PhUSE 2016

Paper AD02

ICON's DIAMOND or an Enhanced Way to Automated Statistical Reporting

Nathalie Fuchs, ICON Clinical Research Ltd, Illkirch, France

ABSTRACT

Statistical reporting of safety and efficacy data for clinical studies is a key component of the Clinical Data Life Cycle. Over the past years, pharmaceutical and clinical research organizations have made many attempts to automate this process by developing standard generic SAS® macro library solutions in-house or purchasing vendor packages.

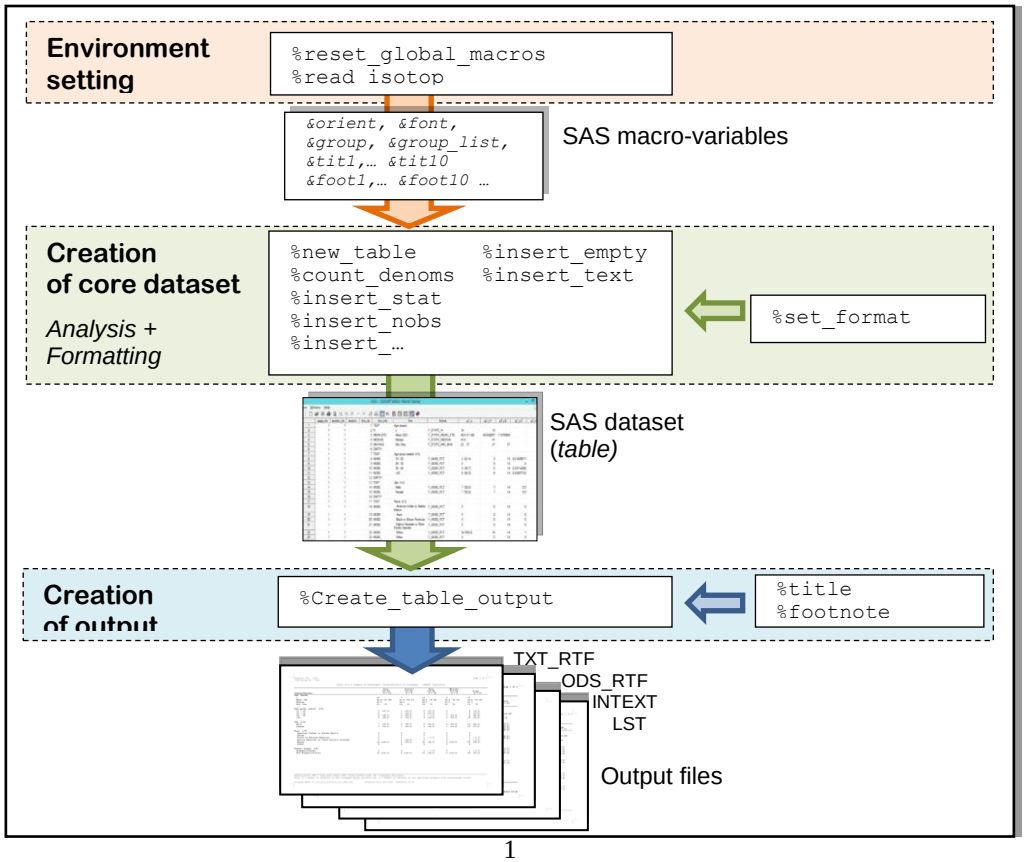
ICON's DIAMOND is a flexible, modular suite of SAS® macro-functions that provides various features for statistical Tables and Listings generation. Each DIAMOND macro-function is designed to perform a specific function and, when logically grouped together, supports each step of output development, including a flexible approach to creating a reporting dataset prior to application of the presentation layer. Environment settings and presentation features are initialized using global macro-variables. The modular macros are used as building blocks to create the majority of table designs, with minimal additional processing. DIAMOND reduces programming time and effort and improves the quality of statistical reports.

INTRODUCTION

Every statistical report is built in 3 main steps:

- the environment setting with the creation of both global report-level macro-variables and specific study-level macro-variables
- the creation of the output dataset with the analysis values
- the creation of the output itself

With ICON's DIAMOND, each of these steps will have their dedicated DIAMOND macro-functions created with an intuitive name. A standard report dataset, called *table*, will be the core of the process.
This can be summarized with the following schema listing the relevant DIAMOND macro-functions. A list of all current macro-functions is given in the Appendix.



In this paper, we will go through each of these steps to give you a brief overview of how the DIAMOND macros work. The aim is not to give you a full documentation of their capability but to provide a general understanding of their functionality. For any specific question, feel free to contact the author.

EXAMPLE AS INTRODUCTION

To give an additional flavor of how the DIAMOND macros work, see the example below without paying too much attention to details. A brief definition is given afterwards.

```
*** Environment setting ***;
%reset_global_macros;

%let rep = T_example;
%read_isotop;

%let group = TRTA;
%let group_list = A@B;

*** Creation of core dataset (table) ***;
%new_table
%count_denoms(DS=dataset, WHR= SAFFL eq 'Y');

%insert_nobs(DS=dataset, TX='Safety Set', WHR=SAFFL eq 'Y', pct=0);
%insert_empty;
%insert_nobs(DS=dataset, TX='Lead-in Safety Set', WHR=SAF01FL eq 'Y');

*** Creation of output ***;
%create_table_output(
    line_width            = 60
   ,group_width           = 16
   ,line_header_txt       = 'Population'
   ,group_denom_in_header = 1
 );
```

First, the global report-level **macro-variables** will be set to their default values with the macro-function `%reset_global_macros`. The name of the output file will be given with the *&rep* macro-variable and the associated macro-variables *&tit1, &tit2,… and &foot1, &foot2*,… retrieved from an ISOTOP.xml file with the macro-function `%read_isotop`.
The **treatment columns** will be defined with the macro-variables *&group/&group_list,* applying the conditions TRTA='A' (1st column) and TRTA='B' (2nd column).
The **core dataset**, called *table*, will be created with an empty structure and completed, step by step, with the subsequent `%insert_nobs` and `%insert_empty` macro-functions.
Before any analysis, the **population** (denominator) will be counted and stored in the macro-variable *&denom_1, &denom_2* and *&denom_t* (total) with the macro-function `%count_denoms`.
Analysis will be performed with the macro-functions `%insert_nobs` which will complete at the same time the core dataset *table*. For customization purpose, an empty line will be inserted in the core dataset *table* with the macro-function `%insert_empty`.
Finally, the **output** will be created with the macro-function `%create_table_output`.

ENVIRONMENT SETTING

When starting a DIAMOND session, the environment needs to be set at both study-level (e.g., libname, paths, etc.) and report-level (e.g., name of the output, titles, and footnotes). Regardless of the creation of the libraries and the link to the DIAMOND macro-functions (see Appendix 1 for an example of set-up file), with DIAMOND and probably all other tools, this is done with global macro-variables.

STUDY-LEVEL

At study-level, default values of each global study-level macro-variables are given at initialization of the session with the call of the DIAMOND macro-functions `%reset_global_macros` and can be updated throughout the study.
However, at each re-call of the `%reset_global_macros` without parameters, each global macro-variable will be reassigned to their default values, given at 1st call of `%reset_global_macros`.

```
%reset_global_macros(
  def_group        = TRTA,   /*report-level*/
  def_group_list   = A@B,
  def_totals       = 0,
  def_subjno       = USUBJID,

  def_page_format= A4,       /*pagination*/
  def_orient     = LS,
  def_font       = 0,
  def_fontsize   = 8,
  def_LS_ls      = 151,
  def_LS_ps      = 52,
  def_LS_marginT = 2 cm,
  def_LS_marginB = 2 cm,
  def_LS_marginL = 2 cm,
  def_LS_marginR = 2 cm,

  def_tit_ver      =ICON,       /*layout*/
  def_foot_ver     =ICON,
  def_layout_center=nocenter,

  def_folder_structure     = ICON,      /*folder, file*/
  def_output_path          = &path\dev\programs,
  def_automatic_log        = 0,
  def_automatic_log_check  = 0
  );
```

All parameters are quite intuitive, except the 1st group which defines the default report-level options. These options can be set by default for the whole study or at a report-level depending on the output. See below for some precision or contact the author for the full list of parameters.
The option `def_subjno=` specifies the variable which gives a unique identifier for the subject to be able to calculate number of subjects vs number of events.
Note that currently Diamond is developed on a Windows® operating system but intended to be platform independent.

REPORT-LEVEL

At report-level, the initialization values are given straight with a `%let` statement and should be at least:

`%let rep=name_of_output;`	To specify the name of the output file
`%let group=variable;` `%let group_list=value1@value2;`	To specify the variable and associated modalities which will define the treatment columns. Each value of *&group_list*, separated with @, will create a condition (i.e. variable="value1") which will create a column in the output. The column will be created with respective order of *&group_list*. Default values can also be given with `%reset_global_macros`.
`%let total=0;`	To specify if a total column must be printed in the final output, with the combination/sum of all other columns. This column is always calculated but not necessarily printed. Default values can also be given with `%reset_global_macros`.
`%let output_mode=ODS_RTF;`	To specify the type of output to be produced (see later, paragraph output file – type of outputs)
`%let output_dataset=1;`	To specify if the core dataset *table* needs to be kept in a specific folder at the end of the analysis.

Titles and footnotes are also created using global macro-variables &tit1, &tit2,… &foot1,&foot2,… at the report-level which will be combined with study-level information with the call of the DIAMOND macro-functions `%titles` and `%footnotes`. These macro-variables tit1, tit2,… can be given straight or retrieved from an ISOTOP.xml file with the call of the DIAMOND macro-function `%read_isotop`.

CORE DATASET

Before having a look to the DIAMOND macro-functions themselves, it's useful to understand what the core dataset, or output dataset, contains and how it is built.
The core dataset, called *table*, will have the same structure and standardized variables across the whole study and reports. At start of the creation of an output, an empty structure will be created with every relevant variable needed

throughout the report. Each DIAMOND call will then add one or several rows to this dataset with respect of the final formatting (right format of the treatment columns, additional empty line,…) and the sorting order.

SAS - [VIEWTABLE: Work.Table]

Window Help

	page_nb	line_nb	line_info	line	format	g1_c	g1_n1	g1_n2	g1_n3	g1_n4	g1_n5	g1_n6	g2_c	g2_n1	g2_n2
1	1	1	NOBS	Safety Set	F_NOBS_NOPCT	12	12	12	1				4	4	4
2	1	2	EMPTY												
3	1	3	NOBS	Lead-in Safety Set	F_NOBS_PCT	11 (91.7)	11	12	0.91667				4 (100.0)	4	4

Protocol No.: 0000
ICON Study No.: 0001

Page 1 of 1

Table 14.X.X Disposition - Safety Population

	A N = 12 n(%)	B N = 4 n(%)
Safety Set	12	4
Lead-in Safety Set	11 (91.7)	4 (100.0)

The columns highlighted in blue will be displayed as-is in the output. The 1st blue column, called *line*, is the text from the left of the report. The remaining blue columns g1_c, g2_c,… are a character version of the treatment groups values created with the respective numeric version g1_n1, g1_n2,… , g2_n1, g2_n2,… and the associated format given in the column *format*. The number of numeric columns and their contents depends on the DIAMOND macro-functions used.
The format of the final treatment column values is defined at a study-level, in the background of the report program, to ensure consistency throughout the study. Each macro-function will have their associated default format. However, it's possible to create a user-defined format with the macro-function `%set_format`, and pass it as a parameter to the DIAMOND macro-functions.

In the core dataset *table*, there will also be some additional information columns in this order:

page_nb	page number which groups lines together for page breaks
line_nb	line number
line_info	name or indication of the DIAMOND macro used
format	format applied to the character version (gx_c with x=1,2,…) of the treatment group values created with a combination of the respective numeric version (gx_n1, gx_n2,… with x=1,2,…).

Some macro-functions create also:

line2	Similar to the column *line*, additional column *line2* displayed at the left of the report (see below for an example)
section	name of a section (parameter, visits,…) which groups lines together
section_nb	numeric version of the column *section*

See the example below created with 5 treatment columns:

```
%let group=treat;
%let group=C1@C2@C3@C4@C5;
```

	page_nb	section_nb	section	line_nb	line_info	line	line2	format	g1_c	g1_n1	g1_n2	g1_n3	g1
30	2	9	RR Interval, Aggregate (msec)	30	N	Day 4 (4 Hours Postdose)	n	F_STATS01EG_N	16	16	.	.	.
31	2	9	RR Interval, Aggregate (msec)	31	MEAN STD		Mean (SD)	F_STATS01EG_MEAN_STD	949.5 (112.7)	949.5	112.7	.	.
32	2	9	RR Interval, Aggregate (msec)	32	STDERR		SE	F_STATS01EG_STDERR	28.2	28.1836	.	.	.
33	2	9	RR Interval, Aggregate (msec)	33	MEDIAN		Median	F_STATS01EG_MEDIAN	943.0	943	.	.	.
34	2	9	RR Interval, Aggregate (msec)	34	MIN MAX		Min, Max	F_STATS01EG_MIN_MAX	734, 1232	734	1232	.	.
35	2	9	RR Interval, Aggregate (msec)	35	EMPTY					.	.	.	.
36	3	9	RR Interval, Aggregate (msec)	36	N	Day 4 (8 Hours Postdose)	n	F_STATS01EG_N	16	16	.	.	.
37	3	9	RR Interval, Aggregate (msec)	37	MEAN STD		Mean (SD)	F_STATS01EG_MEAN_STD	-42.8 (110.8)	-42.75	110.8	.	.
38	3	9	RR Interval, Aggregate (msec)	38	STDERR		SE	F_STATS01EG_STDERR	27.7	27.7097	.	.	.
39	3	9	RR Interval, Aggregate (msec)	39	MEDIAN		Median	F_STATS01EG_MEDIAN	-35.0	-35	.	.	.
40	3	9	RR Interval, Aggregate (msec)	40	MIN MAX		Min, Max	F_STATS01EG_MIN_MAX	-220, 140	-220	140	.	.

(… right end of the dataset *table*)

	g4_n5	g4_n6	g5_c	g5_n1	g5_n2	g5_n3	g5_n4	g5_n5	g5_n6	gt_c	gt_n1	gt_n2	gt_n3	gt_n4	gt_n5	gt_n6
30			16	16						48	48					
31			970.8 (134.6)	970.75	134.6281					963.7 (125.4)	963.66667	125.41				
32			33.7	33.657032						18.1	18.101304					
33			944.0	944						944.0	944					
34			830, 1326	830	1326					734, 1326	734	1326				
35																
36			16	16						48	48					
37			-44.1 (152.9)	-44.125	152.9313					-43.7 (137.3)	-43.66667	137.294				
38			38.2	38.232826						19.8	19.816743					
39			-39.0	-39						-39.0	-39					
40			-282, 288	-282	288					-282, 288	-282	288				

Like other set of columns, the columns g5_c and g5_n1, g5_n2,… have been created with the condition `treat="C5"` (=> *&group*="5th value of *&group_list"*).
At the end, a format is applied to all numeric columns (g5_n1, g5_n2,…) to create the character column g5_c.
The columns gt_c and gt_n1, gt_n2,… (total) have been created with the combination of all treatment columns, meaning the condition `treat in ("C1","C2","C3","C4","C5").`

DIAMOND MACRO-FUNCTIONS

Each DIAMOND macro-function is designed to perform a specific function and can be used individually or logically grouped together with other DIAMOND macro-functions.

The DIAMOND macro-functions are divided in 2 categories: standard and study specific macro-functions.
The standard macro-functions are given in compiled version and the study specific macro-functions are given as the source code so that they can be updated in a flexible way.

LIST OF CURRENT DIAMOND MACRO-FUNCTIONS

Standard macros			
Environment settings	***Creation of core dataset***	***Functions used by other macro-functions***	***Creation of output***
%reset_global_macros %read_isotop %count_denoms	%new_table %insert_nobs %insert_freq %insert_stats %insert_stats_finish %insert_dimensions %insert_mcibin01 %insert_mcidbsi1 %insert_text %insert_empty %stat_ mcibin01 %stat_ mcidbsi1 %end_page %end_section	%fmt_apply %fmt_repeat %next_line %break_line %indent_ods_rtf	%create_table_output %output_file_start %output_file_end %make_rtf

Study specific macros – updated per study request			
Environment settings	***Creation of core dataset***	***Functions used by other macro-functions***	***Creation of output***
%titles %footnotes	%define_right_group	%set_formats	%ods_rtfstyle %fix_labels

All names are quite intuitive. See appendix for details

FREQUENTLY USED DIAMOND MACRO-FUNCTIONS

To make it as simple as possible, we will only describe the relevant features of each frequently used DIAMOND macro-functions.

`%new_table`
Create an empty dataset *table* with every standard variables needed in the analysis.

`%count_denoms(DS=dataset,WHR=condition);`
Counts the denominators (or population) according to the macro-variables *&group/&group_list* and stores the values in the respective macro-variables *&denom_1, &denom_2,… &denom_t* (See paragraph "Environment setting" – "Report-level" for details).
The values can be displayed in the header of the outputs (i.e. [N=&denom_1]) or used in some DIAMOND macro-functions to derive percentages. In this case, the advantage is that the denominators are counted independently from the current dataset.

`%insert_nobs(DS=dataset,WHR=condition,TX='text to display',PCT=1);`
Insert in the core dataset *table*, with the corresponding text in the column *line*, one line with the number of observations for each treatment group defined with *&group/&group_list*. With the option &PCT=0/1, percentages can be displayed or not.

line_nb	line_info	line	format	g1_c	g1_n1	g1_n2	g1_n3	g1_n4	g1_n5	g1_n6	g2_c	g2_n1	g2_n2	g2_n3
1	NOBS	text to display	F_NOBS_PCT	4 (28.6)	4	14	0.285714				4 (80.0)	4	5	0.8

With the DIAMOND macro-function `%insert_nobs`, each set of columns *g2_*, g3_*,…gt_** are defined similar to the following columns:
g1_n1=number of events with the condition
g1_n2=number of events in the population (*&denom_1*)
g1_n3=ratio *g1_n1*/*g1_n2*

```
%insert_freq(DS=dataset,WHR=condition,
VAR=variable,
VALUES='modality1'='text for modality1'
      @'modality2'='text for modality2'
      @'modality3'='text for modality3',
INDENT=2);
```

Insert in the core dataset *table*, several lines with the number of observations for each treatment column and each modalities of the variable specified with the *&var* macro-variable.
The list of modalities given with *&values* is optional but ables to sort the modalities and assign a specific text to each modality. An indentation of the modalities is also possible with the option *&indent*.
As the DIAMOND macro-function `%insert_freq` is built with `%insert_nobs`, the columns *line_info* and *format* will be similar.

line_nb	line_info	line	format	g1_c	g1_n1	g1_n2	g1_n3	g1_n4	g1_n5	g1_n6	g2_c	g2_n1	g2_n2
1	NOBS	>18 - <=25	F_NOBS_PCT	3 (21.4)	3	14	0.2142857				1 (20.0)	1	5
2	NOBS	>26 - <=35	F_NOBS_PCT	0	0	14	0				1 (20.0)	1	5
3	NOBS	>36 - <=45	F_NOBS_PCT	5 (35.7)	5	14	0.3571429				1 (20.0)	1	5
4	NOBS	>45	F_NOBS_PCT	6 (42.9)	6	14	0.4285714				2 (40.0)	2	5

With the DIAMOND macro-function `%insert_freq`, each set of columns *g2_*, g3_*,…gt_** are defined similar to the following columns:
g1_n1=number of events with the condition
g1_n2=number of events in the population (*&denom_1*)
g1_n3=ratio *g1_n1*/*g1_n2*

```
%insert_stats(DS=dataset,WHR=condition,
VAR=variable,
STATS=N=n'
     @MEAN STD='Mean (SD)'
     @MEDIAN='Median'
     @MIN MAX='Min, Max',
INDENT=2);
```

Insert in the core dataset *table* several lines with statistics for the variable &var for each treatment column and each statistical keyword. The keywords are those from a Proc MEANS statement with some additional ones for Geometric means or other statistics. Several statistics can be displayed on one line, and a new line will be separate with an @ symbol.
An indentation of the modalities is also possible with the option &indent.

line_nb	line_info	line	format	g1_c	g1_n1	g1_n2	g1_n3	g1_n4	g1_n5	g1_n6	g2_c	g2_n1	g2_n2
1	N	n	F_STATS_N	14	14						5	5	
2	MEAN STD	Mean (SD)	F_STATS_MEAN_STD	40.6 (11.68)	40.64286	11.6791					37.8 (14.27)	37.8	14.272351
3	MEDIAN	Median	F_STATS_MEDIAN	41.0	41						36.0	36	
4	MIN MAX	Min, Max	F_STATS_MIN_MAX	21, 57	21	57					23, 57	23	57

With the DIAMOND macro-function `%insert_stats`, each set of columns *g2_*, g3_*,…gt_** are defined similar to the following columns:
g1_n1=1st statistic from column *line_info*
g1_n2=2nd statistic from column *line_info*

```
%insert_dimensions(DS=dataset,WHR=condition,
LEVEL1=variable_SOC,
LEVEL1_sort=ALPHA,
LEVEL2=variable_PT,
LEVEL2_sort=EVENTS,
LINE_WIDTH=50,
REPORT_LINE=
       L0_0='Subjects with any TEAEs'
       L1_0=skip#level1
       L2_0=indent2#level2);
```

Insert in the core dataset *table* several lines with a combination of the variables *&level1* and *&level2* for each treatment column. The lines are combined according to the statements given in *&report_line* with, in this example, a skip line before each level1 and an indentation before each level2. The sorting order for each level can be: ALPHA (alphabetical order), SUBJECTS (descending number of subjects in total column) or EVENTS (descending number of events in total column).
This DIAMOND macro-function is typically used for adverse events or medical history tables.

line_nb	line_info	line	format	g1_c	g1_n1	g1_n2	g1_n3	g1_n4	g1_n5	g1_n6	g2_c	g2_n1
1	L0_0	Subjects with Any TEAE	F_DIMENSIONS_L0_0	12 (85.7)	12	14	0.857143	41	41	1	4 (80.0)	4
2												
3	L1_0	Blood and lymphatic system disorders	F_DIMENSIONS_L1_0	3 (21.4)	3	14	0.214286	3	41	0.07317	0	0
4	L2_0	Eosinophilia	F_DIMENSIONS_L2_0	1 (7.1)	1	14	0.071429	1	41	0.02439	0	0
5	L2_0	Leukocytosis	F_DIMENSIONS_L2_0	1 (7.1)	1	14	0.071429	1	41	0.02439	0	0
6	L2_0	Lymphadenopathy	F_DIMENSIONS_L2_0	1 (7.1)	1	14	0.071429	1	41	0.02439	0	0
7												
8	L1_0	Cardiac disorders	F_DIMENSIONS_L1_0	1 (7.1)	1	14	0.071429	1	41	0.02439	0	0
9	L2_0	Angina pectoris	F_DIMENSIONS_L2_0	1 (7.1)	1	14	0.071429	1	41	0.02439	0	0
10												

With the DIAMOND macro-function `%insert_dimensions`, each set of columns *g2_*, g3_*,…gt_** are defined similar to the following columns:
g1_n1=number of subjects with the condition
g1_n2=number of subjects in the population (*&denom_1*)
g1_n3=ratio *g1_n1*/*g1_n2*
g1_n4=number of events with the condition
g1_n5=number of events in total
g1_n6=ratio *g1_n4*/*g1_n5*

OUTPUT FILE

As the core dataset *table* has a unique structure and already the desired formatting, it's now easy to create the output with a quite standard and simple macro call:

```
%create_table_output(
   ,line_width            = 60
   ,group_width           = 20
   ,line_header_txt       = 'Baseline~Characteristic'
   ,group_header2_txt     = 'n (%)'
   ,group_denom_in_header = 1
 );
```

Which is equivalent (but not only) to:

```
%output_file_start;
proc report data = table nowd headline headskip split = "~" missing spacing = 1;
  column line_nb line g1_c g2_c g3_c gt_c;

  define line_nb / order order=internal noprint;
  define line    / display left flow width = 60 "Baseline~Characteristic";
  define g1_c    / display left flow width = 20 "Study Group A~[N=&denom_1.]~n (%)";
  define g2_c    / display left flow width = 20 "Study Group B~[N=&denom_2.]~n (%)";
  define g3_c    / display left flow width = 20 "Study Group C~[N=&denom_3.]~n (%)";
  define gt_c    / display left flow width = 20 "Total~[N=&denom_t.]~n (%)";

  %titles;
  %footnotes;
run;
%output_file_end;
```

Listings are typically created with the 2nd version (i.e., Proc REPORT statement).

TYPE OF OUTPUTS

Only one output can be generated at the same time. As there are several types of outputs, only one DIAMOND macro will generate all the type of outputs, giving an easy way to switch from one type of output to the other.
DIAMOND can produce 4 types of outputs:

- Standard table/listing: an .rtf file produced with a .txt file [output.txt + output.rtf]
  ```
  %let output_mode=TXT_RTF;
  ```
- LST file: file produced with SAS® output window [output.lst]
  ```
  %let output_mode=LST;
  ```
- ODS RTF file: file produced with SAS® ODS RTF [output.rtf]
  ```
  %let output_mode=ODS_RTF;
  ```
- In-text table: file produced with SAS® ODS RTF with minimal features (no pagination, no page breaks, no empty lines,…) which able to copy/paste the output with one click in another document [output.rtf, prefixed with 'it']
  ```
  %let output_mode=INTEXT;
  ```

In addition to these outputs, DIAMOND gives the possibility to store the SAS dataset used for the output. Actually, the core dataset *table,* with fewer variables, will be stored in a specific folder, so that it can be sent to the client without any specific features from DIAMOND.

```
%let output_dataset=1 (or 0);
```

ACKNOWLEDGMENTS

Thanks to the DIAMOND development team for their support, especially Valerie Williams, Janardhan Reddy Mandadi, and Ganesh Prasad; and thanks to all ICON statistical programmers for their valuable input.

CONCLUSION

The DIAMOND macro-functions are user-friendly, reusable, and efficient by reducing programming effort with minimal set up. They are also quite flexible, as they can be logically grouped together within a user-written calling program, in order to produce a variety of reports and meet study reporting requirements. Users may also substitute the presentation layer of DIAMOND with custom code such as Proc REPORT, in order to further customize their outputs. In addition, the user-written calling programs can be easily modified, at the study-level, for reuse within other studies. The DIAMOND macro-functions are standard code which consistently produce a high level of quality outputs and support a full independent validation of the calling programs, per a standard QC process.

CONTACT INFORMATION

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

Nathalie Fuchs
ICON Clinical Research Ltd
5, rue du talus
67400 Illkirch - France
Work Phone: +33.36.707.7132
Email: nathalie.fuchs@iconplc.com
Web: https://www.iconplc.com

APPENDIX

APPENDIX 1: LIST OF DIAMOND MACRO-FUNCTIONS

The DIAMOND macro-functions are divided in 2 categories: standard and study specific macro-functions.
The standard macro-functions are given in compiled version and the study specific macro-functions are given with the code so that they can be updated in a flexible way.

STANDARD MACRO-FUNCTIONS

(In alphabetical order)

Name	Description	Aim
%break_line	Split a text variable (i.e. variable line from core dataset table) at a given number of characters	
%count_denoms	Count denominators according to macro-variables &group/&group_list and store them in macro-variables &denom_1, &denom_2,… &denom_t	
%create_table_output	Create the output with the core dataset table using Proc REPORT and other DIAMOND macro-functions. The type of output will depend on the &output_mode (TXT_RTF, LST, ODS_RTF or INTEXT)	Output
%end_page	Add a page break in the core dataset table by incrementing the variable page_nb from the core dataset table	Insert in dataset *table*
%end_section	Add a section break in the core dataset table by incrementing the variable section_nb from core dataset table	Insert in dataset *table*
%fmt_apply	Used by all %insert_* DIAMOND macro-functions to create the variables g1_c, g2_c,… in core dataset table with the associated numeric version g1_n1, g1_n2,… g2_n1,g2_n2,…	
%fmt_repeat	Used by %set_format to create new format	
%indent_ods_rtf	Applies indenting for ODS RTF and In-text tables (does not work by adding spaces).	Output
%insert_dimensions	Insert frequency counts through combination of several variables (i.e. SOC/PT)	Insert in dataset *table*
%insert_empty	Insert a blank line in the core dataset *table*	Insert in dataset *table*
%insert_freq	Insert frequencies from all modalities of a variable in the core dataset *table*	Insert in dataset *table*
%insert_mcibin01	Insert CI for binomial parameter in the core dataset *table*	Insert in dataset *table*
%insert_mcidbsi1	Insert CI for a difference of 2 binomial parameters in the core dataset *table*	Insert in dataset *table*
%insert_nobs	Insert frequencies from 1 value of a variable in the core dataset *table*	Insert in dataset *table*
%insert_stats	Insert values from quantitative analysis (N, mean, SD,…) in the core dataset *table*	Insert in dataset *table*
%insert_stats_finish	Insert numeric values in the core dataset *table* from a dataset not build with the DIAMOND macro-functions	Insert in dataset *table*
%insert_text	Insert a line with text in the core dataset *table*	Insert in dataset *table*
%make_rtf	Used by DIAMOND macro-function %create_table_output to create the output in rtf (convert .txt file to .rtf file)	Output
%new_table	Create an empty dataset named "*table*" with the required columns for the report	Insert in dataset *table*
%next_line	Applies indenting in header of outputs for ODS RTF and In-text tables (does not work by adding spaces).	Output
%output_file_start	Starts output display	Output
%output_file_end	Ends output display	Output
%read_isotop	Read ISOTOP.xml file to create titles and footnotes associated macro-variables (&tit1, &tit2,… &foot1,&foot2,…). These macro-variables will then be used by the DIAMOND macro-functions %titles/%footnotes to create the titles and footnotes.	Output

%reset_global_macros	Initialize global macro-variables for the study at the beginning of a session and reset all macro-variables to their values by default when added without parameters at the beginning of a program.	
%stat_mcibin01	Calculate CI for binomial parameter in a data step	
%stat_mcidbsi1	Calculate CI for a difference of 2 binomial parameter in a data step	

STUDY SPECIFIC MACRO-FUNCTIONS

These macro-functions will have to be updated, if needed, for each study.

Name	Description
%titles	Use macro-variables &tit1, &tit2,… to display titles with a study specific format (page numbering, date,…)
%footnotes	Use macro-variables &foot1, &foot2,… to display footnotes with a study specific format (page numbering, date,…)
%define_right_group	Define new groups, if needed, for cross-over studies for instance.
%fix_labels	Update *&group_list* values to create labels of header of output
%set_formats	Create format to be used for display in the core dataset *table*
%odsrtf_style	When creating ODS RTF Tables (&output_mode=ODS_RTF), or In-text Tables (&output_mode=INTEXT), create the style *odsrtf* with a Proc TEMPLATE.

APPENDIX 2: SET-UP FILE

```
*** Macro variables ***;
%global protocol icon_study path;
%let protocol      = 0000;
%let icon_study    = 0001;
%let path          = O:\remote\sas_icr_bio_eu\project\0000\dummy\0001\icr_bio;

%global sponsor_study meddra whodde;
%let sponsor_study = Dummy D0000-0001;
%let meddra        = 18.1;
%let whodde        = March 2016, Format B2;


*** Libnames ***;
libname libanal "&path.\dev\datasets\analysis";
libname libraw  "&path.\dev\datasets\raw" access=readonly;
libname diamond "&path.\dev\programs\Utils\diamond" access=readonly;

libname vlibanal "&path.\val\datasets\analysis";


*** Macro functions ***;
options mautosource
        sasautos = ("&path\dev\programs\Utils\diamond\Studyspecific",
                    sasautos);
options mstored
        sasmstore = diamond;


*** Formats ***;
options fmtsearch = (work libraw libanal);


*** Options ***;
options yearcutoff = 1920 missing = "." nodate nonumber mcompilenote = all
```

```
        nosymbolgen nomlogic nomprint nosource nosource2 notes;


*** Codes added for Titles and footnotes using isotop ***;
filename tflspecs "&path.\dev\programs\utils\ISOTOP.xml";
filename xmlmap "&path.\dev\programs\utils\diamond\ISOTOP.map";
libname tflspecs xml xmlmap=xmlmap;


*** Global Macro variables used in Statistical environment ***;
%global output_dataset output_mode;
%let output_mode    = TXT_RTF;
%let output_dataset = 1;

%reset_global_macros(
  def_group           = TRTA,
  def_group_list      = A@B@C,
  def_totals          = 0,
  def_subjno          = USUBJID,

  def_num_col_count   = 6,
  def_page_format     = LETTER,
  def_orient          = LS,
  def_font            = 0,
  def_fontsize        = 8,
  def_LS_ls           = 134,
  def_LS_ps           = 51,
  def_LS_marginT      = 1in,
  def_LS_marginB      = 1in,
  def_LS_marginL      = 1in,
  def_LS_marginR      = 1in,

  def_tit_ver         = ICON,
  def_foot_ver        = ICON,
  def_bookmark_char   = §,
  def_bookmark_ignore = 1,
  def_output_path     = &path\dev\programs,
  def_layout_center   = nocenter,
  def_automatic_log   = 1,
  def_automatic_log_check   = 1,
  def_output_dataset_folder = 1
);
```