu-central-1.amazonaws.com/Archive/2024/Connect/US/Bethesda/PAP_SM04.pdf

[3] Viel, K. 2016. "Capturing Macro Code when Debugging in the Windows Environment: The Power of MFILE and the Simplicity of Pasting." Proceedings of the PharmaSUG 2016 Conference, Denver, CO: PharmaSUG. https://www.pharmasug.org/proceedings/2016/TT/PharmaSUG-2016-TT17.pdf

CONTACT INFORMATION

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

Kevin R. Viel, Ph.D.
Navitas Data Sciences
kevin.viel@navitaslifesciences.com
www.navitasdatasciences.com

Histonis, Incorporated
kviel@histonis.org

Appendix 1. The structural data sets that assign indices (numbers) to labels for each dimension of the "matrix".

```
data __pages ;

  length page_1 $ 200 ;

  do page_1 =   "Safety Population"
              , "Intent-To-Treat Population"
              ;
    page order_1 + 1 ;
    output ;
  end ;

run ;

/**********/
data __sections ;

  length section_1 $ 200 ;

  section 1       = "Race, n(%)" ;
  section order_1 = 1 ;
  output ;

  section_1       = "Age (years)" ;
  section_order_1 + 1 ;
  output ;

  section 1       = "Age Group, n(%)" ;
  section order_1 + 1 ;
  output ;

  section_1       = "Baseline BMI (kg/m2)" ;
  section_order_1 + 1 ;
  output ;

run ;

/**********/
data __rows ;

  length section_1
         row_1
         default     $ 200
         ;

  section 1   = "Race, n(%)" ;
  row 1       = section_1 ;
  row order_1 = 0 ;
  output ;
  default     = "0" ;
  row_1       = "Asian" ;
  row_order_1 + 1 ;
  output ;
  row_1       = "Black" ;
  row_order_1 + 1 ;
  output ;
  row 1       = "White" ;
  row order_1 + 1 ;
  output ;
  row_1       = "Other" ;
  row_order_1 + 1 ;
  output ;
  row_1       = "Missing" ;
  row_order_1 + 1 ;
  output ;

  /***/
  section 1   = "Age Group, n(%)" ;
  row 1       = section_1 ;
  row_order_1 = 0 ;
  default     = " " ;
  output ;
  default     = "0" ;
  row_1       = "18-59" ;
  row order_1 + 1 ;
  output ;
  row 1       = "60+" ;
  row order_1 + 1 ;
  output ;

  do section_1 =   "Age (years)"
                 , "Baseline BMI (kg/m2)"
                 ;

    row 1       = section_1 ;
    row order 1 = 0 ;
    default     = " " ;
    output ;

    default  = "0" ;
    do row 1 =   "n"
               , "Mean (SD)"
               , "Median"
               , "Q1, Q3"
               , "Min, Max"
               ;
      row order_1 + 1 ;
      output ;
      default = " " ;
    end ;
  end ;

run ;

/**********/
data __columns ;

  length column_label $ 200 ;

  column_label = "Group 1" ;
  column       = 1 ;
```

```
111     output ;
112     column label = "Group 2" ;
113     column       + 1 ;
114     output ;
115     column label = "Group 3" ;
116     column       + 1 ;
117     output ;
118     column_label = "Total" ;
119     column       + 1 ;
120     output ;
121
122   run ;
123
124   proc sql
125      noprint ;
126     select max( column )
127               into : num_of_cols separated by ""
128     from __columns
129     ;
130   quit ;
```

Appendix 2. The MAC_TFL_DIMENSIONS macro.

```
%macro mac_tfl_dimensions
   ( pages                =
   , sections_rows        =
   , columns              =
   , dimensions code      = %str()
   , pages ds             = %str()
   , sections ds          = %str()
   , rows_ds              = %str()
   , columns_ds           = %str()
   , help                 = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint orig = %sysfunc(getoption(mprint)) ;
         options nomprint ;

         skip ;
         skip ;
         %put _____________________________________________________________________________________________________________________ ;
         %put Purpose of program:     This macro will take the dimensions data sets and create __LEVELS, which provides          ;
         %put %str(                        )the labels and their respective order variables for the tables and listing, i.e the    ;
         %put %str(                        )"shell" (multiple dimensional matrix) with cells index by the order variables.         ;
         skip ;
         %put Macro Parameter         Description                                                                                    ;
         %put _________________       ____________________________________________________________________________________________ ;


         %put pages                 = The list of pages delimited by #, with the first page prefixed with #. The page order       ;
         %put %str(                        )is the order of apparence in this list.                                               ;
         %put sections_rows         = The sections and rows ed by #, with the first section prefixed with #.  The rows are       ;
         %put %str(                        )nested in sections and delimated by |, with the first row prefixed with |. The order  ;
         %put %str(                        )is the order of apparence in this list.                                               ;
         %put columns               = The list of columns delimited by #, with the first column prefixed with #. The column order  ;
         %put %str(                        )is the order of apparence in this list.                                               ;
         %put dimensions_code       = Optional code run before the cartesian join of the dimension data sets.                      ;
         %put %str(                                )Default: %nrstr(%%)str%str(()).                                               ;
         %put pages_ds              = The pages data set (PAGE_1, PAGE_ORDER_1)                                                    ;
         %put %str(                                )Default: %nrstr(%%)str%str(()).                                               ;
         %put sections_ds           = The sections data set (SECTION_1, SECTION_ORDER_1)                                           ;
         %put %str(                                )Default: %nrstr(%%)str%str(()).                                               ;
         %put rows ds               = The rows data set (ROW 1, ROW ORDER 1)                                                       ;
         %put %str(                                )Default: %nrstr(%%)str%str(()).                                               ;
         %put columns ds            = The Columns data set (COLUMN LABEL, COLUMN)                                                  ;
         %put %str(                                )Default: %nrstr(%%)str%str(()).                                               ;
         %put _____________________________________________________________________________________________________________________ ;
         %put End of help                                                                                                          ;

         options &mprint_orig. ;

         %goto __END ;
     %end ;

  %global num_of_cols ;

  /*************************************************************************************************/
  %if &pages_ds. = %str()
  %then
     %do ;
         data   pages
            ( drop   = r_s ) ;

           length r_s          $ 30000
                  page_1       $   200
                  page_order_1 8
                  ;

           r_s = %unquote(%str(%')&pages.%str(%')) ;

           do page_order_1 = 1 to countc( r_s , "#" ) ;
             page_1 = strip( scan( r_s , page_order_1 , "#" )) ;
             output ;
           end ;

         run ;

         %let pages_ds = __pages ;

     %end ;

  /****/
  %if     &sections_ds. = %str()
      or  &rows_ds.     = %str()
  %then
     %do ;
         data %if &sections_ds. = %str()
              %then
                sections
                  ( keep   = section: )
              ;
              %if &rows_ds. = %str()
              %then
              __rows
                 ( keep   = section:
                            row:
                            default
                 )
              ;
              ;

           length r_s       $ 30000
                  text      $ 10000
                  section_1
                  %if &rows_ds. = %str()
                  %then row_1
                        default
                        ;
                            $   200
                  ;

```

```
            r_s = %unquote(%str(%')&sections_rows.%str(%')) ;

            do section_order_1 = 1 to countc( r_s , "#" ) ;

              text = scan( r_s , section_order_1 , "#" ) ;

              section_1 = strip( scan( text , 1 , "|" )) ;

              %if &sections_ds. = %str() %then output __sections %str(;) ;
              %if &rows_ds. = %str()
              %then
                 %do ;
                     do row order 1 = 1 to countc( text , "|" ) ;
                       row 1   = strip( scan( scan( text , row order 1 + 1 , "|" ) , 1 , "~" )) ;
                       default = strip( scan( scan( text , row_order_1 + 1 , "|" ) , 2 , "~" )) ;
                       output __rows ;
                     end ;
                 %end ;
            end ;

          run ;

          %if &sections ds. = %str() %then %let sections ds =    sections ;
          %if &rows_ds.     = %str() %then %let rows_ds     =   __rows     ;

       %end ; /* END OF sections_ds = str() or rows_ds = str() */

    /****/
    %if &columns_ds. = %str()
    %then
       %do ;
          data   columns
                  ( drop   = r_s ) ;

            length r_s          $ 1000
                   column_label $  200
                   ;

            r_s = %unquote(%str(%')&columns.%str(%')) ;

            do column = 1 to countc( r_s , "#" ) ;
              column label = strip( scan( r_s , column , "#" )) ;
              output ;
            end ;

            call symputx( "num_of_cols"
                        , put( column , 8.0 )
                        ) ;

          run ;

          %let columns_ds = __columns ;

       %end ;

    &dimensions_code.

    /*************/
    /* Cartesian */
    /*************/
    proc sql ;
      create table __levels as
      select a.*
           , b.*
           , c.*
      from &pages_ds. as a
         , ( select aa.*
                  , bb.row 1
                  , bb.row order_1
                  , bb.default
             from &sections ds.    as aa
                , &rows_ds.        as bb
             where aa.section_1 = bb.section_1
           ) as b
         , &columns_ds. as c
      order by a.page_order_1
             , b.section order_1
             , b.row order_1
             , c.column
      ;
    quit ;

    %__END:

%mend  mac_tfl_dimensions ;
```

Appendix 3. The MAC_TFL_BIGN macro.

```
%macro mac_tfl_bign
   ( in_ds                =
   , treat_group_var      =
   , total                = %str()
   , dimension ds         = %str()
   , dimension label      = column_label
   , dimension order      = column
   , big_n_vars           = %str()
   , big_n_where          = %str()

   , help                 = N
   ) ;

  %if &help. = Y
  %then
     %do ;
         %let mprint orig = %sysfunc(getoption(mprint)) ;
         options nomprint ;

         skip ;
         skip ;
         %put _____________________________________________________________________________________________________________________ ;
         %put Purpose of program:     This program will compute the Big N (column header or section header for instance) values for a given  ;
         %put %str(                        )treatment variable (column), column data set, and optional subsetting condition.              ;
         skip ;
         %put Macro Parameter         Description                                                                                          ;
         %put                                                                                                                              ;
         %put in_ds                 = Input data set.                                                                                      ;
         %put treat_group_var       = Variable defining the groups, i.e. the columns.                                                      ;
         %put total          = The name of the summary column.  If this value is not populated, then the column                            ;
         %put %str(                              )Default: %nrstr(%%)str%str(()).                                                         ;
         %put dimension_ds          = The data set that contains the dimension labels and their respective ordinal numbers.                ;
         %put %str(                              )Default: %nrstr(%%)str%str(()).                                                         ;
         %put big n vars            = Additional variables to keep from IN DS (for BIG N WHERE).                                           ;
         %put %str(                              )Default: %nrstr(%%)str%str(()).                                                         ;
         %put big n where           = Options WHERE data set option to IN DS.                                                              ;
         %put %str(                              )Default: %nrstr(%%)str%str(()).                                                         ;
         %put _____________________________________________________________________________________________________________________ ;
         %put End of help                                                                                                                  ;


         options &mprint_orig. ;

         %goto __END ;
     %end ;

  /*************** Big N *********************/
  data big_n
          %if %nrbquote(&big_n_vars.) ne %str() %then ( drop   = &big_n_vars. ) ;
       ;

    length &dimension_label. $ 200 ;

    set &in ds.
          ( keep    = studyid
                      usubjid
                      &treat_group_var.
                      &big_n_vars.
            rename  = ( &treat_group_var. = &dimension_label. )

            %if %nrbquote(&big n where.) ne %str()
            %then where  = ( &big_n_where. ) ;

          )
       ;

    %if &total. ne %str()
    %then
       %do ;
          output ;
          &dimension_label. = "&total." ;
          output ;
       %end ;

  run ;

  proc sql
     undo_policy = none ;
    create table big_n as
    select a.*
         , case when b.&dimension_label. ne " " then b.big_n
                else 0
           end as big n
    from &dimension_ds. as a
         left join ( select &dimension_label.
                           , count( distinct catx( "/"
                                                 , studyid
                                                 , usubjid
                                                 )
                                  ) as big_n
                     from big n
                     group by &dimension_label.
                   ) as b
           on a.&dimension_label. = b.&dimension_label.
    order by a.&dimension_order.
    ;
  quit ;


  %__END:

%mend  mac_tfl_bign ;
```

Appendix 4. The MAC_TFL_ADS_CAT macro.

```
%macro mac_tfl_ads_cat
   ( dimensions      = page_1
                       section_1
                       row 1
                       column_label
   , in ds           =
   , in keep         = %str()
   , out_keep        = %str()
   , variables       =
   , rename          = arm = column_label
   , where           = %str()
   , code            = %str()
   , section_format  = $200.
   , subject id      = usubjid
   , total           = Total
   , denom on        = a.column_label = b.column_label
   , include missing = N
   , debug           = N
   ) ;

  data ads_cat
          ( keep   = &subject_id.
                     &dimensions.
                     value
                     %if %nrbquote(&out_keep.) ne %str() %then &out_keep. ;
          )
       ;

    length &dimensions.
           value        $ 200
           ;


    set &in ds.
           ( keep   = &subject id.
                      &variables.
                      %if %nrbquote(&in_keep.) ne %str() %then &in_keep. ;
             rename = ( &rename. )

             %if %nrbquote(&where.) ne %str()
             %then where  = ( &where. ) ;

           )
       ;

    &code.

    array __v ( * )
          &variables.
          ;

    do  n  = 1 to dim(   v ) ;
      row 1     =   v(  n  ) ;
      section 1 = put( lowcase( vname( __v( _n_ ))) , &section_format. ) ;
      value     = "Y" ;
      output ;
    end ;

  run ;

  %if &total. ne %str()
  %then
     %do ;
         data ads cat ;
           set ads cat ;
           output ;
           column_label = "&total." ;
           output ;
         run ;
     %end ;

  /******/
  %let dimension_number = %eval( %sysfunc( countc( %nrbquote(%sysfunc( compbl( &dimensions. ))) , %str( ))) + 1 ) ;

  ods listing close ;

  proc freq
     data = ads_cat ;

    ods output CrossTabFreqs = ctf
                                   ( keep   = &dimensions.
                                              value
                                              type
                                              frequency
                                     where  = (     _type_ = "%sysfunc( repeat( 1 , &dimension_number.))"
                                                and value  = "Y"
                                              )
                                   )
               ;

    tables %sysfunc( prxchange( s/\s+/%str(*)/
                              , -1
                              , &dimensions.
                              )
                   )
          * value
          / nocol
            norow
            nopercent
            %if &include_missing. = Y %then missing ;
            ;

  run ;

  ods output close ;
  ods listing ;

  %if &debug. = N
  %then
     %do ;
         proc datasets
```

```
112             library = WORK
113             nolist
114             ;
115           delete ads_cat ;
116         quit ;
117       %end ;
118
119     proc sql ;
120       alter table ctf
121       drop value
122          , _type_
123       ;
124     quit ;
125
126     proc sql
127        undo_policy = none
128        ;
129       create table ctf as
130       select a.*
131            , case when     nmiss( a.frequency
132                                 , b.big_n
133                                 ) = 0
134                        and b.big_n ne 0
135                   then a.frequency / b.big_n * 100
136                    else .
137             end as percent
138       from ctf              as a
139            left join big n  as b
140               on &denom_on.
141       ;
142     quit ;
143
144     %if &debug. = N
145     %then
146        %do ;
147            proc datasets
148               library = WORK
149               nolist
150               ;
151             delete big_n ;
152            quit ;
153         %end ;
154
155     data ctf
156            ( drop   = frequency
157                       percent
158                       __:
159            )
160          ;
161
162       length value $ 200
163              __1
164              __2   $  20
165              ;
166
167       set ctf ;
168
169       if frequency ne .
170       then __1 = put( frequency , 8.0 ) ;
171
172       if percent ne .
173       then __2 = put( percent , 8.1 ) ;
174
175       if cmiss( __1
176               , __2
177               ) = 0
178       then
179         do ;
180            if   2 = put( 0 , 8.1 )
181            then value = strip(   1 ) ;
182             else value = cat( strip( __1 )
183                             , " ("
184                             , strip( __2 )
185                             , "%)"
186                             ) ;
187         end ;
188
189     run ;
190
191     proc sort
192        data = ctf ;
193       by %sysfunc( prxchange( s/column_label//i
194                             , -1
195                             , &dimensions.
196                             )
197                  )
198         ;
199     run ;
200
201   %mend  mac tfl ads cat ;
```

Appendix 5. The MAC_TFL_ADS_CON macro.

```
%macro mac tfl ads con
   ( dimensions           = page 1
                            section_1
                            column_label
   , in_ds                =
   , in_keep              = %str()
   , out keep             = %str()
   , variables            =
   , rename               = arm = column_label
   , where                = %str()
   , code                 = %str()
   , section format       = $200.
   , transpose_doloop_code = %str( )
   , summary_stats        = n
                            mean
                            stddev
                            median
                            q1
                            q3
   , summary stats add    = %str()
   , summary rows         = section
                            n
                            mean_p_sd_p
                            median_p_q1_q3_p
   , summary rows add     = %str()
   , subject id           = usubjid
   , total                = Total
   , class                = page 1
                            section 1
                            column_label
   , precision_ds         = %str()
   , precision_keys       = %str()
   , precision_format_var = format
   , not_applicable       = N/A
   , debug                = N
   ) ;

  /******/
  data ads con
          ( keep   = &subject_id.
                     &dimensions.
                     value
                     %if %nrbquote(&out_keep.) ne %str() %then &out_keep. ;
          )
       ;

    length &dimensions. $ 200
           ;

    set &in_ds.
          ( keep   = &subject_id.
                      &variables.
                      %if %nrbquote(&in_keep.) ne %str() %then &in_keep. ;
            rename = ( &rename. )
            %if %nrbquote(&where.) ne %str()
            %then where  = ( &where. ) ;
          )
       ;

    &code.

    array __v ( * )
          &variables.
          ;

    do  n  = 1 to dim(   v ) ;
      %if %nrbquote(&transpose_doloop_code.) = %str()
      %then
         %do ;
             section_1 = put( lowcase( vname( __v( _n_ ))) , &section_format. ) ;
             value     = __v( _n_ ) ;
         %end ;
       %else
          %do ;
              &transpose_doloop_code.
          %end ;

      output ;
    end ;

  run ;

  %if &precision_ds. = %str()
  %then
     %do ;
         %mac u precision
            ( in ds           = ads con
            , out_ds          = precision
            , group           = &precision_keys.
            , var             = value
            , varu            = %str()
            , format          = best32.
            , precision_var   = format
            ) ;
     %end ;

  %if &total. ne %str()
  %then
     %do ;
         data ads con ;
           set ads con ;
           output ;
           column label = "&total." ;
           output ;
         run ;
     %end ;

  /****************/
```

```
109    ods listing close ;
110
111    proc means
112       data = ads con
113       &summary stats.
114       &summary_stats_add.
115       stackodsoutput
116       ;
117
118      ods output summary = summary ;
119
120      class &class. ;
121
122      var value ;
123
124    run ;
125
126    ods output close ;
127    ods listing ;
128
129    %if &debug. = N
130    %then
131       %do ;
132           proc datasets
133              library = WORK
134              nolist
135              ;
136             delete ads_con ;
137           quit ;
138       %end ;
139
140    /*****/
141    %if %nrbquote(&precision_keys.) ne %str()
142    %then
143       %do ;
144           data _null_ ;
145             call symput( "precision_keys"
146                        , cat( '"'
147                             , prxchange( 's/\s+/" , "/'
148                                        , -1
149                                        , compress( compress( "&precision_keys." , '"' ) , "'" )
150                                        )
151                             , '"'
152                             )
153                        ) ;
154           run ;
155
156           data  null  ;
157             call symput( "precision keys2"
158                        , cats( prxchange( "s/\s+/= /"
159                                        , -1
160                                        , compress( %unquote( %str(%')&precision_keys.%str(%')) , '",' )
161                                        )
162                              , "="
163                              )
164                        ) ;
165           run ;
166       %end ;
167
168    /***/
169    data summary
170           ( keep   = &dimensions.
171                      row_1
172                      value
173          )
174       ;
175
176      %if &precision_ds. ne %str()
177      %then
178         %do ;
179
180             if 0 then set &precision_ds. ;
181
182             if _n_ = 1
183             then
184               do ;
185                  dcl hash   format
186                             ( dataset: "&precision_ds." )
187                           ;
188
189                  __rc = __format.definekey
190                               ( &precision_keys. ) ;
191                  __rc = __format.definedata
192                               ( "&precision_format_var." ) ;
193                  __rc = __format.definedone() ;
194               end ;
195         %end;
196
197      set summary
198             ( drop   = nobs
199                        _control_
200                        variable
201             )
202          ;
203
204      length row_1
205             value $ 200
206               1
207               2
208             %if     %sysfunc( prxmatch( /\bmedian p min max p\b/i , &summary rows. ))
209                 or  %sysfunc( prxmatch( /\bmedian_p_q1_q3_p\b/i   , &summary_rows. ))
210             %then __3 ;
211                   $ 100
212             ;
213
214      array   sumstats
215            ( * )
216            &summary stats.
217            &summary_stats_add.
218            ;
219
220      array __sumstatsf
221            ( * )
222            $ 100
```

```
223             __%sysfunc( prxchange( s/ / __/
224                                  , -1
225                                  , %sysfunc( compbl( &summary_stats. &summary_stats_add.))
226                                  )
227                       )
228             ;
229
230         do _n_ = 1 to dim( __sumstats ) ;
231           row_1 = lowcase( vname( __sumstats( _n_ ))) ;
232           __rc = __format.find() ;
233
234           if  _rc ne 0
235           then
236             do ;
237                put "W" "ARNING: "
238                    &precision_keys2.
239                    ;
240                &precision_format_var. = "8.0" ;
241             end ;
242             if  _sumstats( _n_ ) ne .
243             then
244               do ;
245                    sumstatsf(  n  ) = left( putn(   sumstats(  n  ) , &precision format var. )) ;
246                  if prxmatch( "/-0(?:\.0+)?$/" , strip( __sumstatsf( _n_ ))) then __sumstatsf( _n_ ) = substr( __sumstatsf( _n_ ) , 2 ) ;
247             end ;
248         end ;
249
250         %if %sysfunc( prxmatch( /\bsection\b/i , &summary_rows. ))
251         %then
252            %do ;
253                row_1 = section_1 ;
254                value = " " ;
255                output ;
256            %end ;
257
258         %if %sysfunc( prxmatch( /\bn\b/i , &summary_rows. ))
259         %then
260            %do ;
261                row_1 = "n" ;
262                value = __n ;
263                output ;
264            %end ;
265
266         %if %sysfunc( prxmatch( /\bsection_n\b/i , &summary_rows. ))
267         %then
268            %do ;
269                row_1 = section_1 ;
270                value = __n ;
271                output ;
272            %end ;
273
274         %if %sysfunc( prxmatch( /\bmean_p_sd_p\b/i , &summary_rows. ))
275         %then
276            %do ;
277                row_1 = "Mean (SD)" ;
278                if mean   ne . then __1 = __mean ;
279                if stddev ne . then __2 = __stddev ;
280                 else   2 = "&not_applicable." ;
281                value = catx( " "
282                            ,   1
283                            , cats( "("
284                                  , __2
285                                  , ")"
286                                  )
287                            ) ;
288                output ;
289            %end ;
290
291         %if %sysfunc( prxmatch( /\bmean_p_pm_stderr_p\b/i , &summary_rows. ))
292         %then
293            %do ;
294                row_1 = "Mean (+/-SEM)" ;
295                if mean   ne . then __1 = __mean ;
296                if stderr ne .
297                then
298                  do ;
299                       2 =   stderr ;
300                    value = catx( " "
301                                ,   1
302                                , cats( "(+/-"
303                                      , __2
304                                      , ")"
305                                      )
306                                ) ;
307                  end ;
308                 else value = catx( " "
309                                  ,  1
310                                  , "(&not_applicable.)"
311                                  ) ;
312                output ;
313            %end ;
314
315         %if %sysfunc( prxmatch( /\bmean\b/i , &summary_rows. ))
316         %then
317            %do ;
318                row_1 = "Mean" ;
319                value = __mean ;
320                output ;
321            %end ;
322
323         %if %sysfunc( prxmatch( /\bsd\b/i , &summary_rows. ))
324         %then
325            %do ;
326                row_1 = "SD" ;
327                if stddev ne . then value =   stddev ;
328                 else value = "&not_applicable." ;
329                output ;
330            %end ;
331
332         %if %sysfunc( prxmatch( /\bmedian_p_min_max_p\b/i , &summary_rows. ))
333         %then
334            %do ;
335                row_1 = "Median (Min, Max)" ;
336                if median ne . then   1 =   median ;
```

```
            else __1 = " " ;
           if min    ne . then __2 = __min ;
            else   2 = " " ;
           if max    ne . then __3 = __max ;
            else   3 = " " ;
           value = cat( strip( __1 )
                      , " ("
                      , strip( __2 )
                      , ", "
                      , strip( __3 )
                      , ")"
                      ) ;
           output ;
       %end ;

    %if %sysfunc( prxmatch( /\bmedian_p_q1_q3_p\b/i , &summary_rows. ))
    %then
       %do ;
           row_1 = "Median (Q1, Q3)" ;
           if median ne . then __1 = __median ;
            else   1 = " " ;
           if q1    ne . then __2 = __q1 ;
            else   1 = " " ;
           if q3    ne . then __3 = __q3 ;
            else   3 = " " ;
           value = cat( strip( __1 )
                      , " ("
                      , strip( __2 )
                      , ", "
                      , strip( __3 )
                      , ")"
                      ) ;
           output ;
       %end ;

    %if %sysfunc( prxmatch( /\bmedian\b/i , &summary_rows. ))
    %then
       %do ;
           row_1 = "Median" ;
           if median ne . then value = __median ;
           output ;
       %end ;

    %if %sysfunc( prxmatch( /\bp_min_max_p\b/i , &summary_rows. ))
    %then
       %do ;
           row_1 = "(Min, Max)" ;
           if min    ne . then __1 = __min ;
            else   1 = " " ;
           if max    ne . then __2 = __max ;
            else   2 = " " ;
           value = cat( " ("
                      , strip( __1 )
                      , ", "
                      , strip( __2 )
                      , ")"
                      ) ;
           output ;
       %end ;

    %if %sysfunc( prxmatch( /\bmin_max\b/i , &summary_rows. ))
    %then
       %do ;
           row_1 = "Min, Max" ;
           if min    ne . then __1 = __min ;
            else __1 = " " ;
           if max    ne . then __2 = __max ;
            else   2 = " " ;
           value = cat( strip( __1 )
                      , ", "
                      , strip( __2 )
                      ) ;
           output ;
       %end ;

    %if %sysfunc( prxmatch( /\bq1_q3\b/i , &summary_rows. ))
    %then
       %do ;
           row_1 = "Q1, Q3" ;
           if q1    ne . then __1 = __q1 ;
            else __1 = " " ;
           if q3    ne . then __2 = __q3 ;
            else __2 = " " ;
           value = cat( strip( __1 )
                      , ", "
                      , strip( __2 )
                      ) ;
           output ;
       %end ;

  run ;

  proc sort
     data = summary ;
    by %sysfunc( prxchange( s/column_label//i
                          , -1
                          , &dimensions.
                          )
               )
       row_1
       ;
  run ;

%mend  mac_tfl_ads_con ;
```

Appendix 6. The MAC_R_PSRC macro.

```
%macro mac_r_psrc
   ( out_ds          =
   , in_ds           =
   , in_by           = page_1
                       section_1
                       row_1
   , levels_ds       =   levels
   , levels          = page_order_1
                       section_order_1
                       row_order_1
   , column_hash_key = page_1
   , val_var         = value
   , hash_lookup     = Y
   ) ;

   %if     &hash_lookup.               =  Y
       and %nrbquote(&column_hash_key.) ne %str()
       and (     %sysfunc( indexc( %nrbquote(&column_hash_key.) , %str(%'%"))) =  0
             or  (     %sysfunc( indexc( %nrbquote(&column_hash_key.) , %str( )))
                   and %sysfunc( indexc( %nrbquote(&column_hash_key.) , %str(,))) = 0
                 )
           )
   %then
      %do ;

          %let column_hash_key = %qsysfunc( compress( &column_hash_key. , %str(%'%"))) ;

          data _null_ ;

            call symput( "column_hash_key"
                       , cats( '"'
                             , prxchange( 's/\s+/" , "/'
                                        , -1
                                        , "&column_hash_key."
                                        )
                             , '"'
                             )
                       ) ;
            stop ;
          run ;

      %end ;

   /***/
   data &out_ds.
             ( drop = column_label
                      column
               keep = page_:
                      section_:
                      row_:
                      col:
             ) ;

     %if &hash_lookup. = Y
     %then
        %do ;

           if 0 then set &levels_ds. ;

           if  n_ = 1
           then
             do ;
                /* (page) label to page_order_n */
                declare hash l2pn
                          ( ) ;
                __rc = l2pn.DefineKey
                          ( "page_1" ) ;
                __rc = l2pn.DefineData
                          ( "page order 1" ) ;
                __rc = l2pn.DefineDone() ;


                /* (section) label to section_order_n */
                declare hash l2sn
                          ( ) ;
                __rc = l2sn.DefineKey
                          ( "section_1" ) ;
                __rc = l2sn.DefineData
                          ( "section order_1" ) ;
                __rc = l2sn.DefineDone() ;

                /* (row) label to row_order_1 */
                declare hash l2r1
                          (  ) ;
                __rc = l2r1.DefineKey
                          ( "page_1"
                          , "section_1"
                          , "row_1"
                          ) ;
                  rc = l2r1.DefineData
                          ( "row order 1" ) ;
                __rc = l2r1.DefineDone() ;

                /* (column) label to column (number) */
                declare hash l2c
                         ( ) ;
                __rc = l2c.DefineKey
                          ( %if %nrbquote(&column_hash_key.) ne %str()
                            %then &column hash_key. , ;
                            "column_label"
                          ) ;
                  rc = l2c.DefineData
                          ( "column" ) ;
                __rc = l2c.DefineDone() ;

                do until ( end1 ) ;
                  set &levels_ds.
                      end = end1
                      ;
                  by &levels. ;

```

```
                  if first.page_order_1    then __rc = l2pn.ref() ;
                  if first.section order 1 then   rc = l2sn.ref() ;
                  if first.row order_1     then __rc = l2r1.ref() ;
                  __rc = l2c.ref() ;

                end ;

                /*********/
                call missing( page_order_1
                            , page 1
                            , section order_1
                            , section 1
                            , row order_1
                            , row 1
                            , column
                            , column_label
                            ) ;

             end ; /* END OF  n  = 1 */
       %end ; /* END OF hash_lookup = Y */

    array col( &num_of_cols. ) $ 200 ;

    do until ( last.%scan( &in_by. , -1 , %str( )) ) ;
      set &in ds. ;
      by &in_by. ;

      /***/
      %if &hash_lookup. = Y
      %then
         %do ;
              rc = l2pn.find() ;
            if   rc ne 0
            then put "W" "ARNING: l2pn "
                 page_1=
                 ;

              rc = l2sn.find() ;
            if   rc ne 0
            then put "W" "ARNING: l2sn "
                 section_1=
                 ;

            __rc = l2r1.find() ;
            if __rc ne 0
            then put "W" "ARNING: l2r1 "
                 page_1=
                 section_1=
                 row_1=
                 ;

            __rc = l2c.find() ;
            if __rc ne 0
            then put "W" "ARNING: l2c "
                 page_1=
                 section_1=
                 column_label=
                 ;

            if __rc = 0 then col( column ) = &val_var. ;

         %end ;
       %else col( column ) = &val_var. %str(;) ;
    end ;

  run ;

  proc sort
     data = &out ds. ;
    by page order 1
       section order_1
       row_order_1
       ;
  run ;

%mend  mac_r_psrc ;
```

Appendix 7. An example of a REPORT_DEFINE_OPTIONS data set and a call to MAC_R_REPORT.

```
data report_define_options ;

  length page_1         $ 200
         variable       $  32
         define options $ 200
         cellwidth      $  10
         ;

  do page_1 =   "Safety Population"
              , "Intent-To-Treat Population"
              ;

    order       = 0 ;
    cellwidth   = " " ;

    /***/
    define options = "order order = data noprint" ;
    do variable =   "page_order_1"
                  , "page_1"
                  , "section_order_1"
                  , "row_order_1"
                  ;
      order + 1 ;
      output ;
    end ;

    /***/
    variable = "row_1" ;
    order + 1 ;
    define_options = compbl( '" "
                             order
                             style ( column ) = { cellwidth       = 30%
                                                  just            = l
                                                  vjust           = top
                                                }
                             style ( header ) = { just            = l }
                            '
                           ) ;
    output ;

    /***/
    define options =  " " ;
    cellwidth   = "15%" ;
    do  n  = 1 to 4 ;
      order + 1 ;
      variable  = cats( "col" , put( _n_ , 8.0 )) ;
      output ;
    end ;

  end ; /* CYCLED THROUGH page_1 */

run ;

proc sort
   data = tfl_outsas ;
  by page order_1
     type_order
     descending header_order
     section_order_1
     row_order_1
     ;
run ;

/********************/
options nobyline ;

title j = c "#byval( page_1 )" ;

ods listing close ;
ods rtf
    file  = "&path.\tfl_outsas.rtf"
    ;

%mac r report
   ( in_ds                      = tfl_outsas
   , page_by_vars               = page_1
   , num_of_cols                = &num_of_cols.
   , report_define_options_ds   = report_define_options
   , report options             = style( report ) = { width             = 100%
                                                      frame             = hsides
                                                      rules             = groups
                                                      cellpadding       = 1pt
                                                      cellspacing       = 0pt
                                                      borderwidth       = 1
                                                      fontfamily        = "Courier New"
                                                      fontsize          = 9pt
                                                      backgroundcolor   = white
                                                    }
                                  style( header ) = { backgroundcolor   = white
                                                      fontweight        = medium
                                                    }
                                  style( column ) = { just              = c
                                                      vjust             = m
                                                      backgroundcolor   = white
                                                    }
                                  missing

   , code_after_define          = compute before section_order_1
                                         / style = { height = 18pt }
                                           ;
                                     line " " ;
                                  endcomp ;

                                  compute row_1 ;

                                     if row_order_1 > 0
                                     then call define ( _col_
                                                      %str(,) "style"
                                                      %str(,) "style = { leftmargin = 1% }"
                                                      ) ;

```

```
112                                 endcomp ;
113
114     , report_by                 = page_1
115     ) ;
116
117   ods rtf close ;
118   ods listing ;
119
120   title " " ;
121   footnote ;
```