Wow, It's Sooo Simple
(My First Encounter with Proc Report)

Submitted by Angie C. Wang
Coopers & Lybrand Consulting, San Francisco, CA

I Abstract

As a Consultant for Coopers & Lybrand, I use SAS to develop many applications at client sites. At CitiBank, I was asked to develop a reporting system that measure the cost-effectiveness and response rate of direct marketing campaigns. As a result of this project, I learned to use both Proc SQL and Proc Report for the first time. I was impressed by how easy it was to learn as well as how good the layout looked without a big time investment. This paper will briefly describe the development process and then will focus on how Proc SQL was used to manipulate the data and how Proc Report was used to create the report and its ease of use.

II Introduction

The Direct Marketing Performance Measurement (DMPM) system was developed at CitiBank to give the direct marketing program managers an automated, standardized way to measure the cost effectiveness and response rate of direct marketing campaigns. Six key areas are shown on the report; quantity promoted, marketing cost, number of accounts booked, response rate, total dollars booked, average dollars booked and cost per account. In addition, the response rate is cross-tabbed with geo-demographic attributes such as market, segment, income, marital status, etc.

A direct mail tape is created containing customer and campaign information; it is sent to a vendor who produces the actual mailing. The information includes the household number, customer number, current bank services and an 19 byte source code. The source code contains information specific to a campaign such as market, segment, offer, package, etc.

The customer information on the mail tape is used to extract data from multiple DB2 tables using COBOL. The extract is compared with the mail tape to determine if a customer opened an account within the correct time period and with the appropriate services. Flags are set and a SAS file used for reporting is created.

III Development Process

An Excel mock-up was given to the users. The report contained an executive summary, 3 levels of detail and a geo-demographic analysis. The geo-demo analysis contained attributes of the responders versus non-responders for campaign specific items such as offer, packages, etc. In the mock-up, the Level 1 and the Geo-demographic analysis were shown as rows whereas the data was really in columns as different variables. Lastly, appearance was a major concern to the client. The client was a knowledgeable SAS user which made the communication and development process fairly smooth. He decided that Proc Report was the way to go since it looks better than Proc Print and we wanted to avoid the much more time consuming Data Null.

The transformation of columns into rows was a major challenge. One solution was to

perform multiple summaries and concatenate the results. However, we were looking for a more elegant solution. Initially, we thought to let Proc Report perform all the calculations. We first tried using the _Cn_ designation. The results were not correct. However, we did not take the time to investigate what the logical error was, we just tried another procedure. That was the condition under which I worked.

I then tried Proc Tabulate. The results were correct but "header" information was repeated for each table. I changed the option to suppress the print and got too many blank lines instead. In addition, I could not control where the page breaks occurred. The next thing I tried was Proc SQL and it did the trick. In addition to the geo-demo and Level 1 portions, we decided to use Proc SQL to summarize the data into the other levels as well.

The Level 1 analysis contained campaign specific information for each campaign and it varies from campaign to campaign. Since the campaign information varied and we were processing multiple campaigns at the same time, we made the SQL query into a macro with macro variables for parameters. We can invoke it any number of times, passing it different campaign specific parameters each time, thereby creating multiple tables and joining them together with the union command.

The same approach was used for the geo-demographic analysis as well. The macro for the SQL query was a little more complex. Lastly, since we were processing multiple campaigns, we also made the Proc Report code into macros, one for each level of summary with a macro variable for the campaign.

III Turning Columns into Rows

How often do you run into the situation where the data you want is in variables, but the user wants them displayed as the rows within the report. In order to give the users what they want, you probably had to do a lot of awkward manipulations. Figure 1 shows the original data layout and how we want to manipulate it

I needed the following, the number of households, the number of respondents, the booking rate, total funds booked, average funds booked, marketing costs and the cost of acquisition for each account. Some of these fields were available in the report file, some had to be calculated. If I were to use multiple Proc Summaries, I would have had to create a counter ahead of time or use the _FREQ_ variable at the appropriate level for the number of households. For the calculated fields, I also had to create them ahead of time. For each campaign with different campaign specific details, this had to be repeated.

Due to the repetition, we knew it had to be a macro. By using Proc SQL, the counter and the calculated fields were create at the same time. There were also 2 macro variables, one was the campaign specific source code, one was the name of the results table. Another variable was also created using the source code item. This was later used as the format in the Proc Report. The macro is invoked multiple times; the resultant tables are then joined together with another SQL statement using the Union command. The final table is then used in the Proc Report. Figure 2 shows the Proc SQL code that changed columns into rows.

III Proc Report - Some Basics

When Proc Report first came out, the demonstrations were in either Display Manager or Windows. I tried it a few times in Display Manger and was not very

successful. This time, I used Proc Report in a batch environment and was able to produce very professional looking reports in a short time. After using Proc SQL to manipulate my data, I am now ready to produce my report. I'll describe the basic components and some simple options that I used to produced the report shown in Appendix 1.

The 2 main parts of Proc Report are the COLUMN and DEFINE statements. The Column statement simply list the columns of your report in the order that you wish them to appear. This includes computed columns as well as those columns that will not be printed but will be used for the computations.

The DEFINE statement and its various options are what gives your report the appearance you want. For example, you can specify various attributes, such as width and format as well as justification of your column headers. You can give your column headers any label you want just by enclosing them in quotes.

Most importantly, you use the Define statement to specify how the column is to be used in the report. The 3 that I used the most often are, Group, Analysis and Computed. The usage of Group is for "Class" variables whereas Analysis and Computed are for the numeric variables.

Or, when a column is defined as a Group, it is used to group values together giving you a summary of the numeric variables that you have defined as Analysis or Computed.

When you need to create a calculated field, the column is defined as Computed. A COMPUTE block follows in which the formula is given; it is ended with a ENDCOMP statement. Proc Report can give you simple statistics that you can use in your calculations without having to create them seperately. These statistics within the compute block are referred to by their compound name. Figure 3 contains the Proc Report code that produced the report shown in Appendix 1.

Lastly, if you want subtotals, you can use the BREAK statement for the columns that you have defined as groups. There are options to specify such as under or over lining, summarization, page breaks, etc.

As the title says, It's Sooo Simple. Just start with a simpl report and you'll soon be using it to create all your reports.

ACKNOWLEDGEMENT

I like to thank Lawrence Kwok for all his help with my coding during this project.

FIGURE 1

SAMPLE DATA

BEFORE

HHNO CAMPAIGN LISTCD1 PACKAGE MKTCD SEGMENT RESPDNT NONRESP RESPBAL COSTPCE

AFTER

CAMPAIGN	ATTRCODE	QTYPROMO	ACCTBOOK
	(LISTCD1)	(HHNO)	(RESPDNT)
	(PACKAGE)		
	(MKTCD)		

FIGURE 2

```
%MACRO CRETBL(TBLNAME, COLNAME);

PROC SQL;
        CREATE TABLE &TBLNAME AS
        SELECT CAMPAIGN,
        "&COLNAME" AS ATTRCODE, &COLNAME AS VARCODE,
        COUNT(HHNO) AS QTYPROMO,
        SUM(RESPDNT) AS ACCTBOOK,
        SUM(RESPDNT)/COUNT(HHNO) AS BOOKRATE.
        SUM(RESPBAL) AS FUNDBOOK,
        ROUND(SUM(RESPBAL)/SUM(RESPDNT)) AS AVGFUN,
        ROUND(COUNT(HHNO)*MEAN(COSTPCE)) AS MKTCOST,
        (ROUND(COUNT(HHNO)*MEAN(COSTPCE)))/SUM(RESPDNT) AS ACQCOST,
FROM REPFILE
GROUP BY CAMPAIGN, &COLNAME
ORDER BY CAMPAIGN, &COLNAME;
%MEND CRETBL;

%CRETBL(TABLE1, LISTCD1);
%CRETBL(TABLE2, PACKAGE);
%CRETBL(TABLE3, MKTCD);
%CRETBL(TABLE4, SEGMENT);
%CRETBL(TABLE5, OFFER);

PROC SQL;
        CREATE VIEW LVLRPT AS
        SELECT * FROM TABLE1
        UNION
        SELECT * FROM TABLE2
        UNION
        SELECT * FROM TABLE3
        UNION
        SELECT * FROM TABLE4
        UNION
        SELECT * FROM TABLE5;
```

FIGURE 3

```
%MACRO LEVEL1(CAMPGN);

PROC FORMAT CNTLIN=FMTLIB.F&CAMPGN;

PROC REPORT NOWINDOWS DATA=LEVEL1 HEADSKIP;
        WHERE CAMPAIGN = &CAMPGN;

        COLUMN          CAMPAIGN ATTRCODE NUCODE QTYPROMO ACCTBOOK RATE PERCOL  BOOKRATE
                        FUNDBOOK AVGFUN AVGFUND COSTPCE TOTCOST AVGCOST COSTACQ;

DEFINE CAMPAIGN         / GROUP FORMAT=CAMPGN. RIGHT WIDTH=18 'CAMPAIGN';
DEFINE ATTRCODE         / GROUP LEFT WIDTH=16 'SOURCE CODE / COMPONENT' FORMAT=$ATRCODE.;
DEFINE NUCODE           / GROUP LEFT WIDTH=15 ' ' FORMAT=$NUCODE. ORDER=INTERNAL;
DEFINE QTYPROMO         / SUM FORMAT=COMMA9. WIDTH=9 CENTER 'QUANTITY / PROMOTED';
DEFINE ACCTBOOK         / SUM FORMAT=COMMA7. WIDTH=7 CENTER '# ACCTS / BOOKED';
DEFINE BOOKRATE         / ANALYSIS SUM NOPRINT;
DEFINE RATE             / COMPUTED FORMAT=6.3 WIDTH=7 'BOOKING RATE' CENTER;
DEFINE PERCOL           /  COMPUTED FORMAT=$1. WIDTH=1 SPACE=0 ' ';
DEFINE FUNDBOOK         / SUM FORMAT=DOLLAR11. WIDTH=12 CENTER 'TOTAL FUNDS / BOOKED';
DEFINE AVGFUN           / ANALYSIS SUM NOPRINT;
DEFINE AVGFUND          / COMPUTED FORMAT=DOLLAR11. WIDTH=13 CENTER  'AVERAGE FUNDS / BOOKED';
DEFINE COSTPCE          / ANALYSIS MEAN NOPRINT;
DEFINE TOTCOST          / ANALYSIS SUM NOPRINT;
DEFINE AVGCOST          / COMPUTED FORMAT=DOLLAR11. WIDTH=12 CENTER   'MARKETING / COST';
DEFINE COSTACQ          / COMPUTED FORMAT=DOLLAR11. WIDTH=12 CENTER   'COST PER ACCT / BOOKED';

COMPUTE COSTACQ;
        COSTACQ = (QTYPROMO.SUM*(TOTCOST.SUM/QTYPROMO.SUM))/ACCTBOOK.SUM;
ENDCOMP;

COMPUTE AVGCOST;
        AVGCOST = (QTYPROMO.SUM*COSTPCE.MEAN);
ENDCOMP;

COMPUTE PERCOL / LENGTH=1 CHAR;
        PERCOL = '%';
ENDCOMP;

COMPUTE RATE;
        RATE = (ACCTBOOK.SUM/QTYPROMO.SUM)*100;
ENDCOMP;

COMPUTE AVGFUND;
        AVGFUND = FUNDBOOK.SUM/ACCTBOOK.SUM;
ENDCOMP;

BREAK AFTER ATTRCODE  / OL SKIP SUMMARIZE SUPPRESS;
BREAK AFER CAMPAIGN   / PAGE;

%MEND LEVEL1;
```

1 DIRECT MARKETING PERFORMANCE MEASUREMENT SYSTEM

REPORT: BOOKING RATE ANALYSIS REPORT - LEVEL I

RUNDATE: 08APR94
DATA MONTH-FEBRUARY 1994

PROGRAM: 4Q93 CMMA LAUNCH
COST PER PIECE IS $1.86, EVALUATION PERIOD - 10/06/93 TO 12/15/93

CAMPAIGN	SOURCE CODE COMPONENT		QUANTITY PROMOTED	# ACCTS BOOKED	BOOKING RATE	TOTAL FUNDS BOOKED	AVERAGE FUNDS BOOKED	MARKETING COST	COST PER ACCT BOOKED
4Q93 CMMA LAUNCH	DROP	FIRST DROP	31,636	39	0.123 %	$61,809	$1,585	$58,843	$1,509
		SECOND DROP	31,653	34	0.107 %	$60,495	$1,779	$58,875	$1,732
		THIRD DROP	31,596	29	0.092 %	$156,064	$5,382	$58,769	$2,027
		FOURTH DROP	31,574	22	0.070 %	$68,619	$3,119	$58,728	$2,669
			126,459	124	0.098 %	$346,986	$2,798	$235,214	$1,897
	LIST CODE 1	RETAIL	18,159	61	0.336 %	$193,139	$3,166	$33,776	$554
		BANKCARD	108,300	63	0.058 %	$153,848	$2,442	$201,438	$3,197
			126,459	124	0.098 %	$346,986	$2,798	$235,214	$1,897
	MARKET	CBFL	36,506	21	0.058 %	$36,081	$1,718	$67,901	$3,233
		CBMA	16,459	29	0.176 %	$166,985	$5,758	$30,614	$1,056
		CBCA	71,096	74	0.104 %	$143,920	$1,945	$132,239	$1,787
		CBNV	2,398	0	0.000 %	$0	0	$4,460	0
			126,459	124	0.098 %	$346,986	$2,798	$235,214	$1,897
	PACKAGE	LEAD W/ACCT OPE	57,762	74	0.128 %	$170,297	$2,301	$107,437	$1,452
		ACCT OPEN/SAVE	52,233	41	0.078 %	$161,727	$3,945	$97,153	$2,370
		ACCT OPEN/REST	16,464	9	0.055 %	$14,962	$1,662	$30,623	$3,403
			126,459	124	0.098 %	$346,986	$2,798	$235,214	$1,897
	SEGMENT	T1-3/INVEST	520	2	0.385 %	$558	$279	$967	$484
		T1-3/ESA	1,235	12	0.972 %	$97,797	$8,150	$2,297	$191
		T1-3/BANKCARD	11,292	19	0.168 %	$43,868	$2,309	$21,003	$1,105
		T1-3/CHECK ONLY	5,112	28	0.548 %	$50,915	$1,818	$9,508	$340
		BANKCARD ONLY	108,300	63	0.058 %	$153,848	$2,442	$201,438	$3,197
			126,459	124	0.098 %	$346,986	$2,798	$235,214	$1,897

PREPARED BY DATABASE MARKETING
CALL VINCE BOWEY (510) 268-5497 WITH QUESTIONS