Paper 1730 - 2014

Proc Tabulate: Extending This Powerful Tool Beyond Its Limitations

Justin Jia, Canadian Imperial Bank of Commerce (CIBC), Toronto, Ontario, Canada

Amanda Lin, Bell Canada, Toronto, Ontario, Canada

ABSTRACT

Proc Tabulate is a powerful tool for creating tabular summary reports. Its advantages, over Proc Report, are that it requires less code, allows for more convenient table construction, and it uses syntax that makes it easier to modify a table's structure. However, its inability to compute the sum, difference, product and ratio of column sums has hindered its use in many circumstances. This paper will illustrate and discuss some creative approaches and methods for overcoming these limitations, which allow users to produce needed reports and still enjoy Proc Tabulate's simplicity and convenience. These methods and skills can have prominent applications in a variety of marketing research, business intelligence and analytics fields.

INTRODUCTION

In the process of data analysis and reporting through SAS® programming, usually one of the final steps is to generate external tabular reports in various formats, such as Excel, PDF, HTML and RTF. For this purpose, we can choose different methods to use depending on the output data structure and the kind of reports that are needed. For example, if we just want to generate detail Excel reports, we can output the SAS data directly to Excel by using Export Wizard, Proc Export or the Libname Excel Engine in SAS9 . These approaches are very simple and easy to use. However, more complex external tabular reports are sometimes needed and can be attained by using the SAS ODS facility and report-writing procedures such as Proc Print, Proc Report and Proc Tabulate. These advanced ways allow us to design and enhance the layout, format, style and attributes of generated reports and, as such, can have important use in real work.

The ODS facility, Proc Print, Proc Report and Proc Tabulate are the most useful SAS procedures for creating tabular reports. Generally speaking, Proc Print is used to create simple detail reports, while Proc Report and Proc Tabulate can produce complex summary reports with a variety of descriptive statistics (count, sum, minimum, maximum, mean, standard deviation, median etc.). Proc Report combines features of the PRINT, MEANS, and TABULATE procedures together with features of the DATA step (Compute Block), thereby making it a very versatile and flexible report-writing tool. It is capable of producing both simple detail reports and enhanced summary reports. As its name suggests, Proc Tabulate is specially designed for tabulation purposes. It is the most powerful and convenient tool in creating tabular summary reports, especially when the tabulation has multiple nesting and complicated structures. In creating summary reports, Proc Tabulate has distinct advantages over Proc Report, because it requires less SAS code, is extremely powerful in table construction, and is also the easiest of the methods for enabling structure modification.

Just as every coin has two sides, each procedure has its own advantages and disadvantages. In tabulation, Proc Tabulate achieves the above advantages at the cost of losing flexibility in comparison to Proc Report. One of its noteworthy drawbacks or limitations is that it cannot perform further calculations on the basis of summarized results (e.g., to compute the sum, difference, product and ratio of column sums). Unfortunately, this is often a necessary requirement in generating tabular summary reports in real work. For example, the difference and ratio of column sums are often needed in creating complex customer metrics and profiles in customer analyses and business analytics. Actually, a lot of data analysts and SAS programmers have encountered this problem and have been annoyed by this limitation in using Proc Tabulate in work. Consequently, some people give up on Proc Tabulate, the most powerful tabulation weapon. Instead, they utilize Proc Report, which might involve more and often quite complex SAS coding. Other people choose to make manual changes in Excel, after ODS creates the result file. Making manual changes in Excel can be tedious, time-consuming and error-prone, and this process becomes more cumbersome when there are a large number of Excel reports involved.

Therefore, here arises the interesting and challenging question addressed in the present paper: are there any methods one can use so that one can utilize Proc Tabulate to create tabular summary reports that involve calculations on column sums?

This paper will illustrate and discuss some creative approaches and methods to rise above the limitations and utilize Proc Tabulate to reach the desired goal.

CASE STUDY A: COMPUTE DIFFERENCE AND RATIO OF COLUMN SUMS ARISING FROM DIFFERENT ANALYSIS VARIABLES.

In this paper, we use three case studies to illustrate and discuss the different approaches and methods to extend Proc Tabulate beyond its limitations. The input data set WORK.Raw is derived from the SASHELP.Pricedata data set using the code shown below. This derived input data set has 1020 observations and 4 variables as shown in Table 1. We will use this data set as our input data in all the 3 case studies.

```
****************************** Create Input Data Set. ************************;
data Raw;
set SASHELP.Pricedata (rename=(Sale=Sales Price3=Refund));
Fiscal_Year= year(date);
keep Fiscal_Year Sales Refund ProductLine;
format Sales Refund dollar12.0;
run;
```

Table 1. Attributes of the Variables in the WORK.RAW Data Set

Variable	Type	Len	Format	Meaning
ProductLine	Char	5		Product Line
Fiscal_Year	Num	8		Fiscal Year
Sales	Num	8	DOLLAR12.	Dollar value of Sales
Refund	Num	8	DOLLAR12.	Dollar value of Refund

As shown in Table 2, the first task is to create a five-year summary report on the basis of each fiscal year and product line. In this report, the Sales and Refund columns represent the sum of the sales and refund values for each Product Line within a given year, the Net Sales column represents the difference between the Sales and Refund amounts, and the Refund Ratio column represents the quotient or ratio (expressed as a percentage) resulting from dividing Refund by Sales.

Table 2. Five Year Sales and Marketing Summary Report of WLMU Company: Case A.

	1998				1999				2000				2001				2002			
	Sales	Refund	Net Sales	Refund Ratio (% of Sales)	Sales	Refund	Net Sales	Refund Ratio (% of Sales)	Sales	Refund	Net Sales	Refund Ratio (% of Sales)	Sales	Refund	Net Sales	Refund Ratio (% of Sales)	Sales	Refund	Net Sales	Refund Ratio (% of Sales)
Product Line	**$**	**$**	**$**	**%**	**$**	**$**	**$**	**%**	**$**	**$**	**$**	**%**	**$**	**$**	**$**	**%**	**$**	**$**	**$**	**%**
Line1	$13,582	$1,187	$12,395	8.7%	$13,910	$1,182	$12,728	8.5%	$13,923	$1,192	$12,731	8.6%	$14,038	$1,192	$12,846	8.5%	$13,725	$1,192	$12,533	8.7%
Line2	$20,645	$1,583	$19,062	7.7%	$21,186	$1,576	$19,610	7.4%	$20,902	$1,590	$19,312	7.6%	$21,036	$1,590	$19,446	7.6%	$20,532	$1,590	$18,942	7.7%
Line3	$18,615	$1,583	$17,032	8.5%	$18,507	$1,576	$16,931	8.5%	$18,196	$1,590	$16,606	8.7%	$18,654	$1,590	$17,064	8.5%	$18,665	$1,590	$17,075	8.5%
Line4	$19,875	$1,583	$18,292	8.0%	$19,802	$1,576	$18,226	8.0%	$19,726	$1,590	$18,136	8.1%	$19,824	$1,590	$18,234	8.0%	$19,901	$1,590	$18,311	8.0%
Line5	$10,323	$792	$9,531	7.7%	$10,281	$788	$9,493	7.7%	$10,295	$795	$9,500	7.7%	$10,266	$795	$9,471	7.7%	$10,318	$795	$9,523	7.7%
Total	$83,040	$6,728	$76,312	8.1%	$83,686	$6,700	$76,986	8.0%	$83,042	$6,757	$76,285	8.1%	$83,818	$6,757	$77,061	8.1%	$83,141	$6,757	$76,384	8.1%

Paper Overview

The present paper presents three examples and, for each, several different solutions are presented. In each case, Method 1 shows how one might solve the problem using Proc Report, Methods 2 and 3 illustrate how to achieve the same goal by using Proc Tabulate with or without pre-rollup of the raw data. Then we give a brief summary on the advantages and disadvantages of the presented methods, along with their performance efficiencies in terms of CPU time.

Creating the Final Reports.

All of the methods shown in this paper were designed to produce reports in Excel, but the methods themselves are applicable for producing any type of output, such as Excel, PDF, HTML and RTF. To produce any of the following reports in an Excel format, only three lines of code have to precede the Proc Report or Proc Tabulate code, and two lines have to follow the code. An example of those five lines of code is shown here:

```
Filename Report   "File path/PID1266_Sales_Report.xls";
ODS listing close;
ODS MSOFFICE2K  file= Report  style= journal ;
<your Proc Report or Proc Tabulate code>
ODS  _All_ close;
ODS  listing;
```

For case A, the required ratio and difference of column sums are derived from two independent analysis variables Sales and Refund. Therefore it is relatively easy to fulfill, we can realize it through below approaches.

Method A1: Proc Report Approach.

The first approach is to utilize Proc Report and ODS to create the desired summary report involving the ratio and difference of column sums. We can apply this procedure directly on the detail level data to create the final summary report without any pre-summarization or pre-rollup.

The SAS program shown below illustrates how to build the Excel summary report with Proc Report and ODS. We choose ODS MSOFFICE2K as our output destination because it conforms to the Microsoft HTML specification for HTML files. In the Proc Report procedure, we first use Column statement to describe the structure and arrangement of all columns and column headings. Please note that the Fiscal_Year column will span across the 4 numeric columns of Sales, Refund, Net_Sales, and Refund_Ratio, and the quoted text is the heading for each column. The following Define statements declare the features and attributes of each column. ProductLine is defined as a group variable while Fiscal_Year as an across variable, Sales and Refund are defined as analysis variables and their sums are wanted. Net_Sales and Refund_Ratio, which are not in the input data set WORK.Raw, are defined as computed variables and they will appear in the final report. Then Compute Blocks are utilized to compute the difference and ratio of column sums, and the total summary at the end of the report.

```
************************ Method A1: Proc Report Approach.************************;
Filename Report "Path/PID1266_WLMU_Sales_Report.xls";
ODS listing close;
ODS MSOFFICE2K  file= Report style= journal ;
Title  "Five Year Sales and Marketing Summary Report of WLMU Company: Case A.";

proc report data= Raw nowindows;
column ProductLine Fiscal_Year, ( ("Sales" Sales) ("Refund" Refund)  ("Net Sales"
Net_Sales)  ("Refund Ratio (% of Sales)" Refund_Ratio ) );

define ProductLine/'Product Line' group;
define Fiscal_Year/' ' across ;
define Sales/ sum '$'  f=dollar12.0 ;
define Refund/ sum '$' f=dollar12.0;
define Net_Sales/ computed '$' f=dollar12.0;
define Refund_Ratio/ computed '%' f=percentn9.1;

compute Net_Sales;
_C4_=_C2_ - _C3_;                    _C8_=_C6_ - _C7_;
_C12_=_C10_ - _C11_;                 _C16_=_C14_ - _C15_;
_C20_=_C18_ - _C19_;
endcomp;

compute Refund_Ratio;
_C5_=_C3_ / _C2_;                    _C9_=_C7_ / _C6_;
_C13_=_C11_ / _C10_;                 _C17_=_C15_ / _C14_;
_C21_=_C19_ / _C18_;
endcomp;

rbreak after / summarize;
compute ProductLine;
if _break_='_RBREAK_' then ProductLine='Total';
endcomp;
run;
ODS  _All_ close;   ODS  listing;
```

To use Compute Blocks to achieve our goal, the key point is to reference report items by the column number[1]--- the position of the column from the left edge of the report including both the input dataset columns and computed columns, no matter whether they appear in the final report or not. Due to the across spanning structure of the report, we must reference and calculate the columns by column number rather than by column name or alias. When the Column statement is executed, SAS will create 1 + 5 × 4 = 21 columns in total at back stage. These columns are named according to their horizontal position: _C1_ column corresponds to the group variable ProductLine, _C2_, _C3_, _C4_, _C5_ columns are for the **Sales, Refund, Net_Sales, Refund_Ratio** columns in the Fiscal_Year of 1998. We can therefore use Compute Block to reference and compute the Net_Sales and Refund_Ratio as:

```
_C4_=_C2_ - _C3_;
_C5_=_C3_ / _C2_;
```

Similar patterns and formulas are applied to reference and calculate the columns for the other years.

This direct Proc Report approach can be used to calculate the sum, difference, product and ratio of column sums with the flexible Compute Block tool. However, it requires very sophisticated SAS programming skills to build the summary report because we need to use the advanced ACROSS snapping, Compute Block and column number reference techniques. Therefore, the SAS coding can be very challenging and time-consuming if we need design and construct a complicated summary report.

Method A2: Proc Tabulate Approach With Pre-Rollup.

The second solution to Case A is to utilize Proc Tabulate and ODS. In this approach, we need to first use Proc SQL, Proc Means or a DATA step to rollup the detail level data to the desired summarization level, and then utilize Proc Tabulate and ODS to create the final report. The following program illustrates this approach.

```
*************** Method A2: Proc Tabulate Approach with Pre-Rollup.**************;
proc format fmtlib;
picture tabpct  (round)
-100-<0='009.9%' (prefix='-')
0-100  ='009.9%';
run;

proc means data= Raw  noprint;
class ProductLine Fiscal_Year;
var  Sales Refund;
types  Fiscal_Year ProductLine*Fiscal_Year ;
output out=A2_Rollup(drop=_Freq_)    Sum= ;
run;

data A2_Rollup;
set A2_Rollup;
if _Type_=1 then ProductLine= "Total";

Net_Sales= Sales - Refund;
Refund_Ratio= Refund/Sales;
format Net_Sales dollar12.0 Refund_Ratio percentn9.1;
run;

proc tabulate data= A2_Rollup;
class ProductLine Fiscal_Year /missing;
var   Sales Refund Net_Sales  Refund_Ratio;
table ProductLine='Product Line',
      Fiscal_Year=' '*(Sales='Sales'*SUM='$'*f=dollar12.0
      Refund='Refund'*SUM='$'*f=dollar12.0
      Net_Sales='Net Sales'*SUM='$'*f=dollar12.0
      Refund_Ratio='Refund Ratio(% of Sales)'*SUM='%'*f=percentn9.1 )/misstext=' '
row=float;
run;
```

As shown above, firstly we use Proc Format to create a custom PICTURE format Tabpct. When we use the percentage statistics (PCTN, PCTSUM, ROWPCTN, COLPCTN, ROWPCTSUM, COLPCTSUM) in Proc Tabulate, Proc Tabulate will multiply the calculations by 100 automatically, therefore we can NOT use a SAS PERCENT format with Proc Tabulate. For this reason, we need to define a custom PICTURE format to display the percentage properly. Proc Means is chosen to rollup the raw data because it can group and analyze data for all the combinations of class variables in one step. TYPES statement is used to control the grouping of class variables, since we only need the combinations of ProductLine*Fiscal_Year (sums for each product line and fiscal year) and Fiscal_Year (totals for all product lines). The rolluped data is output to A2_Rollup data set and the following DATA step is utilized to compute the sum difference Net_Sales and sum ratio Refund_Ratio. Then we employ Proc Tabulate merely to write out the summarized results and generate the final report, which is identical to the report generated by Method A1.

This Proc Tabulate approach is flexible and versatile, it can compute the sum, difference, product and ratio of column sums. Although it requires pre-rollup, it is easy and straightforward in programming as compared to Proc Report method.

Method A3: Proc Tabulate Approach Without Pre-Rollup.

We can also apply Proc Tabulate to work on the detail level data directly, without any pre-rollup. The code below shows this more direct approach with Proc Tabulate.

```
************** Method A3: Proc Tabulate Approach Without Pre-Rollup. *************;
data A3_Difference;
set Raw;
Net_Sales=sum(Sales, -Refund);
run;

proc tabulate data=A3_Difference ;
class ProductLine Fiscal_Year /missing;
var   Sales Refund Net_Sales;
table ProductLine='Product Line' All='Total',
      Fiscal_Year=' '*(Sales='Sales'*SUM='$'*f=dollar12.0
                Refund='Refund'*SUM='$'*f=dollar12.0
               Net_Sales='Net Sales'*SUM='$'*f=dollar12.0
                Refund='Refund Ratio (% of Sales)'*PCTSUM<Sales>='%'*f=tabpct. )
/misstext=' ' row=float;
run;
```

In nature, Proc Tabulate does not compute quotients of column sums, therefore we can NOT use it to calculate ratios of column sums. This is one of its inherent limitations. However, we can regard a ratio as a special percentage, and we know that the PCTSUM statistic can calculate the percentage that one sum represents of another sum if we define the denominator variable[2]. Therefore, we can use Proc Tabulate to compute the quotient of column sums (i.e. column sum ratio) in this tricky way. This is the key point in our application of Proc Tabulate without pre-rollup. Similarly, we can use PCTN statistic to compute the ratio of frequency counts as well.

VAR1*PCTSUM <VAR2> **VAR1*PCTN <VAR2>**

Please note: The two analysis variables Var1 and Var2 must be previously declared in VAR statement.

To compute the difference of column sums, there is no such shortcut that we can take advantage of. Therefore, we have to use the DATA step to create a new variable Net_Sales in the input data set, which computes the difference between Sales and Refund at detail level. We can do it in this way due to below mathematical formulas.

$$\sum(a_i - b_i) = \sum a_i - \sum b_i \qquad \sum(a_i + b_i) = \sum a_i + \sum b_i$$

Please note, here we must use SUM function rather than an arithmetic expression to calculate the difference between Sales and Refund at detail level, because SUM function can ignore missing values in case Sales or Refund has missing values. After this manipulation, we then apply Proc Tabulate on this A3_Difference data set to create the final summary report, in which Sales is defined as the percentage denominator of the PCTSUM statistic. This approach produces the identical summary report as in Methods A1 and A2, but it is easier and simpler in SAS coding as compared to Proc Report.

This Proc Tabulate approach does not require pre-rollup of raw data. It can create summary reports having the sum, difference and ratio of column sums except for the product of column sums because of $\sum(a_i \times b_i) \neq \sum a_i \times \sum b_i$ in mathematics. If we really need the product of column sums, we have to use Methods A1 or A2 to achieve it.

We also tested the relative performance efficiencies of the above three approaches with a similar but large dataset. This testing data set had 4 variables and 285627 observations. As for the single procedure performance, the CPU time of Proc Tabulate in Method A3 (0.21s) is much shorter than those of Proc Report in Method A1 (0.56s) and Proc Means in Method A2 (0.63s). However, the overall CPU time of Method A3 (0.72s) is similar to that of Method A2 (0.74s), but is slightly longer than that of Proc Report method (0.56s). This difference is largely attributed to the DATA step in Method A3, which works on detail level data and takes longer time. Therefore, when we have a very large input data set, we need consider the trade-off between coding efficiency and processing efficiency in choosing the appropriate approach.

CASE STUDY B: COMPUTE DIFFERENCE AND RATIO OF COLUMN SUMS ARISING FROM SAME ANALYSIS VARIABLE.

The methods and approaches discussed in Case Study A are applicable to compute the difference and ratio of column sums arising from different analysis variables. However, if we want to compute the difference and ratio of column sums originating from the same analysis variable, the task and solution become much more complicated and challenging than they were with Case Study A.

As shown below, we need create a five-year marketing report with annual Sales summary data for each product line. Furthermore, in order to track the business operation and identify annual sales fluctuations, we need compare the sum of Sales of each fiscal year to that of the previous fiscal year. We name this year over year sales change as YOY Change. The YOY Change is needed in both dollar value ($, column sum difference) and percentage (%, column sum ratio) relative to the sales in the previous fiscal year.

Table 3. Five Year Sales and Marketing Summary Report of WLMU Company: Case B.

	1998			1999			2000			2001			2002		
	Sales	YOY Change	YOY Change	Sales	YOY Change	YOY Change	Sales	YOY Change	YOY Change	Sales	YOY Change	YOY Change	Sales	YOY Change	YOY Change
Product Line	$	$	%	$	$	%	$	$	%	$	$	%	$	$	%
Line1	$13,582			$13,910	$328	2.4%	$13,923	$13	0.1%	$14,038	$115	0.8%	$13,725	($313)	-2.2%
Line2	$20,645			$21,186	$541	2.6%	$20,902	($284)	-1.3%	$21,036	$134	0.6%	$20,532	($504)	-2.4%
Line3	$18,615			$18,507	($108)	-0.6%	$18,196	($311)	-1.7%	$18,654	$458	2.5%	$18,665	$11	0.1%
Line4	$19,875			$19,802	($73)	-0.4%	$19,726	($76)	-0.4%	$19,824	$98	0.5%	$19,901	$77	0.4%
Line5	$10,323			$10,281	($42)	-0.4%	$10,295	$14	0.1%	$10,266	($29)	-0.3%	$10,318	$52	0.5%
Total	**$83,040**			**$83,686**	**$646**	**0.8%**	**$83,042**	**($644)**	**-0.8%**	**$83,818**	**$776**	**0.9%**	**$83,141**	**($677)**	**-0.8%**

This task is much more challenging and difficult than Case A, because the difference and ratio of column sums are actually derived from the same analysis variable Sales but at different levels of the grouping variable Fiscal_Year. Hereby we provide three methods to create the above marketing summary report.

Method B1: Proc Report Approach.

We can create the above summary report by directly applying Proc Report on the raw data, below is the SAS code. Like Method A1, we first use Column statement to define the structure and arrangement of columns. It will generate 1 + 5 × 3=16 columns in total because of column spanning, whereas _C1_ column corresponds to the group variable ProductLine, _C2_, _C3_, _C4_ to the **Sales, Sales_Change, Sales_Change_PCNT** columns in the Fiscal_Year of 1998. The following Define statements define the attributes of each column, then we use Compute Blocks and column number references to define and compute the dollar value and percentage of the YOY Change in each fiscal year. Please note that for the first report fiscal year of 1998, we must set the YOY Change to missing values since they have no reference values to compare to.

```
********************** Method B1: Proc Report Approach. **********************;
Title  "Five Year Sales and Marketing Summary Report of WLMU Company: Case B.";
proc report data= Raw  nowindows ;
column ProductLine Fiscal_Year, ( ("Sales" Sales) ("YOY Change" Sales_Change) ("YOY
Change"  Sales_Change_PCNT)) ;
define ProductLine/ 'Product Line' group;
define Fiscal_Year/ '  '  across center;
define Sales/ sum '$'  f=dollar12.0  ;
define Sales_Change/ computed '$' f=dollar12.0;
define Sales_Change_PCNT/ computed '%' f=percentn9.1;

compute Sales_Change;
_C3_=. ;                _C6_=_C5_ - _C2_;          _C9_=_C8_ - _C5_;
_C12_=_C11_ - _C8_;     _C15_=_C14_ - _C11_;
endcomp;

compute Sales_Change_PCNT;
_C4_=.;                 _C7_= _C6_ / _C2_;         _C10_= _C9_ / _C5_;
_C13_= _C12_ / _C8_;    _C16_= _C15_ / _C11_;
endcomp;

rbreak after / summarize;
compute ProductLine;
if _break_='_RBREAK_' then ProductLine='Total';
endcomp;
run;
```

Similar to Method A1, this direct Proc Report approach is very flexible and versatile in nature and can compute the sum, difference, product and ratio of column sums, but its SAS coding is quite time-consuming and challenging.

Method B2: Proc Tabulate Approach With Pre-Rollup.

The second method to create the desired report is by using Proc Tabulate with pre-rollup of the raw data similar to that in Method A2.

As indicated below, we first utilize Proc Means to roll up the raw data to the needed grouping level, then use a DATA step to compute the year over year sales change. Because DATA step is an iterative process and it processes only one observation in each iteration, therefore we have got a challenge: how can we compute the difference between Sales values in the current observation and in the previous observation? We have few solutions to solve this problem. The first solution is to use LAG or DIF function, which can returns values from a queue[3]. However, the drawback of using LAG/DIF function is that it is very tricky and not easy to control the process, especially when we have complicated conditional processing in the program. Therefore, we decide to give up on it. Instead, we can use RETAIN statement to realize it. RETAIN can prevent a variable that is created by an INPUT or assignment statement from initialization and retain its value from one iteration of the DATA step to the next[4]. Therefore, as shown in Solution I, the variable Pre_Sales is used to keep the value of Sales retained from previous iteration. Then the year over year Sales changes, Sales_Change and Sales_Change_PCNT, are computed and the record is written out to the output data set YOY_Change by the explicit OUTPUT statement. After output, we replace the value of Pre_Sales with the value of Sales in the current observation, and the new value is retained to the next iteration. In this way, the YOY sales change is computed at the rollup level. Table 4 displays the partial printout of the generated data set.

```
************** Method B2: Proc Tabulate Approach With Pre-Rollup. **************;
proc means data= Raw  noprint;
class ProductLine Fiscal_Year;
var  Sales Refund;
types  Fiscal_Year ProductLine*Fiscal_Year ;
output out=B2_Rollup(drop=_Freq_)    Sum=;
run;

************ Solution I: Use RETAIN statement. **************;
data YOY_Change;
set  B2_Rollup;
by ProductLine Fiscal_Year;
retain Pre_Sales ;
if _Type_=1 then ProductLine= "Total";
if first.ProductLine then call missing(Pre_Sales, Sales_Change, Sales_Change_PCNT);
Sales_Change      = Sales- Pre_Sales;
Sales_Change_PCNT = Sales_Change/Pre_Sales;
output;
Pre_Sales= Sales;
format Sales_Change_PCNT  percentn9.1;
run;

************ Solution II: Use Multiple SET statements. **************;
data YOY_Change;
set  B2_Rollup;
by ProductLine Fiscal_Year;
if _Type_=1 then ProductLine = "Total";
if _N_> 1 then set B2_Rollup(keep=Sales rename=(Sales=Pre_Sales ));
if first.ProductLine then call missing(Pre_Sales) ;
Sales_Change      = Sales- Pre_Sales;
Sales_Change_PCNT = Sales_Change/Pre_Sales;
format Sales_Change_PCNT percentn9.1;
run;

************** Use Proc Tabulate to Output Results to Excel file. **************;
proc tabulate data= YOY_Change;
class ProductLine Fiscal_Year /missing;
var   Sales Refund  Sales_Change Sales_Change_PCNT;
table ProductLine='Product Line',
      Fiscal_Year=' '*(Sales='Sales'*SUM='$'*f=dollar12.0
      Sales_Change='YOY Change'*SUM='$'*f=dollar12.0
    Sales_Change_PCNT='YOY Change'*SUM='%'*f=percentn9.1)/misstext=' ' row=float;
run;
```

Table 4. Partial Printout of the Generated YOY_Change Data Set.

ProductLine	Fiscal_Year	_TYPE_	Sales	Refund	Pre_Sales	Sales_Change	Sales_Change_PCNT
Total	1998	1	$83,040	$6,728			
Total	1999	1	$83,686	$6,700	83040	646	0.8%
Total	2000	1	$83,042	$6,757	83686	-644	-0.8%
Total	2001	1	$83,818	$6,757	83042	776	0.9%
Total	2002	1	$83,141	$6,757	83818	-677	-0.8%
Line1	1998	3	$13,582	$1,187			
Line1	1999	3	$13,910	$1,182	13582	328	2.4%
Line1	2000	3	$13,923	$1,192	13910	13	0.1%
Line1	2001	3	$14,038	$1,192	13923	115	0.8%
Line1	2002	3	$13,725	$1,192	14038	-313	-2.2%
Line2	1998	3	$20,645	$1,583			
Line2	1999	3	$21,186	$1,576	20645	541	2.6%
Line2	2000	3	$20,902	$1,590	21186	-284	-1.3%
Line2	2001	3	$21,036	$1,590	20902	134	0.6%
Line2	2002	3	$20,532	$1,590	21036	-504	-2.4%

An alternative solution is to use multiple SET statements as shown in the Solution II. We use the IF-THEN conditional processing to SET the same dataset B2_Rollup starting from the second iteration and rename Sales as Pre_Sales. In this way, the Pre_Sales variable actually carries on the value of Sales in the previous iteration so that we can compute the year over year sales changes. This solution produces the same results as Solution I, but it requires less SAS code.

After computing the YOY sales changes, we utilize Proc Tabulate and ODS to output the results and generate the required summary report. This Proc Tabulate approach is more straightforward and easier to code and modify than the direct Proc Report approach, and it can compute the sum, difference, product and ratio of column sums as well.

Method B3: Proc Tabulate Approach Without Pre-Rollup.

We can also apply Proc Tabulate on the detail level data without pre-rollup to generate the summary report, however, the pre-manipulation is really tricky.

```
************* Method B3: Proc Tabulate Approach Without Pre-Rollup. *************;
data B3;
set Raw(in=A)
    Raw(in=B keep=ProductLine Fiscal_Year Sales rename=(Sales=Pre_Sales));

if A=1 and Fiscal_Year > 1998 then Sales_Change=Sales;
else if B=1 then do;
    if Fiscal_Year < 2002 then do;
      Fiscal_Year= Fiscal_Year + 1;
      Sales_Change= -Pre_Sales;
    end;
    else delete;
end;
run;

proc tabulate data= B3;
class ProductLine Fiscal_Year /missing;
var   Sales Pre_Sales Sales_Change;
table ProductLine='Product Line' All='Total',
      Fiscal_Year=' '*(Sales='Sales'*SUM='$'*f=dollar12.0
      Sales_Change='YOY Change'*SUM='$'*f=dollar12.0
 Sales_Change='YOY Change'*PCTSUM<Pre_Sales>='%'*f=tabpct.)/misstext=' ' row=float;
run;
```

As illustrated above, we first use a DATA step to perform the pre-manipulation of raw data without rollup. The Sales variable in the second copy of Raw data set is renamed as Pre_Sales, and this data set is stacked to the original Raw

data set. The IN contributor is used to identify the source of each observation. If an observation comes from the original data set, then we assign the value of Sales to the newly created Sales_Change variable if the fiscal year is greater than 1998. If an observation is from the modified data set and the Fiscal_Year is not 2002, we change the value of Fiscal_Year to next fiscal year and assign the minus value of Pre_Sales to Sales_Change; otherwise we will delete the record. Also, for the first fiscal year of 1998, we set Sales_Change to missing values. After this pre-manipulation, we apply Proc Tabulate on the generated data set B3. In effect, the SUM statistic will actually produce the sum difference between Sales and Pre_Sales ($, dollar value of YOY change). Similarly, the percentage of YOY change is computed by using the PCTSUM statistic in the same way as in Method A3.

This Proc Tabulate approach generates the same results as Methods B1 and B2. It is able to compute the sum, difference and ratio of column sums except for the product of column sums. It is very concise in SAS coding.

In respect to their performance efficiencies, the durations of overall CPU time are 0.56s, 0.64s and 1.40s for Methods B1, B2 and B3 respectively based on the same testing data set in Case A. Method B3 costs longer CPU time than others mainly because of the pre-manipulation DATA step, however, it is easy and simple to code and modify in comparison with Methods B1 and B2.

CASE STUDY C: COMPUTE DIFFERENCE AND RATIO OF COLUMN SUMS REFERENCING A FIXED COLUMN.

In Cases A and B, the difference and ratio of column sums are computed based on different reference columns. Under some circumstances, we also need to compute them referencing a fixed column. For example, as demonstrated in the below summary report, we not only need the summarized sales data for each fiscal year, we also want to track the sales growths (both dollar value and percentage) in every 2 years as of the starting year 1998. In this case, all the column sum ratio and column sum difference are calculated by using the sum of sales in 1998 as the reference value. Hereby we illustrate how to fulfill it.

Table 5. Five Year Sales and Marketing Summary Report of WLMU Company: Case C.

	1998	1999	2000	2001	2002	3-Year Growth		5-Year Growth	
Product Line	**$**	**$**	**$**	**$**	**$**	**$**	**%**	**$**	**%**
Line1	$13,582	$13,910	$13,923	$14,038	$13,725	$341	2.5%	$143	1.1%
Line2	$20,645	$21,186	$20,902	$21,036	$20,532	$257	1.2%	($113)	-0.5%
Line3	$18,615	$18,507	$18,196	$18,654	$18,665	($419)	-2.3%	$50	0.3%
Line4	$19,875	$19,802	$19,726	$19,824	$19,901	($149)	-0.7%	$26	0.1%
Line5	$10,323	$10,281	$10,295	$10,266	$10,318	($28)	-0.3%	($5)	0.0%
Total	**$83,040**	**$83,686**	**$83,042**	**$83,818**	**$83,141**	**$2**	**0.0%**	**$101**	**0.1%**

Method C1: Proc Report Approach

In this case, we can create the required summary report through the direct application of Proc Report due to the flexible features of Compute Block.

```
********************** Method C1: Proc Report Approach. **********************;
Title  "Five Year Sales and Marketing Summary Report of WLMU Company: Case C.";

proc report data= Raw   split='/' nowindows;
column ProductLine Fiscal_Year, (Sales ) ('3-Year Growth' G1 G2) ('5-Year Growth'
G3 G4);

define ProductLine/ 'Product Line' group;
define Fiscal_Year/ ' '  across center;
define Sales/ sum ' $'  f=dollar12.0 ;
define G1/ computed '$'  f=dollar12.0;
define G2/ computed '%' f=percentn9.1;
define G3/ computed '$'  f=dollar12.0;
define G4/ computed '%' f=percentn9.1;

compute G1;       G1=_C4_ - _C2_;     endcomp;
compute G2;       G2=G1/_C2_;         endcomp;
compute G3;       G3=_C6_ - _C2_;     endcomp;
compute G4;       G4=G3/_C2_;         endcomp;
```

```
rbreak after / summarize;
compute ProductLine;
if _break_='_RBREAK_' then ProductLine='Total';
endcomp;
run;
```

As shown above, we apply Proc Report directly on the raw data to create the final summary report. The arrangement, structure, headings and attributes of columns are defined by the Column statement and following Define statements. Then the difference and ratio of column sums relative to the sum of sales in 1998 are computed by the Compute Block. This flexible method is applicable to compute all kinds of arithmetic calculations of column sums.

Method C2: Proc Tabulate Approach With Pre-Rollup

In addition to using Proc Report, we can also employ Proc Tabulate to produce the summary report. However, in this case, we must pre-rollup the raw data before we apply Proc Tabulate. Pre-rollup is mandatory rather than optional to realize it.

```
*************** Method C2: Proc Tabulate Approach with Pre-Rollup.**************;
proc means data= Raw  noprint;
class ProductLine Fiscal_Year;
var  Sales Refund;
types  Fiscal_Year ProductLine*Fiscal_Year ;
output out=C2_Rollup(drop=_Freq_)    Sum=;
run;

data C2_Compute;
set C2_Rollup;
by ProductLine Fiscal_Year;
retain Ref_Value;

if _Type_=1 then ProductLine= "Total";
if first.ProductLine then N=0;
N+1;
if N=1 then Ref_Value = Sales;
if N>1 and  mod(N, 2) =1 then  do;
Duration = N;
Growth=Sales - Ref_Value;
Growth_PCNT= Growth/Ref_Value;
end;
else call missing(Growth, Growth_PCNT);

format Growth Ref_Value dollar12.0 Growth_PCNT percentn9.1;
drop N Ref_Value ;
run;

proc format fmtlib;
value duration
3="3-Year Growth"
5="5-Year Growth" ;
run;

proc tabulate data= C2_Compute  ;
format Duration duration.;
class ProductLine Fiscal_Year Duration /missing;
var   Sales Growth Growth_PCNT;
table ProductLine='Product Line',
      Fiscal_Year=' '*(Sales=' '*SUM='$'*f=dollar12.0)
      Duration=' '*(Growth=' '*SUM='$'*f=dollar12.0
      Growth_PCNT=' '*SUM='%'*f=percentn9.1) /misstext=' ' row=float;
run;
```

As illustrated above, we first use Proc Means to rollup the raw data to the grouping level, and then utilize a DATA step to compute the sales growths in every 2 years. In the DATA step, the Sales value in the first fiscal year of each product line is assigned to a new variable Ref_Value, and it is retained to all the other fiscal years by the RETAIN

statement. The SUM statement is to create a counter variable N for each product line. The IF-THEN conditional processing will control the computation of sales growths (both dollar value and percentage) relative to the reference sales value when the counter variable N has an odd value. We then run Proc Tabulate on the generated data set C2_Compute to create the desired report.

This Proc Tabulate approach produces the same report as Method C1 with the advantage of being easier to code and modify. Testing with the same data set in Case A, its overall CPU time turns out to be 0.63s, while the overall CPU time of Method C1 is 0.52s. Therefore, this Proc Tabulate approach is comparable to the Proc Report approach in performance efficiency.

CONCLUSION

As discussed in this paper, we can use different methods and approaches to overcome the limitations of Proc Tabulate, a powerful tabulation tool. Therefore, we can rise above the hindrance and extend its use to create needed reports, and still enjoy its simplicity and convenience in SAS coding and table construction. Although the three cases are based on two grouping variables, however, all the approaches are applicable and generalizible to more grouping variable cases. The illustrated methods and skills can have important applications in a wide variety of fields such as customer data analyses, business analytics, marketing research and scientific research areas.

REFERENCES

[1] SAS Support Website:
http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146851.htm

[2] SAS Support Website,
http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000146762.htm

[3] SAS Support Website
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000212547.htm

[4] SAS Support Website,
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000214163.htm

ACKNOWLEDGEMENTS

Special thanks to Dr. Arthur Tabachneck (myQNA Inc.) and Cynthia Zender (SAS Institute) for their valuable feedback and suggestions during reviewing and proof-reading this paper. We are very grateful to them for their great help and support. Thank you so much.

CONTACT INFORMATION

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

Justin Jia
Customer Marketing and Strategies, CIBC, Canada
Email: justin.jia@cibc.com

Amanda Lin
Risk Management, Bell Canada
Email: amanda_shan_shan.lin@bell.ca