Customizing a Correlation Matrix That Would Make Pearson Proud

Aaron Hill, MDRC, New York, NY

ABSTRACT

PROC CORR is a great tool for exploratory analysis, statistical modeling, and data screening. However, with more than ten variables in a correlation matrix, output can be difficult to read. This paper will describe how to use PROC REPORT and ODS to produce a continuous, unbroken matrix formatted with visual cues and traffic-lighting. The final result: a visual masterpiece with maximum readability and utility.

INTRODUCTION

Exploratory analysis and statistical modeling usually involves searching for correlations among many variables. PROC CORR computes correlation statistics and displays them in a matrix. But when there are more than ten variables, the listing output from PROC CORR becomes so extensive that the matrix breaks and wraps over several pages. This makes the listing output hard to read and interpret. The solution: output a dataset from the CORR procedure, use PROC REPORT to control what is displayed and how it is formatted, and the Output Delivery System (ODS) to write an HTML file that can be read with Microsoft Excel.

The dataset used in the examples is SASHELP.CITIMON. This is a sample financial sector dataset, available from SAS®, from which 18 numeric variables will be used to search for correlations. When using the CORR procedure to generate a matrix of these 18 variables, SAS generates four pages of output, breaking the matrix into four quadrants. To adapt this output to meet our needs, we want to see everything in one window, print it on a single page, suppress the duplicate correlations, and visually flag correlations above a specified threshold.

A MATRIX READABLE BY EXCEL IS EASY TO GENERATE

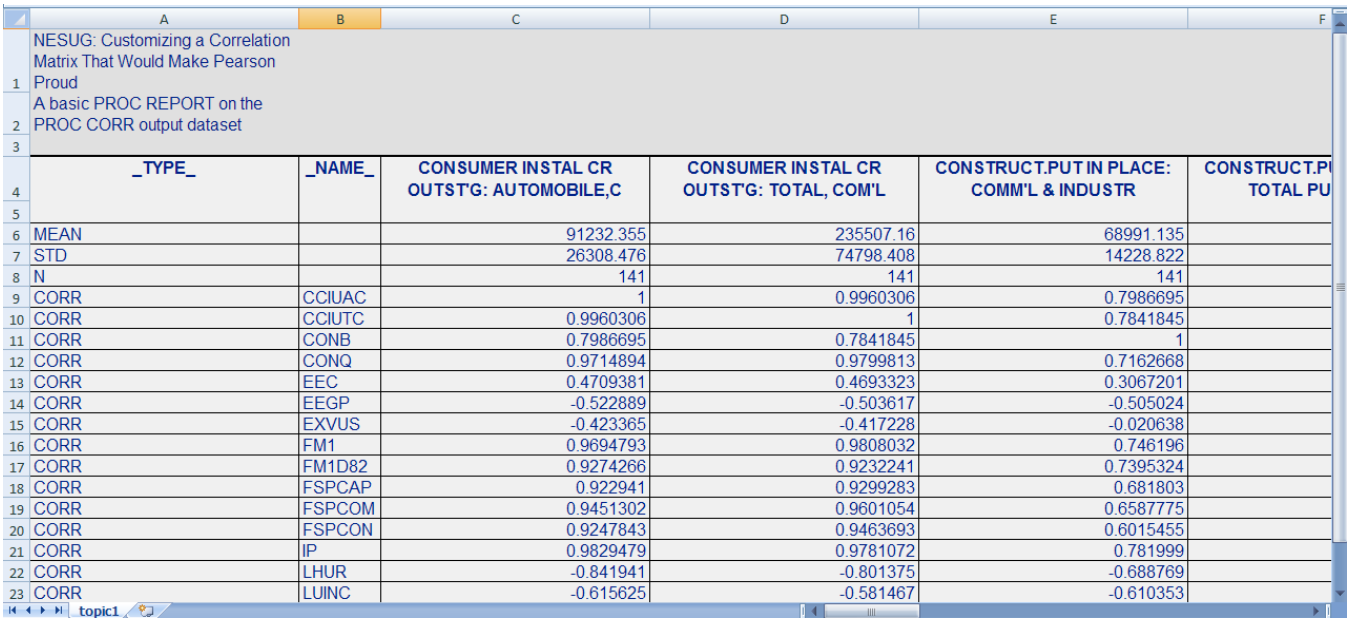
We will use the CORR procedure to generate a correlation matrix and output it to a new dataset. PROC CORR has several output dataset options allowing a choice of correlation statistic; this paper will discuss only the Pearson's correlation statistic (OUTP=). The output dataset includes a new variable, _TYPE_, that specifies whether an observation is a simple statistic (MEAN, STD, N) or a correlation statistic (CORR). Another new variable, _NAME_, essentially serves as 'row headers' because the 'CORR' observations are structured in the same order as the correlation matrix. Therefore, it is important not to sort or change the order of the output dataset.

The following code generates the correlation matrix and outputs it to HTML, which can be read by Microsoft Excel. The *.xls* extension is used in naming the file so that the default program to open the file is Excel.

```
proc corr data = sashelp.CITIMON outp=citimon2 nomiss;
var CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP FSPCOM FSPCON IP
LHUR LUINC PW RCARD RTRR;
title2 'Standard listing output from PROC CORR';
run;

ods listing close;
ods html file = "C:\SAS\NESUG\topic1.xls";

proc report data = citimon2;
title 'NESUG: Customizing a Correlation Matrix That Would Make Pearson
Proud';
title2 'A basic PROC REPORT on the PROC CORR output dataset';
run;

ods html close;
ods listing;
```

Figure 1 shows a portion of the HTML file opened by Excel. It is important to acknowledge that HTML is not the only ODS format for sending SAS output to Excel. In fact, some would argue that it isn't the best option for sending SAS output to Excel. Admittedly, other ODS options, especially tagsets, allow you to exert more control

over the format. There are some downsides to writing HTML: Excel interprets your SAS formats as it sees 'best' and many SAS formatting instructions are not valid in ODS HTML. But, there are some important advantages to outputting to HTML: it's simpler to program and you're less likely to encounter problems loading the file in Excel. Also, there are many 'tricks' you can use—as we will see in this paper—to get Excel to read the file so it will display your output as you want to see it. It is important to think about how SAS will *write* the HTML, but equally important to think about how Excel will *read* it.



	A	B	C	D	E	F
1	NESUG: Customizing a Correlation Matrix That Would Make Pearson Proud					
2	A basic PROC REPORT on the PROC CORR output dataset					
3						
4	_TYPE_	_NAME_	CONSUMER INSTAL CR OUTST'G: AUTOMOBILE,C	CONSUMER INSTAL CR OUTST'G: TOTAL, COM'L	CONSTRUCT.PUT IN PLACE: COMM'L & INDUSTR	CONSTRUCT.PI TOTAL PU
5						
6	MEAN		91232.355	235507.16	68991.135	
7	STD		26308.476	74798.408	14228.822	
8	N		141	141	141	
9	CORR	CCIUAC	1	0.9960306	0.7986695	
10	CORR	CCIUTC	0.9960306	1	0.7841845	
11	CORR	CONB	0.7986695	0.7841845	1	
12	CORR	CONQ	0.9714894	0.9799813	0.7162668	
13	CORR	EEC	0.4709381	0.4693323	0.3067201	
14	CORR	EEGP	-0.522889	-0.503617	-0.505024	
15	CORR	EXVUS	-0.423365	-0.417228	-0.020638	
16	CORR	FM1	0.9694793	0.9808032	0.746196	
17	CORR	FM1D82	0.9274266	0.9232241	0.7395324	
18	CORR	FSPCAP	0.922941	0.9299283	0.681803	
19	CORR	FSPCOM	0.9451302	0.9601054	0.6587775	
20	CORR	FSPCON	0.9247843	0.9463693	0.6015455	
21	CORR	IP	0.9829479	0.9781072	0.781999	
22	CORR	LHUR	-0.841941	-0.801375	-0.688769	
23	CORR	LUINC	-0.615625	-0.581467	-0.610353	

topic1

Figure 1: topic1.xls

THE OUTPUT IS BETTER, BUT NOT EXACTLY WHAT I WANTED…

There are some definite improvements in topic1.xls (Figure 1) over the listing output. Most importantly, the matrix is unbroken, which greatly improves the readability and interpretability of the matrix. But, there is also a lot of room for improvement:

- The titles cause first column to be unnecessarily wide.
- The summary statistics (MEAN, STD, and N) aren't a necessary component of the matrix.
- _TYPE_ does not need to be displayed in the matrix.
- The entire matrix—though unbroken—is far too wide to fit into a single window or print onto a single page.
- Excel chooses the best format and column width—usually dictated by the widest item in the column. The default column header is the variable label, which causes each column to be very wide.
- _NAME_ effectively serves as the 'row headers' in the correlation matrix. They should match the column headers.
- The correlation statistic values aren't formatted. We don't need to see any more detail than two places after the decimal.
- The default grey cells and blue font color could be more visually appealing.

The next section will cover making improvements to the formatting by using a temporary template and more PROC REPORT statements and options.

EXERT MORE CONTROL OVER THE FORMAT

All titles and footnotes should be suppressed so they do not appear in the HTML output. If any titles or footnotes remain, they will be embedded in cells in the first column, which will make the column much wider than needed. Later in the paper we will discuss better ways to include titles or footnotes in the Excel file with COMPUTE blocks.

```
title;
footnote;
```

A template allows you to exert control over the appearance of the output. However, many of the attributes in

DEFINE statements are not valid for HTML output. It is important to check SAS documentation for PROC TEMPLATE to identify the valid attributes. Here, we are using an extremely basic TEMPLATE procedure to strip all of the default ODS HTML styles by creating (and later applying) a temporary template, which will defer to the Excel defaults instead. We could do much more with the TEMPLATE procedure to format the output, but here, we're going for simplicity—and the Excel defaults are very simple. The single definition we make in this PROC TEMPLATE step is a STYLE statement that defines borders around the cells.

```
proc template;
define style corrmatx;
   style table / borderwidth=10;
end;
run;
```

When formatting options are not valid in ODS HTML, or, if you are not happy with the way Excel interprets the format, you can sometimes remedy this by supplying a format that Excel will recognize. In the following code we are going to specify two Excel number formats (Microsoft CSS attributes). These formats—though not used in SAS—are embedded in the output HTML, and recognized by Excel when it reads the file. The definitions are preceded by **mso-** and some are only recognized by Excel and/or other Microsoft Office applications.

In the ODS HTML statement, we're going to add a HEADTEXT= option to apply the CORRMATX temporary template we just created. Also in this statement, we can specify the style of the TD tag, a tag recognized by Excel that defines the format of a standard cell in an HTML table. In this case, we're going to use **mso-number-format:\@** which converts all numeric cell values to text. This is useful because it gives you more control over how numeric values are actually displayed. Although you can specify SAS formats when writing the output, Excel decides the 'best' way it should be formatted. As a result, leading and trailing zeros can be omitted (for example, *-1.80* output with the SAS format 5.2 would be displayed in Excel as *-1.8*). Converting it to text allows you to display it exactly as you intended.

```
ods html file = "C:\SAS\NESUG\topic2.xls"
headtext="<style> td {mso-number-format:\@} </style>"  style=corrmatx;
```

In the PROC REPORT statement, we can specify the style of the column headings with a CSS attribute. In this case, we're going to use **mso-rotate:90** in a STYLE= option. This will rotate the column headers by 90 degrees. We are doing this because the width of the column in Excel will largely be determined by the widest item in that cell. Since our correlation values are ultimately going to be displayed using a format with a width of five, any column headers wider than five will cause the column to expand. The result would be different sized columns. Rotating the headers will equalize the width of the columns, making the matrix easier to read. It is also helpful to specify the height of the header, to prevent text wrapping. In this example, **height:50pt** is appropriate.

```
proc report data = citimon2
style(header)={htmlstyle="mso-rotate:90; height:50pt"};
```

Restricting the observations to the correlation statistics removes the unnecessary summary statistics from the matrix.

```
where _type_ = 'CORR';
```

To gain more control over how ODS HTML writes a file you intend to open in Excel, we will use a COLUMN statement to define the arrangement of all columns and DEFINE statements to specify how to use and display the report items. It is important in the COLUMN statement to maintain the variable order in which they were originally arranged in the output dataset! Otherwise, the matrix will be jumbled.

In each of the DEFINE statements we are replacing the default column headers (variable labels) with specified values. (I like to use variable names because they are concise.) In the first DEFINE statement, we are removing the column header altogether and formatting the values to bold; the first column contains the values that serve as the 'row headers' and they should match the style of the column headers. In the subsequent DEFINE statements, we specify the column headers as the variable name and apply the SAS format of 5.2. These DEFINE statements are very repetitive in nature—which would lend itself nicely to code generation with a macro. Remember, not all

attributes specified would be valid in ODS HTML. Common attribute definitions such as WIDTH= in the DEFINE statements would be ignored.

```
column _name_ CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP FSPCOM
FSPCON IP LHUR LUINC PW RCARD RTRR;
define _name_ / display '' style=[font_weight=bold];
define CCIUAC / display 'CCIUAC' format = 5.2;
define CCIUTC / display 'CCIUTC' format = 5.2;
define CONB   / display 'CONB'   format = 5.2;
define CONQ   / display 'CONQ'   format = 5.2;
define EEC    / display 'EEC'    format = 5.2;
define EEGP   / display 'EEGP'   format = 5.2;
define EXVUS  / display 'EXVUS'  format = 5.2;
define FM1    / display 'FM1'    format = 5.2;
define FM1D82 / display 'FM1D82' format = 5.2;
define FSPCAP / display 'FSPCAP' format = 5.2;
define FSPCOM / display 'FSPCOM' format = 5.2;
define FSPCON / display 'FSPCON' format = 5.2;
define IP     / display 'IP'     format = 5.2;
define LHUR   / display 'LHUR'   format = 5.2;
define LUINC  / display 'LUINC'  format = 5.2;
define PW     / display 'PW'     format = 5.2;
define RCARD  / display 'RCARD'  format = 5.2;
define RTRR   / display 'RTRR'   format = 5.2;

run;
ods html close;
ods listing;
```

Figure 2 shows the resulting output from the previous series of code.

V5

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		CCIUAC	CCIUTC	CONB	CONQ	EEC	EEGP	EXVUS	FM1	FM1D82	FSPCAP	FSPCOM	FSPCON	IP	LHUR	LUINC	PW	RCARD	RTRR
2	CCIUAC	1.00	1.00	0.80	0.97	0.47	-0.52	-0.42	0.97	0.93	0.92	0.95	0.92	0.98	-0.84	-0.62	0.81	0.22	0.97
3	CCIUTC	1.00	1.00	0.78	0.98	0.47	-0.50	-0.42	0.98	0.92	0.93	0.96	0.95	0.98	-0.80	-0.58	0.84	0.19	0.99
4	CONB	0.80	0.78	1.00	0.72	0.31	-0.51	-0.02	0.75	0.74	0.68	0.66	0.60	0.78	-0.69	-0.61	0.64	0.38	0.78
5	CONQ	0.97	0.98	0.72	1.00	0.47	-0.42	-0.49	0.96	0.88	0.92	0.96	0.96	0.96	-0.78	-0.51	0.84	0.13	0.96
6	EEC	0.47	0.47	0.31	0.47	1.00	-0.23	-0.39	0.43	0.41	0.41	0.43	0.42	0.49	-0.52	-0.31	0.36	0.06	0.43
7	EEGP	-0.52	-0.50	-0.51	-0.42	-0.23	1.00	0.18	-0.55	-0.74	-0.57	-0.47	-0.39	-0.46	0.49	0.62	-0.10	-0.40	-0.47
8	EXVUS	-0.42	-0.42	-0.02	-0.49	-0.39	0.18	1.00	-0.38	-0.41	-0.40	-0.45	-0.46	-0.39	0.50	0.06	-0.19	0.33	-0.33
9	FM1	0.97	0.98	0.75	0.96	0.43	-0.55	-0.38	1.00	0.95	0.95	0.97	0.95	0.96	-0.73	-0.60	0.84	0.19	0.99
10	FM1D82	0.93	0.92	0.74	0.88	0.41	-0.74	-0.41	0.95	1.00	0.93	0.90	0.85	0.90	-0.77	-0.71	0.65	0.29	0.91
11	FSPCAP	0.92	0.93	0.68	0.92	0.41	-0.57	-0.40	0.95	0.93	1.00	0.98	0.94	0.91	-0.73	-0.65	0.76	0.20	0.94
12	FSPCOM	0.95	0.96	0.66	0.96	0.43	-0.47	-0.45	0.97	0.90	0.98	1.00	0.99	0.93	-0.71	-0.54	0.84	0.11	0.96
13	FSPCON	0.92	0.95	0.60	0.96	0.42	-0.39	-0.46	0.95	0.85	0.94	0.99	1.00	0.91	-0.66	-0.45	0.86	0.05	0.95
14	IP	0.98	0.98	0.78	0.96	0.49	-0.46	-0.39	0.96	0.90	0.91	0.93	0.91	1.00	-0.86	-0.67	0.84	0.24	0.97
15	LHUR	-0.84	-0.80	-0.69	-0.78	-0.52	0.49	0.50	-0.73	-0.77	-0.73	-0.71	-0.66	-0.86	1.00	0.72	-0.50	-0.30	-0.73
16	LUINC	-0.62	-0.58	-0.61	-0.51	-0.31	0.62	0.06	-0.60	-0.71	-0.65	-0.54	-0.45	-0.67	0.72	1.00	-0.36	-0.56	-0.59
17	PW	0.81	0.84	0.64	0.84	0.36	-0.10	-0.19	0.84	0.65	0.76	0.84	0.86	0.84	-0.50	-0.36	1.00	0.01	0.89
18	RCARD	0.22	0.19	0.38	0.13	0.06	-0.40	0.33	0.19	0.29	0.20	0.11	0.05	0.24	-0.30	-0.56	0.01	1.00	0.22
19	RTRR	0.97	0.99	0.78	0.96	0.43	-0.47	-0.33	0.99	0.91	0.94	0.96	0.95	0.97	-0.73	-0.59	0.89	0.22	1.00
20																			

Figure 2: topic2.xls

THE OUTPUT IS EVEN BETTER, BUT…

The improvements in topic2.xls (Figure 2) are considerable. Everything fits into the window (with 18 variables!) and could be printed onto a single page. The matrix is symmetrical and balanced. The formats are consistent, and only the necessities are displayed. The overall formatting is clean and simple. The column headers match the row headers. But, there is still room for improvement:

- Fully-populated correlation matrices contain a mirror image of duplicates above the diagonal. It is not necessary to see these.
- With so many variables in the matrix, it's hard to spot-check correlations within threshold ranges. It would be helpful to have colorful conditional formatting for cells containing ranges that I specify.
- Using variable names for column and row headers made the formatting easier and cleaner…but now I don't know what the variables mean anymore. We need a legend that contains variable definitions using variable labels.

These issues are discussed in the next section.

VISUAL CUES IMPROVE INTERPRETABILITY OF A LARGE MATRIX

We will delete the duplicate correlations in a DATA step. While we're in the DATA step, we can also tidy up a couple of other things that will make the PROC REPORT code simpler: 1) restrict the data in the SET statement to only the correlation statistics, and 2) assign formats to the correlation statistics so that this does not need to be done in each of the DEFINE statements in PROC REPORT.

The DATA step code shown below deletes the duplicate correlation statistics. It establishes a counter flag that ranks the order of the observations, starting at one. The variables are arrayed (again, maintaining the original order is important!) and a DO loop counts through the observations and deletes all values where the DO variable is greater than or equal to the counter flag. This logic deletes the matrix diagonal and the values above it.

```
data citimon3;
set citimon2 (where=(_type_='CORR'));

halfmatflg+1;

array deletes(18) CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP
FSPCOM FSPCON IP LHUR LUINC PW RCARD RTRR;

do i = 1 to 18;
if halfmatflg =< i then deletes(i) = .;
end;

drop halfmatflg i;

format CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP FSPCOM FSPCON
IP LHUR LUINC PW RCARD RTRR 5.2;
run;
```

Create conditional formats based on the cell value by creating a temporary format. The format can then be applied in STYLE options in DEFINE statements in PROC REPORT. While CALL DEFINE is also a valid technique for establishing conditional formats, temporary formats are easy to create and readable by Excel.

```
proc format;
value corrfmt
low - -.9, .9 - high = 'red'
-.9< - -.8, .8 -< .9 = 'orange'
-.8< - -.7, .7 -< .8 = 'yellow';
run;
```

We now need a way to add a legend beneath the matrix that will show the variable names and labels and make the matrix more interpretable. Although it might seem that a PROC CONTENTS after the PROC REPORT or a PROC PRINT of the variable name and label variables from a PROC CONTENTS OUT= dataset would work, this would have an adverse effect on the matrix. Again, the column width would be dictated by these elements and

the first columns would become too wide and ruin the symmetry of the matrix. The solution: LINE statements in a COMPUTE AFTER block following the DEFINE statements in PROC REPORT. These statements will span text over the width of the entire matrix, but will not affect the width of the columns. In order to write the LINE statements, we'll need to get access to the variable labels by creating macro variables or calling them from a DICTIONARY table. Here, we're going to call them from DICTIONARY.COLUMNS in PROC SQL and write them into a macro variable that automatically generates a series of LINE statements. The SQL code below will generate the following LINE statement for each variable in the matrix; the macro functions %NRSTR are necessary to prevent an attempted macro call if a macro trigger is used in a variable label:

```
%NRSTR(%NRSTR(line "variable-name: variable-label");
```

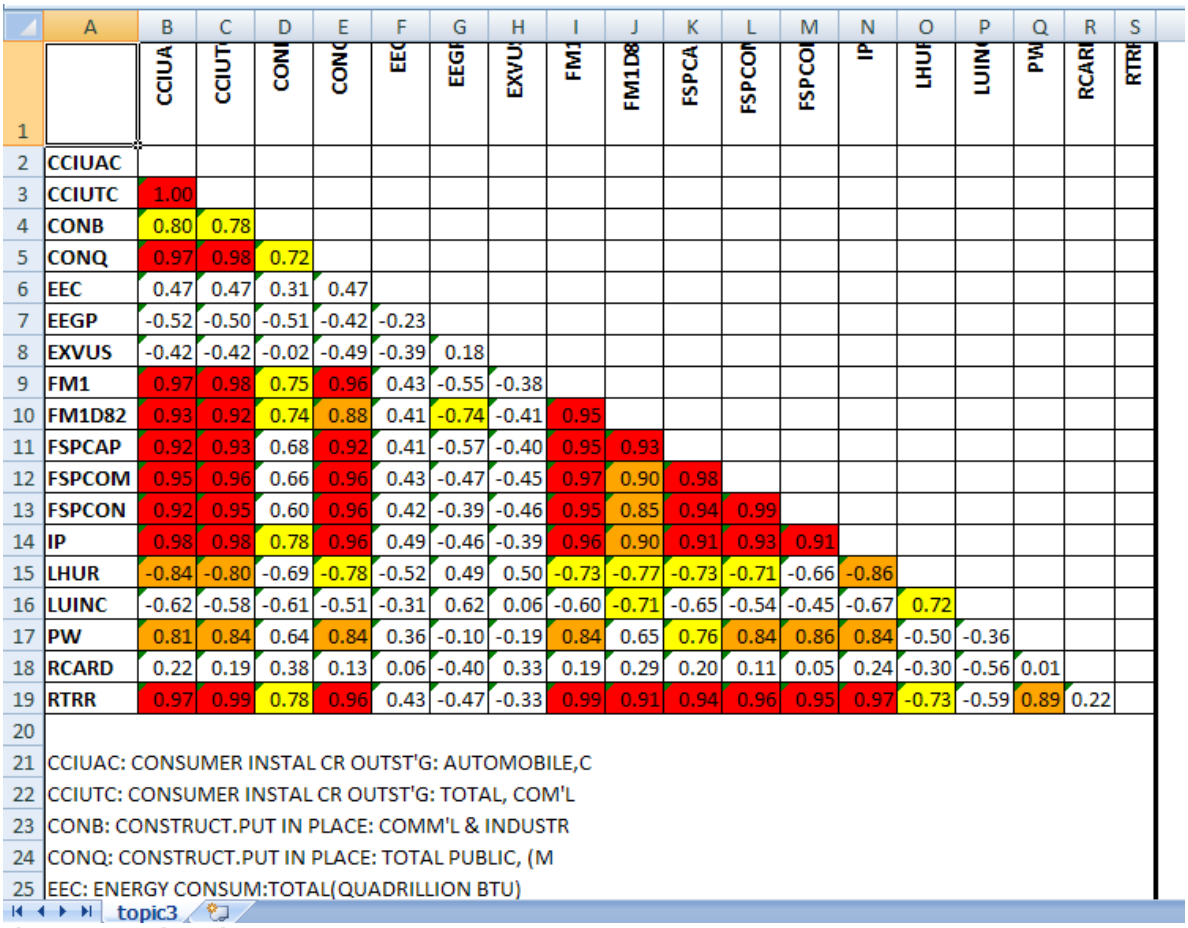
```
proc sql;
select '%NRSTR(%NRSTR(' || 'line "' || trim(name) || ': ' || trim(label) ||
'"));'
into :legend separated by ' '
from dictionary.columns
where libname='SASHELP' and memname='CITIMON'
and name in ("CCIUAC" "CCIUTC" "CONB" "CONQ" "EEC" "EEGP" "EXVUS" "FM1"
"FM1D82" "FSPCAP" "FSPCOM" "FSPCON" "IP" "LHUR" "LUINC" "PW" "RCARD" "RTRR");
quit;
```

Now we can use basically the same PROC REPORT code as in the previous iteration, eliminating the WHERE statement, eliminating the format options in each of the DEFINE statements, applying the STYLE= option in each of the DEFINE statements, and adding a COMPUTE block that references the macro variable that stores the LINE statements.

```
ods html file = "C:\SAS\NESUG\topic3.xls"
headtext="<style> td {mso-number-format:\@} </style>"  style=corrmatx;

proc report data = citimon3
style(header)={htmlstyle="mso-rotate:90; height:50pt"};

column _name_ CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP FSPCOM
FSPCON IP LHUR LUINC PW RCARD RTRR;
define _name_ / display ''  style=[font_weight=bold];
define CCIUAC / display 'CCIUAC' style=[background=corrfmt.];
define CCIUTC / display 'CCIUTC' style=[background=corrfmt.];
define CONB   / display 'CONB'   style=[background=corrfmt.];
define CONQ   / display 'CONQ'   style=[background=corrfmt.];
define EEC    / display 'EEC'    style=[background=corrfmt.];
define EEGP   / display 'EEGP'   style=[background=corrfmt.];
define EXVUS  / display 'EXVUS'  style=[background=corrfmt.];
define FM1    / display 'FM1'    style=[background=corrfmt.];
define FM1D82 / display 'FM1D82' style=[background=corrfmt.];
define FSPCAP / display 'FSPCAP' style=[background=corrfmt.];
define FSPCOM / display 'FSPCOM' style=[background=corrfmt.];
define FSPCON / display 'FSPCON' style=[background=corrfmt.];
define IP     / display 'IP'     style=[background=corrfmt.];
define LHUR   / display 'LHUR'   style=[background=corrfmt.];
define LUINC  / display 'LUINC'  style=[background=corrfmt.];
define PW     / display 'PW'     style=[background=corrfmt.];
define RCARD  / display 'RCARD'  style=[background=corrfmt.];
define RTRR   / display 'RTRR'   style=[background=corrfmt.];

compute after / style=[just=left];
  line '       ';
  &legend
endcomp;
run;
```

```
ods html close;
ods listing;
```

Figure 3 shows the results of this series of code.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1		CCIUA	CCIUT	CONI	CONC	EEC	EEGF	EXVU	FM1	FM1D8	FSPCA	FSPCOI	FSPCOI	IP	LHUF	LUINC	PW	RCARI	RTRR
2	CCIUAC																		
3	CCIUTC	1.00																	
4	CONB	0.80	0.78																
5	CONQ	0.97	0.98	0.72															
6	EEC	0.47	0.47	0.31	0.47														
7	EEGP	-0.52	-0.50	-0.51	-0.42	-0.23													
8	EXVUS	-0.42	-0.42	-0.02	-0.49	-0.39	0.18												
9	FM1	0.97	0.98	0.75	0.96	0.43	-0.55	-0.38											
10	FM1D82	0.93	0.92	0.74	0.88	0.41	-0.74	-0.41	0.95										
11	FSPCAP	0.92	0.93	0.68	0.92	0.41	-0.57	-0.40	0.95	0.93									
12	FSPCOM	0.95	0.96	0.66	0.96	0.43	-0.47	-0.45	0.97	0.90	0.98								
13	FSPCON	0.92	0.95	0.60	0.96	0.42	-0.39	-0.46	0.95	0.85	0.94	0.99							
14	IP	0.98	0.98	0.78	0.96	0.49	-0.46	-0.39	0.96	0.90	0.91	0.93	0.91						
15	LHUR	-0.84	-0.80	-0.69	-0.78	-0.52	0.49	0.50	-0.73	-0.77	-0.73	-0.71	-0.66	-0.86					
16	LUINC	-0.62	-0.58	-0.61	-0.51	-0.31	0.62	0.06	-0.60	-0.71	-0.65	-0.54	-0.45	-0.67	0.72				
17	PW	0.81	0.84	0.64	0.84	0.36	-0.10	-0.19	0.84	0.65	0.76	0.84	0.86	0.84	-0.50	-0.36			
18	RCARD	0.22	0.19	0.38	0.13	0.06	-0.40	0.33	0.19	0.29	0.20	0.11	0.05	0.24	-0.30	-0.56	0.01		
19	RTRR	0.97	0.99	0.78	0.96	0.43	-0.47	-0.33	0.99	0.91	0.94	0.96	0.95	0.97	-0.73	-0.59	0.89	0.22	
20																			
21	CCIUAC: CONSUMER INSTAL CR OUTST'G: AUTOMOBILE,C																		
22	CCIUTC: CONSUMER INSTAL CR OUTST'G: TOTAL, COM'L																		
23	CONB: CONSTRUCT.PUT IN PLACE: COMM'L & INDUSTR																		
24	CONQ: CONSTRUCT.PUT IN PLACE: TOTAL PUBLIC, (M																		
25	EEC: ENERGY CONSUM:TOTAL(QUADRILLION BTU)																		

topic3

Figure 3: topic3.xls

HAPPY DAYS ARE HERE AGAIN, BUT…

The output in topic3.xls (Figure 3) is nothing short of a visual masterpiece. The matrix is easy to read, and it's easy to visually identify the correlations by their strength. The only remaining problem: there's a lot of repetitive code crying out for a macro. Furthermore, for this to be a maximally useful tool in data analysis and exploration, it needs to be automated in such a way that it's quickly adaptable and easy to execute. The next section presents a macro that accomplishes that.

MACRO CODE TO GENERATE THIS MATRIX FOR ANY DATASET AND VARIABLE LIST

The macro presented below, CORRMATX, generates an HTML file readable by Excel that displays a Pearson's correlation matrix, formatted to display the maximum possible information on a single screen, with visual cues to highlight correlations of a specified strength.

CORRMATX parameters and definitions

libnamex	REQUIRED: The libref of the dataset to be used for the correlation matrix
dsname	REQUIRED: The name of the dataset to be used for the correlation matrix
vars	REQUIRED: The list of variables to be included in the matrix
savpath	REQUIRED: The path where the Excel file should be saved
savfile	Optional: The name of the Excel file (default is CORRMATX)
hight	Optional: The threshold at which correlations should be flagged by a red cell color. Must be a value greater than 0 and less than 1. Default value: .5

medt	Optional: The threshold at which correlations should be flagged by an orange cell color. Must be a value greater than 0 and less than hight. Default: nothing flagged as orange.
lowt	Optional: The threshold at which correlations should be flagged by a yellow cell color. Must be a value greater than 0 and less than medt. Should only be specified when both hight and medt were specified. Default: nothing flagged as yellow.

```
%macro corrmatx(libnamex=, dsname=, vars=, savpath=, savfile=CORRMATX, hight=.5,
medt=0, lowt=0);
options missing='';

%let cmvarn = %sysfunc(countw(&vars));
%local i;

proc corr data = &libnamex..&dsname outp=corrmtx nomiss;
var &vars;
run;

proc template;
define style corrmatx;
   style table / borderwidth=10;
end;
run;

* suppress titles and footnotes;
title; footnote;

data corrmtx2;
set corrmtx (where=(_type_='CORR'));
halfmatflg+1;
array deletes(&cmvarn) &vars;

do i = 1 to &cmvarn;
if halfmatflg =< i then deletes(i) = .;
end;

drop halfmatflg i;
format &vars 5.2;
run;

proc format;
value corrfmtx
low - -&hight, &hight - high = 'red'
%if &medt ne 0 %then %do;
-&hight< - -&medt, &medt -< &hight = 'orange' %end;
%if &lowt ne 0 %then %do;
-&medt< - -&lowt, &lowt -< &medt = 'yellow'; %end;
run;

proc sql;
select '%NRSTR(%NRSTR(' || 'line "' || trim(name) || ': ' || trim(label) || '"));'
into :legend separated by ' '
from dictionary.columns
where libname=%UPCASE("&libnamex") and memname=%UPCASE("&dsname")
and name in (
%do i = 1 %to &cmvarn;
"%scan(&vars, &i) "
%end;
);
quit;
```

```
ods html file = "&savpath\&savfile..xls"
headtext="<style> td {mso-number-format:\@} </style>"  style=corrmatx;

proc report data = corrmtx2
style(header)={htmlstyle="mso-rotate:90; height:50pt"};

column _name_ &vars;
define _name_ / display ''  style=[font_weight=bold];

%do i = 1 %to &cmvarn;
define %scan(&vars, &i) / display " %scan(&vars, &i)" style=[background=corrfmtx.];
%end;

compute after / style=[just=left];
line '            ';
&legend
endcomp;
run;

ods html close;
options missing='.'; /* restore default */

%mend corrmatx;


/* SAMPLE MACRO CALL THAT GENERATES THE SAME MATRIX FROM SASHELP.CITIMON */
%corrmatx(libnamex=SASHELP,
          dsname=CITIMON,
          vars=CCIUAC CCIUTC CONB CONQ EEC EEGP EXVUS FM1 FM1D82 FSPCAP FSPCOM
               FSPCON IP LHUR LUINC PW RCARD RTRR,
          savpath=C:\SAS\NESUG,
          savfile=topic4,
          hight=.9, medt=.8, lowt=.7)

/* SAMPLE MACRO CALL USING ANOTHER SASHELP DATASET */
%corrmatx(libnamex=sashelp,
          dsname=usecon,
          vars=AIRRPMD AIRRPMT CHEMICAL COAL DURABLES HS1FAM HSTOTAL NONDUR
               PETROL TOBACCO VEHICLES,
          savpath=C:\SAS\NESUG,
          savfile=top4ex2)
```

CONCLUSIONS

Highly specialized output readable by Excel is attainable with ODS HTML. Many programming tools can be utilized to control the final result, including OUT= options, DATA step alterations, formats (SAS and CSS), templates, and PROC REPORT. It is important to think about how SAS will write the output, and also about how Excel will read it.

The end result is especially worth the programming investment when:

- output interpretability is greatly enhanced, and,
- the output is generated often.

REFERENCES

Chapman, David. 2003. “Using Formats and Other Techniques to Complete PROC REPORT Tables.” Proceedings of the SUGI 28 Conference. http://www2.sas.com/proceedings/sugi28/132-28.pdf

DelGobbo, Vincent. 2003. "A Beginner's Guide to Incorporating SAS Output in Microsoft Office Applications." Proceedings of the SUGI 28 Conference. http://www2.sas.com/proceedings/sugi28/052-28.pdf

DelGobbo, Vincent. 2004. "Moving Data and Analytical Results between SAS and Microsoft Office." Proceedings of the SESUG 2004 Conference. http://analytics.ncsu.edu/sesug/2004/HW06-DelGobbo.pdf

Parker, Chevell. 2003. "Generating Custom Excel Spreadsheets using ODS." Proceedings of the PharmaSUG 2003 Conference. http://www.lexjansen.com/pharmasug/2003/sasinstitute/sas129.pdf

FURTHER READING

Croghan, Carry. 2007. "Reducing PROC CORR Output using ODS and Data Step." Proceedings of the 2007 SESUG Conference. http://analytics.ncsu.edu/sesug/2007/PO17.pdf

ACKNOWLEDGMENTS

The author thanks the following for their review and guidance: Chris Bost, Paulette Staum, and Victoria Deitch.

CONTACT INFORMATION

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

Aaron Hill
MDRC
16 East 34th Street, 19th Floor
New York, NY 10016
Work Phone: (212) 340-8898
Email: aaron.hill@mdrc.org, aaron.hill@me.com