PhUSE 2013

Paper PP04

Descriptive Summary Table Made Easy

Murshed Siddick, Shafi Consultancy, London, UK
Rafi Rahi, Shafi Consultancy, London, UK

ABSTRACT

Descriptive summary tables are an everyday part of the clinical trial reporting world, whether it's calculating N, Mean, Std, or N and Percent etc., these are the bread and butter of reporting. However, the amount of time spent to produce these is still extraordinary. The solution to this involves creating a set of small and very easy to understand macros, which allows the user to develop and produce descriptive summary tables within minutes. These can be used to produce or validate most safety tables without any problems. The idea is that the user calls the macros for the various parts of the table and then puts them together. This allows users to create many types of tables with the minimal effort. It also means that when Statisticians would like to change the table afterwards, it can be done with the minimum of effort.

INTRODUCTION

This paper describes the structure of a SAS program that creates a descriptive statistics summary table (e.g. N and percentage of categorical data or N, Mean, Median, Standard Deviation of continuous data) using the SAS macro programming language. Producing the descriptive statistics summary table is probably the most commonly repeated feature of any clinical trial reporting process. By using a SAS macro code to generate these tables quickly and effectively we can take the whole data analysis process to a new level of efficient coding.

In a typical study, multiple members of a study team are usually involved in writing programs to produce tables and listings from the analysis datasets. Each member either produces their own code or adjusts their code from a previous study. Once the codes are written, they are then checked and validated by resolving the issues found before producing the final tables. This process, however, has a number of drawbacks:

1. Team members have to spend a lot of time to write the codes, then test and validate them before producing the final tables.
2. As lots of programmers reproduce similar programs in different ways; they are time consuming to **validate and maintain over time**, e.g.
    - It is often frustrating for a programmer to look at someone else's code and resolve a coding issue. However, this may be required if an issue is exposed at a late stage, and the original programmer is absent or busy on another study.
    - If a future update affects a study such that the codes need to be reproduced to reflect those updates then the study team leader has to bring together his full team again, which at times can be quite challenging.

These drawbacks can be overcome by using a simple macro system for producing a variety of descriptive tables so that it will be easy to validate and maintain the programs over time. Instead of creating their individual code, team members now get access to a set of validated macros and perform a set of simple actions:

1. Calling the macros in the right order
2. Passing all necessary parameters to each macro
3. Ensuring all the intermediate datasets produced are meaningful at each stage
4. Ensuring the final table is produced in the prescribed layout.

This reduces the chance of making unwanted mistakes as the code is highly streamlined consisting of a few simple macro calls to produce a table. It also gives added flexibility and control to the study team leader. For example, the team leader can instruct a particular team member to look into certain issues in a table rather than needing to go back to each individual member. Similarly if there is a need for tables to be updated at a later stage, the team leader can instruct a specific member to carry out all these changes rather than needing to pool all of his/her team together.

MACRO CODE FOR TABLE

CODE OVERVIEW

The production of a descriptive summary table is divided into five key stages as in Diagram 1. Each of these stages is carried out by calling a designated macro in order to produce the final table. A typical code will look like that in Diagram 2. The macro code collects baseline values of three variables 'age', 'race' and 'sex' and other relevant information from a number of analysis datasets and collates them into a single dataset. The final output will then contain the descriptive summary for each of these demography variables (as shown in Diagram 3).

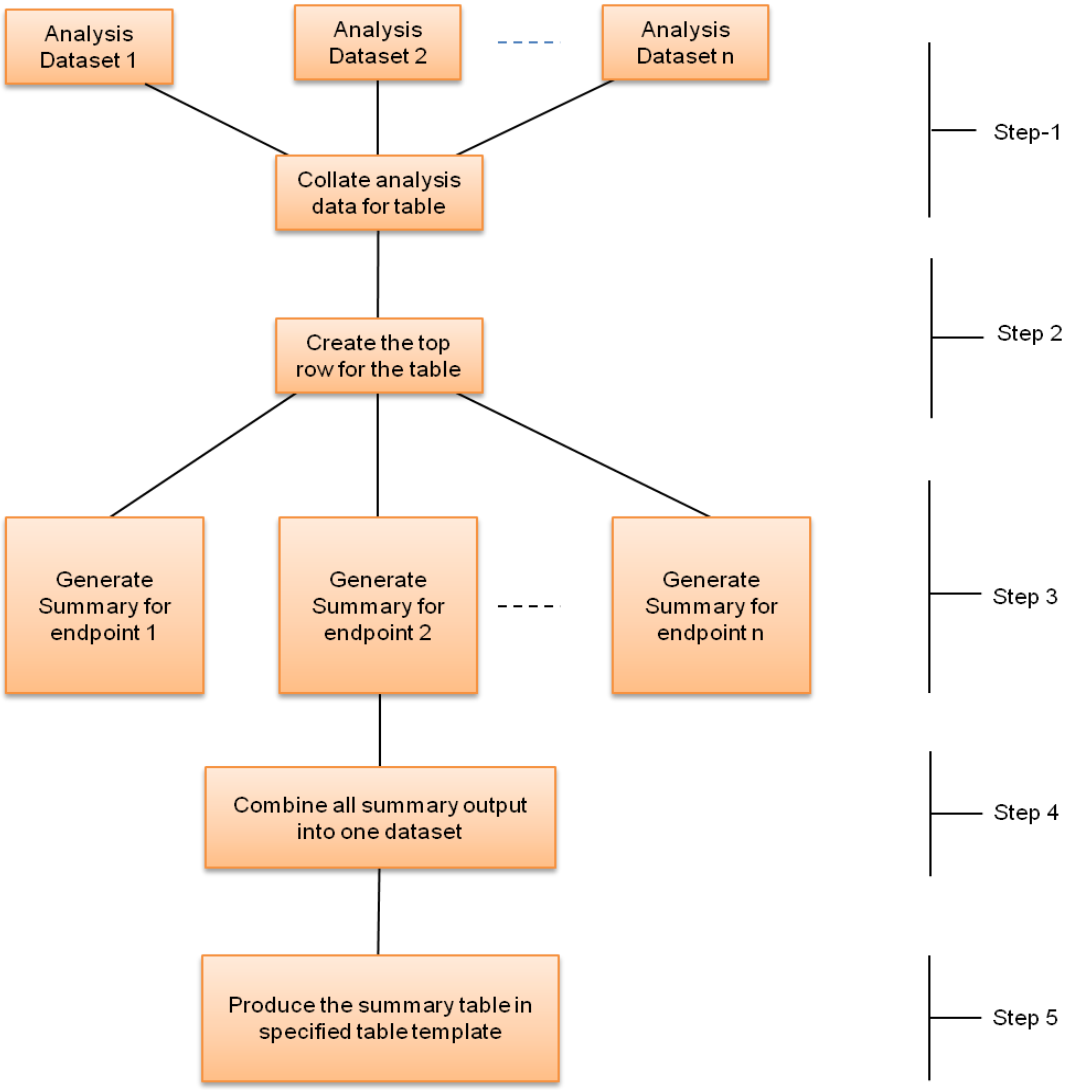


Diagram 1: Different Stages in Producing a Descriptive Summary Table

The datasets produced during the code execution for a table are identified with the single prefix name. It is goodto give full or part of the table name as the prefix name so that the datasets for a particular table can be easily located and distinguished by the table name. For example, we used 'demog_' as a prefix for creating the demography table. The table program then creates a series of datasets (intermediate and final datasets) all starting with the 'demog_' prefix.

```
TITLE "The Demographic Data Summary";

%LET prefix=demog_;

%get_data(inpopu  = popu,    popu    = TREATED,                    ---- Step -1
          intrt   = trt,     inbase  = base,
          outdata = &prefix.alldata);

%count_summary(indata     = &prefix.alldata,                       ---- Step -2
               v_sort     = 0,
               v_label    = "Total treated",
               outdata    = &prefix.x_toprow,
               outbign    = &prefix.bign);

%stat_summary(indata      = &prefix.alldata,                       ---- Step -3
              v_val       = age,
              v_label     = "Age[years]",
              v_sort      = 1,
              outdata     = &prefix.x_age);

%count_summary(indata     = &prefix.alldata,
               inbign     = &prefix.bign,
               v_val      = sex,
               v_label    = "Gender",
               v_valdc    = sexdc,
               v_sort     = 2,
               outdata    = &prefix.x_sex);

%count_summary(indata     = &prefix.alldata,
               inbign     = &prefix.bign,
               v_val      = race,
               v_label    = "Race",
               v_valdc    = racedc,
               v_sort     = 3,
               outdata    = &prefix.x_race);

%combine(indata  = &prefix.x_,                                     ---- Step -4
         outdata = &prefix.table);

%report1(indata    = &prefix.table,                                ---- Step -5
         pgbrk_opt = 2,
         brk_after = 1 2
         outdata   = &prefix.2report);

TITLE;
```

Diagram 2: An Example Code – Creating Demography Table from Baseline Data

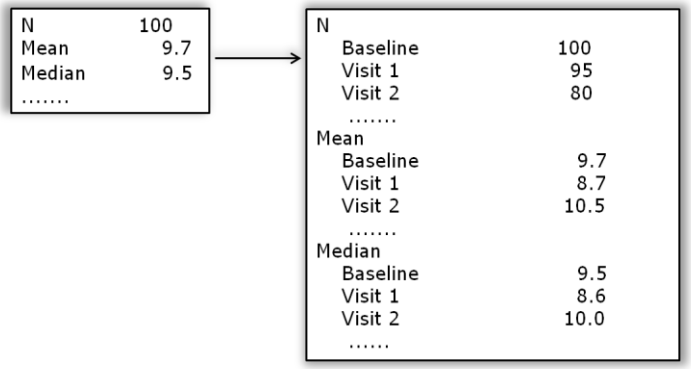
TABLE OUTPUT:

A typical output for a table program is shown below. Although the layout of the table is dictated by the display template, most of the tables follow a simple layout where the columns represent treatment groups and rows represent summary values (e.g. N and percentage for male, female or N, Mean, Median etc for Age etc). Summaries from a single variable (e.g. age or sex) are printed together and labelled with an appropriate variable name (e.g. “Age [Year]” for ‘age’, “Gender” for ‘sex’).

	TREATMENT GROUP						
	Red	Blue	Green	Brown	Black	White	Gray
Total treated	135 (100.0)	135 (100.0)	133 (100.0)	133 (100.0)	133 (100.0)	132 (100.0)	131 (100.0)
Age [years]							
N	135	135	133	133	133	132	131
Mean	57.9	57.0	56.8	56.1	58.5	56.8	56.1
SD	10.5	11.8	10.9	10.4	10.5	9.4	10.2
Min	34.0	24.0	22.0	27.0	29.0	34.0	27.0
Median	58.0	58.0	56.0	56.0	60.0	57.0	56.0
Max	78.0	78.0	80.0	80.0	79.0	77.0	80.0
Gender							
Male	77 (57.0)	70 (51.9)	65 (48.9)	83 (62.4)	70 (52.6)	73 (55.3)	73 (55.7)
Female	58 (43.0)	65 (48.1)	68 (51.1)	50 (37.6)	63 (47.4)	59 (44.7)	58 (44.3)
Race							
Amer.Ind./Alaska Nat	0 (0.0)	2 (1.5)	1 (0.8)	2 (1.5)	1 (0.8)	0 (0.0)	0 (0.0)
Asian	1 (0.7)	3 (2.2)	3 (2.3)	4 (3.0)	2 (1.5)	1 (0.8)	2 (1.5)
Black/African Amer.	14 (10.4)	18 (13.3)	16 (12.0)	21 (15.8)	14 (10.5)	15 (11.4)	17 (13.0)
Hawaiian/Pacif. Isle	0 (0.0)	1 (0.7)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)	0 (0.0)
White	120 (88.9)	111 (82.2)	113 (85.0)	106 (79.7)	116 (87.2)	116 (87.9)	112 (85.5)

Diagram 3: A Demography Table from Baseline Data (for a dummy study)

We often summarise variables over a treatment period i.e. variables recorded at multiple time points. These are usually the study endpoints data. In such case, the code allows the option to summarise variables' values over multiple time points i.e. classify each summary category (N, Mean etc.) into multiple time points.



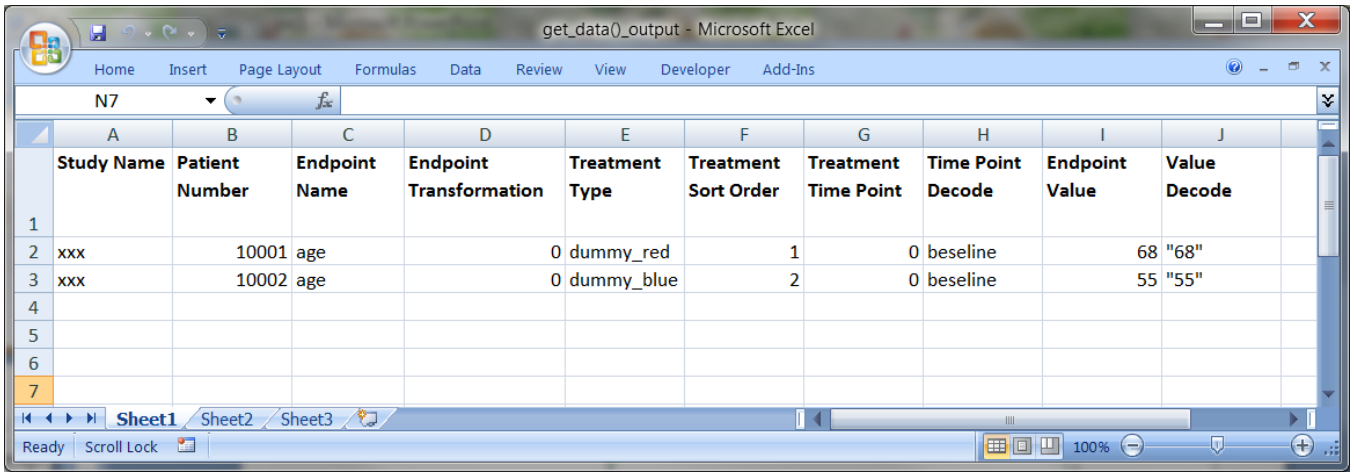
COMPONENTS OF A TABLE PROGRAM

There are five macros to carry out the different steps of table production:

1. get_data(),
2. count_summary(),
3. stat_summary(),
4. combine() and
5. report1().

GET_DATA()

Get_data() collates all the endpoint data, which we are interested in, in a single dataset and arrange them in a specific format so that data is ready to be read by the rest of the macro code. It is important to choose and decide a suitable format and use that format consistently by all members in a study. In our case we have chosen a longitudinal data format simply because it is easy for collating the endpoints from our analysis datasets which stores endpoint data in longitudinal format.



	A	B	C	D	E	F	G	H	I	J
1	Study Name	Patient Number	Endpoint Name	Endpoint Transformation	Treatment Type	Treatment Sort Order	Treatment Time Point	Time Point Decode	Endpoint Value	Value Decode
2	xxx	10001	age	0	dummy_red	1	0	beseline	68	"68"
3	xxx	10002	age	0	dummy_blue	2	0	beseline	55	"55"
4										
5										
6										
7										

Diagram 4: Data Structure for the Macro Input File

The advantage of choosing this format is that it is quite easy to organise the analysed data regardless of how they appear in the analysis datasets. If a study is creating the analysis dataset quite differently from our desired format, the only thing that may need to be adapted in the whole code is the get_data() macro. Once the get_data() macro is adapted for the analysis datsets, the rest of the code is universal for any types of clinical trial reporting.

COUNT_SUMMARY() AND STAT_SUMMARY()

Count_summary() calculates the N and percentage for categorical data e.g. race, smoking category etc, whereas stat_summary() macro calculates the descriptive summary statistics for continuous data, e.g. age, weight etc. Count_summary() and stat_summary() macros are called in step 3 (see Diagram 1, 2) as many times as required to calculate the summaries of all the desired variables. These are the two key macros in our table production process.

For each of these macro call the programmer has to provide the variable name (v_val) to be analysed, the time point classifier (v_classifier) to be read, a suitable label (v_label) to be displayed as a header and a sort number (v_sort) for ordering the different variables.

TOP ROW AND BIG N

In most cases, a descriptive summary table requires a top row containing the **total count** i.e. total number of patients in each treatment group. This is basically the output of the count_summary() macro with no variable name and no time points specified (Diagram 2). When variable names and time classification parameters are not provided count_summary() macro simply counts the total population size in each treatment group. This has to be done in step 2 before any summary calculations (step 3) take place as these total counts, often referred as “big N”, are used for the percentage calculations in step 3.

COMBINE()

This combines all the different summary outputs (produced in step 2 and 3) into one final dataset. The final dataset stores all the content of the summary table.

REPORTN()

Reportn() macro produces the descriptive summary table in the specified layout and file format provided where n stands for the matching table templates. If the specified table template is different from the standard template assumed in Diagram 3, we can create new reportn() macros to match the different display template. As a result, for different table template specifications we will have different report macros, e.g. report1(), report2() etc. and then the user can simply call the relevant reportn() macro to produce the tables in his/her specific template.

LIMITATIONS AND BENEFITS

The code developed in our process assumes that both the input file format and the output table template are quite set (Diagram 3 and 4). However, these limitations can also be seen as strength from the management point of view**.** If in a study the analysis datasets are arranged in quite a different data structure (than that in Diagram 4) and the summary tables are required in a somewhat different template, then only the structure of the get_data() and report1() macro need to be updated and all the programs calling these macros will not have to be modified. Given the macro is developed with good programming practice, and the macros are small and perform a very basic task, the changes should not be very time consuming. Due to structuring the table creation in clearly identified sections and carrying out each section using an independent macro, a structured workflow can be suggested for an effective table production life cycle.

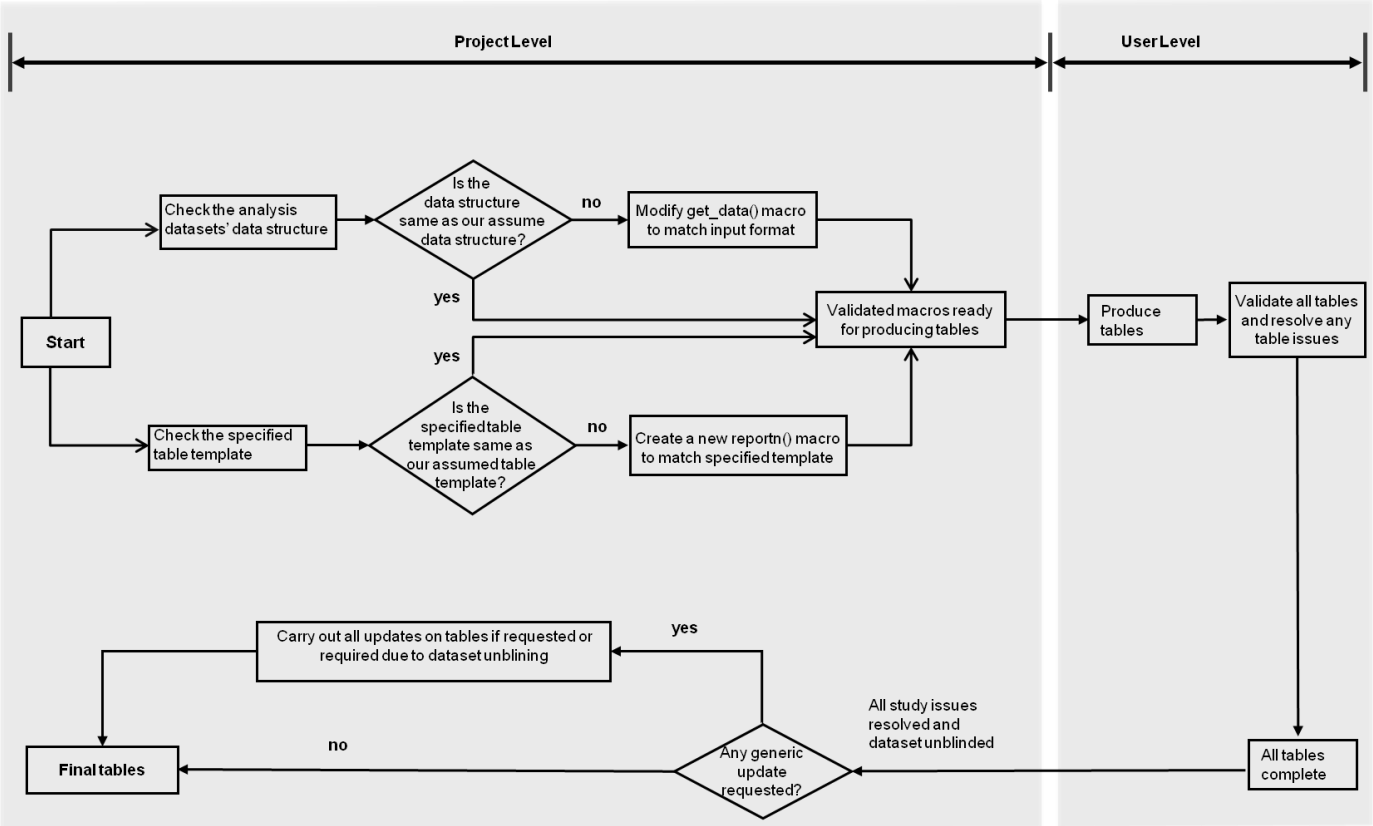


Diagram 5: Effective Table Production Life Cycle

CONCLUSION

In this paper, we explained five macros and a scheme for producing descriptive summary tables using these macros. By identifying the key stages and developing macros for each of these stages we have streamlined the table production process and resolved the issues associated with them being programmed independently. Programmers can create tables for many endpoints as and when they need without needing to allocate too much extra time for them. All they need to ensure is that data is collated in a simple specific structure to be able to be read by the rest of the code properly. More importantly, the scheme allows much more centralised control over the workflow and makes it easy to maintain the code and update the tables centrally whenever necessary.

ACKNOWLEDGMENTS

The authors wish to thank Shafi Chowdhury for supervising the project, reviewing the results and giving valuable feedback to achieve the project goal.

CONTACT INFORMATION

Please feel free to contact us for further information or with any questions or comments you may have.

Muhammad Murshed Siddick
Shafi Consultancy Limited.
Regus House, Highbridge,
Oxford Road, Uxbridge,
Middlesex, UB8 1HR
Phone: 01895 876 533
Email: murshed@shaficonsultancy.com
Web: www.ShafiConsultancy.com

Rafi Ahmed Rahi
Shafi Consultancy Limited.
Regus House, Highbridge,
Oxford Road, Uxbridge,
Middlesex, UB8 1HR
Phone: 01895 876 533
Email: rafi@shaficonsultancy.com
Web: www.ShafiConsultancy.com