QUICK TIPS AND TRICKS IN SAS PROGRAMMING

Khoi Dinh To
Office of Planning and Decision Support, Virginia Commonwealth University

ABSTRACT

In this presentation, the following quick tips and tricks are discussed:

1. Masking cell contents to hide sensitive data in procedure tabulate
2. Moving macro variables and their values from server domain to local domain and vice versa in SAS EG (Enterprise Guide)
3. Combining procedure tabulate and lag(n) function for rates of change
4. Deleting unnecessary total rows (in groups that have only one category) in procedure tabulate and procedure report
5. Building, storing, and calling a user-defined function
6. Saving SAS logs to a permanent document for future reference

KEYWORDS

masking, macro variables, lag(n) function, rate of change, delete total rows, SAS log

SYNTAX AND EXPLANATION

1. Masking cell contents

In the following table, cells having less than 10 observations were masked with (*) by using “format within format”. Also note that we can apply multiple formats to the data. For example, cells having less than 10 observations were masked with (*) and colored in yellow.

	Fall 2008		Fall 2009		Fall 2010		Fall 2011		Fall 2012	
	DFW	Other	DFW	Other	DFW	Other	DFW	Other	DFW	Other
MATH200	179	401	211	474	299	529	299	533	289	512
MATH201	117	168	79	203	135	206	128	216	155	235
MATH211	43	70	35	79	35	78	23	79	37	123
UNIV111	301	2,598	249	2,333	244	2,288	261	2,248	224	1,967
UNIV112	104	492	80	601	96	680	87	886	99	962
UNIV200	.	.	.	.	310	1,172	275	1,438	241	1,878
SOCY101	219	873	165	886	181	893	94	577	107	890
STAT210	303	837	253	849	257	859	254	871	248	931

ACCT202	80	100	66	170	114	211	132	203	145	205
BIOZ151	92	397	108	430	39	317	81	420	70	455
CHEM101	371	612	353	709	336	721	413	714	390	832
ENGL200	448	1,103	392	1,198	*	25	.	.	.	.
THEA311	*	40	.	39	*	24	.	28	.	28
TOTAL	2,258	7,691	1,991	7,971	2,056	8,003	2,047	8,213	2,005	9,018

```
proc format;
value less_than_ten_b (notsorted)
0 -< 10 = 'Yellow'
OTHER   = 'White'
;
value less_than_ten_c (notsorted)
0 -< 10 = '*'
OTHER   = [comma15.]
; run;

title 'Masking cell contents';
footnote1 justify=center '(*) Cells having fewer than 10 students';
proc tabulate data=ol missing;
format academic_period $term. final_grade $dfw.;
class course_identification / order=data;
class academic_period final_grade / order=data preloadfmt;
table course_identification='' all='TOTAL',
academic_period=''*final_grade=''*n=''*format=less_than_ten_c.*{style={backgr
ound=less_than_ten_b.}};
run;
```

2. Moving macro variables and their values from server domain to local domain and vice versa in SAS EG (Enterprise Guide)

For SAS Enterprise Guide (EG) users, sometimes macro variables and their values need to be brought over to the local workspace from the server, especially when multiple data sets or outputs need to be written to separate files in a local drive. Manually retyping the macro variables and their values in the local workspace after they have been created in the server workspace would be time-consuming and error-prone, especially when we have quite a number of macro variables and values to bring over.

Instead, this task can be achieved in an efficient manner by using dictionary tables and call symput routine, as illustrated in more detail below. The same approach can also be used to bring macro variables and their values from the local to the server workspace.

Suppose that we have the following user-defined macro variables in the server workplace:

```
GLOBAL _SASSERVERNAME   'SASApp' (This is the server workspace.)
GLOBAL COLL_1           AH
GLOBAL COLL _10         IN
GLOBAL COLL _11         LF
GLOBAL COLL _12         MD
GLOBAL COLL _13         NR
GLOBAL COLL _14         PH
GLOBAL COLL _15         SW
GLOBAL COLL _16         VR
GLOBAL COLL _17         WS
GLOBAL COLL _2          AR
GLOBAL COLL _3          BE
GLOBAL COLL _4          BU
GLOBAL COLL _5          DN
GLOBAL COLL _6          ED
GLOBAL COLL _7          EG
GLOBAL COLL _8          GR
GLOBAL COLL _9          HS
GLOBAL DEPTS            ANAT BIOC BUSS CHSC EDUC GVPA MASC MUSC NURS SLWK
WRLD
GLOBAL FIRST_FALL       200910
GLOBAL FIRST_SUMMER     200830
GLOBAL LAST_FALL        201510
GLOBAL LAST_SUMMER      201430
GLOBAL MAJORS           BIO,MATH,PHYS,COMM,CHEM,HIST,GEO,ENG,FIN,INS
GLOBAL N_COLL           17
GLOBAL N_DEPT           11
```

What we would like to do now is bring these macro variables and their values to the local workspace. In order to achieve this, we need to obtain a SAS macro dictionary table where all macro variables and their values are stored.

```
proc sql;
create table macro_list_all as select distinct * from dictionary.macros
; quit;
```

Below is what our SAS macro dictionary table looks like. It contains all macro variables and values that have been created by the system or the current user during a session. The ones highlighted in yellow are those we would want to bring to the local workspace.

scope	name	offset	value
AUTOMATIC	SYSDATE	0	15JUN15
AUTOMATIC	SYSDATE9	0	15JUN2015
AUTOMATIC	SYSDAY	0	Monday
AUTOMATIC	SYSSCP	0	LIN X64
AUTOMATIC	SYSSCPL	0	Linux
AUTOMATIC	SYSSITE	0	0070001453
AUTOMATIC	SYSVLONG	0	9.02.02M3P041310
AUTOMATIC	SYSVLONG4	0	9.02.02M3P04132010
GLOBAL	COLL_1	0	AH
GLOBAL	COLL _10	0	IN
GLOBAL	COLL _11	0	LF
GLOBAL	COLL _12	0	MD
GLOBAL	COLL _13	0	NR
GLOBAL	COLL _14	0	PH
GLOBAL	COLL _15	0	SW
GLOBAL	COLL _16	0	VR
GLOBAL	COLL _17	0	WS
GLOBAL	COLL _2	0	AR
GLOBAL	COLL _3	0	BE
GLOBAL	COLL _4	0	BU
GLOBAL	COLL _5	0	DN
GLOBAL	COLL _6	0	ED
GLOBAL	COLL _7	0	EG
GLOBAL	COLL _8	0	GR
GLOBAL	COLL _9	0	HS
GLOBAL	DEPTS	0	ANAT BIOC BUSS CHSC EDUC GVPA MASC MUSC NURS SLWK WRLD
GLOBAL	FIRST_FALL	0	200910
GLOBAL	FIRST_SUMMER	0	200830
GLOBAL	LAST_FALL	0	201510
GLOBAL	LAST_SUMMER	0	201430
GLOBAL	MAJORS	0	BIO,MATH,PHYS,COMM,CHEM,HIST,GEO,ENG,FIN,INS
GLOBAL	N_COLL	0	17
GLOBAL	N_DEPT	0	11
GLOBAL	SQLEXITCODE	0	0
GLOBAL	SQLOBS	0	0
GLOBAL	SQLOOPS	0	0
GLOBAL	SQLRC	0	0
GLOBAL	SQLXOBS	0	0
GLOBAL	SYS_SQL_IP_ALL	0	-1
GLOBAL	SYS_SQL_IP_STMT	0	
GLOBAL	_CLIENTAPP	0	SAS Enterprise Guide'

GLOBAL	_CLIENTAPPABREV	0	EG
GLOBAL	_CLIENTMACHINE	0	PRO-GIN-KDT-WD1'
GLOBAL	_CLIENTPROJECTNAME	0	bringing macro variables from server to local.egp'
GLOBAL	_CLIENTTASKLABEL	0	01. server'
GLOBAL	_CLIENTUSERID	0	kdto'
GLOBAL	_CLIENTUSERNAME	0	Khoi D To'
GLOBAL	_CLIENTVERSION	0	7.100.0.2002'
GLOBAL	_EG_WORKSPACEINIT	0	1
GLOBAL	_SASHOSTNAME	0	sasbicomp.vcu.edu'
GLOBAL	_SASPROGRAMFILE	0	
GLOBAL	_SASSERVERNAME	0	SASApp'

We trim the macro dictionary table a bit and just keep macro variables and values that we have created and would like to bring to the local workspace. These user-defined macro variables have global scope and their names do not start with 'SYS', 'SQL', 'SAS', or '_'.

```
data macro_list; set macro_list_all;
where scope = 'GLOBAL'
  and (substr(name,1,3) not in ('SYS','SQL','SAS')
  and substr(name,1,1) not in ('_'));
run;
```

scope	name	offset	value
GLOBAL	COLL_1	0	AH
GLOBAL	COLL_10	0	IN
GLOBAL	COLL_11	0	LF
GLOBAL	COLL_12	0	MD
GLOBAL	COLL_13	0	NR
GLOBAL	COLL_14	0	PH
GLOBAL	COLL_15	0	SW
GLOBAL	COLL_16	0	VR
GLOBAL	COLL_17	0	WS
GLOBAL	COLL_2	0	AR
GLOBAL	COLL_3	0	BE
GLOBAL	COLL_4	0	BU
GLOBAL	COLL_5	0	DN
GLOBAL	COLL_6	0	ED
GLOBAL	COLL_7	0	EG
GLOBAL	COLL_8	0	GR
GLOBAL	COLL_9	0	HS
GLOBAL	DEPTS	0	ANAT BIOC BUSS CHSC EDUC GVPA MASC MUSC NURS SLWK WRLD
GLOBAL	FIRST_FALL	0	200910

GLOBAL	FIRST_SUMMER	0	200830
GLOBAL	LAST_FALL	0	201510
GLOBAL	LAST_SUMMER	0	201430
GLOBAL	MAJORS	0	BIO,MATH,PHYS,COMM,CHEM,HIST,GEO,ENG,FIN,INS
GLOBAL	N_COLL	0	17
GLOBAL	N_DEPT	0	11

The next step is to bring this “macro_list” data set to the local workspace and run the following data step to recreate the macro variables and their values:

```
data _null_; set macro_list;
call symput(name, value);
run;

%put _user_;
```

The log shows the following results:

```
GLOBAL _SASSERVERNAME   'Local' (This is the local workspace.)
GLOBAL COLL_1           AH
GLOBAL COLL _10         IN
GLOBAL COLL _11         LF
GLOBAL COLL _12         MD
GLOBAL COLL _13         NR
GLOBAL COLL _14         PH
GLOBAL COLL _15         SW
GLOBAL COLL _16         VR
GLOBAL COLL _17         WS
GLOBAL COLL _2          AR
GLOBAL COLL _3          BE
GLOBAL COLL _4          BU
GLOBAL COLL _5          DN
GLOBAL COLL _6          ED
GLOBAL COLL _7          EG
GLOBAL COLL _8          GR
GLOBAL COLL _9          HS
GLOBAL DEPTS            ANAT BIOC BUSS CHSC EDUC GVPA MASC MUSC NURS SLWK
WRLD
GLOBAL FIRST_FALL       200910
GLOBAL FIRST_SUMMER     200830
GLOBAL LAST_FALL        201510
GLOBAL LAST_SUMMER      201430
GLOBAL MAJORS           BIO,MATH,PHYS,COMM,CHEM,HIST,GEO,ENG,FIN,INS
GLOBAL N_COLL           17
GLOBAL N_DEPT           11
```

3. Combining procedure tabulate and lag(n) function for rates of change

For SAS users, procedure tabulate and procedure report (and its compute blocks) are probably among the most common procedures that are used to calculate and display data. It is, however, pretty difficult to calculate and display changes from one column to another using data from other rows by just using these two procedures. Compute blocks in procedure report are capable of calculating additional columns, but it would also be challenging to pick up values from other rows as inputs.

This presentation shows how procedure tabulate can work with the lag(n) function to calculate rates of change from one period of time to another as it offers the flexibility of feeding into calculations the data retrieved from other rows of the report. Procedure report is then used to produce the desired output. The same approach can also be used in a variety of scenarios to produce customized reports.

Student Enrollment by Major

MCV-Campus Schools												
	Fall 2009			Fall 2010			Fall 2011			Fall 2012		
	In State	Out State	Total	In State	Out State	Total	In State	Out State	Total	In State	Out State	Total
Allied Health Professions												
Baccalaureate	165	12	177	176	10	186	176	8	184	183	12	195
Masters	390	154	544	391	167	558	385	163	548	388	141	529
Doctoral	249	90	339	234	95	329	213	78	291	209	73	282
Grad. Post-Bacc Certs	10	2	12	5	2	7	9	.	9	6	2	8
Grad. Post-Masters Certs	20	4	24	18	2	20	25	3	28	23	7	30
All Degrees and Certs	**834**	**262**	**1,096**	**824**	**276**	**1,100**	**808**	**252**	**1,060**	**809**	**235**	**1,044**
Headcount Growth	.	.	.	-10	14	4	-16	-24	-40	1	-17	-16
% Growth over prior Fall	.	.	.	-1.2	5.3	0.4	-1.9	-8.7	-3.6	0.1	-6.7	-1.5
School of Dentistry												
Baccalaureate	57	3	60	64	2	66	59	3	62	55	1	56
Masters	10	22	32	11	24	35	6	27	33	7	26	33
First Professional	245	152	397	255	153	408	268	153	421	269	147	416
All Degrees and Certs	**312**	**177**	**489**	**330**	**179**	**509**	**333**	**183**	**516**	**331**	**174**	**505**
Headcount Growth	.	.	.	18	2	20	3	4	7	-2	-9	-11
% Growth over prior Fall	.	.	.	5.8	1.1	4.1	0.9	2.2	1.4	-0.6	-4.9	-2.1
……. (More schools/colleges here) …….												
ALL DEGREE PROGRAMS												
Baccalaureate	845	45	890	886	40	926	856	36	892	763	35	798

Masters	735	284	1,019	720	302	1,022	711	299	1,010	726	266	992
First Professional	1,111	547	1,658	1,130	548	1,678	1,170	534	1,704	1,198	524	1,722
Doctoral	411	325	736	394	338	732	355	321	676	348	311	659
Grad. Post-Bacc Certs	90	48	138	63	35	98	63	25	88	70	31	101
Grad. Post-Masters Certs	29	4	33	25	2	27	40	4	44	40	8	48
All Degrees and Certs	**3,221**	**1,253**	**4,474**	**3,218**	**1,265**	**4,483**	**3,195**	**1,219**	**4,414**	**3,145**	**1,175**	**4,320**
Headcount Growth	.	.	.	-3	12	9	-23	-46	-69	-50	-44	-94
% Growth over prior Fall	.	.	.	-0.1	1	0.2	-0.7	-3.6	-1.5	-1.6	-3.6	-2.1

Step 1: Using proc tabulate to output aggregate data into data sets

```
* output individual rows;

proc tabulate data=bio2 missing out=bio3;
class college1 cat academic_period residency;
table college1*cat all*cat, academic_period*(residency all);
run;

data bio3; set bio3;
if college1  = ' ' then college1  = 'UZ'; /* university total rows */
if residency = ' ' then residency = 'Z';  /* residency total columns */
run;

* output total rows;
* (this data set is used to calculate rates of change);

proc tabulate data=bio2 missing out=bio_total;
class college1 /* cat */ academic_period residency;
table college1 all, academic_period*(residency all);
run;

data bio_total; set bio_total;
if college1  = ' ' then college1  = 'UZ'; /* university total rows */
if residency = ' ' then residency = 'Z';  /* residency total columns */
run;
```

Below is part of the data set “bio_total” we get from proc tabulate above.

COLLEGE1	ACADEMIC_PERIOD	RESIDENCY	_TYPE_	_PAGE_	_TABLE_	N
AH	201010	N	111	1	1	262
AH	201010	R	111	1	1	834
AH	201010	Z	110	1	1	1096
AH	201110	N	111	1	1	276
AH	201110	R	111	1	1	824
AH	201110	Z	110	1	1	1100
AH	201210	N	111	1	1	252
AH	201210	R	111	1	1	808
AH	201210	Z	110	1	1	1060

AH	201310	N	111	1	1	235
AH	201310	R	111	1	1	809
AH	201310	Z	110	1	1	1044
DN	201010	N	111	1	1	177
DN	201010	R	111	1	1	312
DN	201010	Z	110	1	1	489
DN	201110	N	111	1	1	179
DN	201110	R	111	1	1	330
DN	201110	Z	110	1	1	509
DN	201210	N	111	1	1	183
DN	201210	R	111	1	1	333
DN	201210	Z	110	1	1	516
DN	201310	N	111	1	1	174
DN	201310	R	111	1	1	331
DN	201310	Z	110	1	1	505

Step 2: Using the lag(n) function to set up the data so that year-on-year changes (in numbers and percentages) can be calculated

```
* calculate total row number changes and total row percentage changes;

proc sort data=bio_total; by college1 academic_period residency; run;

data bio_total_b; set bio_total;
by college1;
count + 1;
if first.college1 then count = 1;

n_3 = lag3(n);
change = n - n_3;
change_pct = (change/n_3)*100;
```

Each school/college is treated as one group.

Using lag(3) function to set up the data as we have 3 rows for each school/college in a year: In-state, Out-state, and Total. Then year-on-year rates of change are calculated.

```
if count in (1,2,3) then do;
   n_3 = .;
   change = .;
   change_pct = .;
end;
run;
```

The rates of change for first three rows of each school/college are deleted as they are the first year of the time series.

Below is part of the data set “bio_total_b” after the rates of change are calculated.

COLLEGE1	ACADEMIC_PERIOD	RESIDENCY	N	count	n_3	change	change_pct
AH	201010	N	262	1			
AH	201010	R	834	2			
AH	201010	Z	1096	3			
AH	201110	N	276	4	262	14	5.3435
AH	201110	R	824	5	834	-10	-1.1990
AH	201110	Z	1100	6	1096	4	0.3650
AH	201210	N	252	7	276	-24	-8.6957
AH	201210	R	808	8	824	-16	-1.9417
AH	201210	Z	1060	9	1100	-40	-3.6364
AH	201310	N	235	10	252	-17	-6.7460
AH	201310	R	809	11	808	1	0.1238
AH	201310	Z	1044	12	1060	-16	-1.5094
DN	201010	N	177	1			
DN	201010	R	312	2			
DN	201010	Z	489	3			
DN	201110	N	179	4	177	2	1.1299
DN	201110	R	330	5	312	18	5.7692
DN	201110	Z	509	6	489	20	4.0900
DN	201210	N	183	7	179	4	2.2346
DN	201210	R	333	8	330	3	0.9091
DN	201210	Z	516	9	509	7	1.3752
DN	201310	N	174	10	183	-9	-4.9180
DN	201310	R	331	11	333	-2	-0.6006
DN	201310	Z	505	12	516	-11	-2.1318

Step 3: Using proc report to produce a report in the desired format

```
* some extra steps before stacking data sets together;

data bio_total; set bio_total;
cat = 98; /* total */
run;
```

```
data change_num; set bio_total_b(drop=count n n_3 change_pct);
rename change = n;
cat = 99; /* total number changes */
run;

data change_pct; set bio_total_b(drop=count n n_3 change);
rename change_pct = n;
cat = 100; /* total percentage changes */
run;

* stack all data sets together;

data all; set bio3 bio_total change_num change_pct; run;

* use proc report to produce the final report;

title1 'Student Enrollment by Major';
title2 '(Census 2)';
proc report data=all missing;
format college1 $college. cat cat. academic_period $term.
       residency $residency.;
column ('MCV-Campus Schools' college1 cat academic_period, residency, n);
define college1        / group  '' order=data preloadfmt noprint;
define cat             / group  '' order=data preloadfmt;
define academic_period / across '' order=data preloadfmt;
define residency       / across '' order=data preloadfmt;
define n               / analysis sum '' f=comma15.;

compute before college1 /style={just=l background=lightblue
font_weight=bold};
line college1 $college.;
endcomp;

compute cat;
if cat=98  then call define(_row_,'style','style={font_weight=bold}');
if cat=99  then call define(_col_,'style','style={font_style =italic}');
if cat=100 then call define(_col_,'style','style={font_style =italic}');
endcomp;

compute n;
if cat=100 then call define(_col_,'format','comma10.1');
endcomp;

compute after college1;
line ' ';
endcomp;
run;
```

Define font style and weight for the three rows: total, number changes, and percentage changes

Define the number format for the percentage changes

4. Deleting unnecessary total rows (in groups that have only one category) in procedure tabulate and procedure report

See the original table below. Total rows in yellow may need to be deleted as they are redundant.

		Fall 2008		Fall 2009		Fall 2010	
		Res	Non	Res	Non	Res	Non
Accounting	Accountancy	34	14	37	9	44	12
	Accounting	375	18	444	19	474	22
	Taxation	14	1	13	1	20	2
	TOTAL	423	33	494	29	538	36
Arts-Dean's Office	Cinema	0	0	0	0	0	0
	Film	70	6	71	10	76	10
	Fine Arts	9	49	7	52	5	51
	Interdisciplinary Studies	58	1	63	1	67	2
	TOTAL	137	56	141	63	148	63
Biochemistry	Biochemistry	52	35	25	28	23	24
	TOTAL	52	35	25	28	23	24
Biology	Biology	1,384	172	1,501	177	1,654	162
	TOTAL	1,384	172	1,501	177	1,654	162

Below is the revised table after deleting redundant total rows.

		Fall 2008		Fall 2009		Fall 2010	
		Res	Non	Res	Non	Res	Non
Accounting	Accountancy	34	14	37	9	44	12
	Accounting	375	18	444	19	474	22
	Taxation	14	1	13	1	20	2
	TOTAL	423	33	494	29	538	36
Arts-Dean's Office	Cinema	0	0	0	0	0	0
	Film	70	6	71	10	76	10
	Fine Arts	9	49	7	52	5	51
	Interdisciplinary Studies	58	1	63	1	67	2
	TOTAL	137	56	141	63	148	63
Biochemistry	Biochemistry	52	35	25	28	23	24
Biology	Biology	1,384	172	1,501	177	1,654	162

```
title 'Proc tabulate - Deleting total rows in groups having only one category
(original)';
footnote;
proc tabulate data=example1 missing out=example2;
format academic_period $term_short. major1 $major. residency $residency.
department1 $dept. major1 $maj.;
class department1 major1 / order=data preloadfmt;
class academic_period residency / order=data preloadfmt;
table department1=''*(major1='' all='TOTAL'),
academic_period=''*residency=''*n=''*f=comma15.;
run;

***********;

data example2; set example2;
if major1 = ' ' then major1 = 'ZZZ'; /* total lines */
run;

proc sort data=example2; by department1 academic_period residency major1;
run;

data example2; set example2;
by department1 academic_period residency /* major1 */;
count + 1;
if first.residency then count = 1;
run;

data example3; set example2;
if major1 = 'ZZZ' and count = 2 then delete;
run;

title 'Proc tabulate - Deleting total rows in groups having only one
category';
proc tabulate data=example3 missing;
format academic_period $term_short. major1 $major. residency $residency.
department1 $dept. major1 $maj.;
class department1 major1 / order=data preloadfmt;
class academic_period residency / order=data preloadfmt;
var n;
table department1=''*major1='',
academic_period=''*residency=''*n=''*sum=''*f=comma15.;
run;

*****************************;

title 'Proc report - Deleting total rows in groups having only one category
(original)';
proc report data=example1 missing out=example2b;
format academic_period $term_short. major1 $major. residency $residency.
department1 $dept. major1 $maj.;
column department1 major1 academic_period,residency,n;
define department1 / ' ' group order=data preloadfmt
style(column)={font_weight=bold};
define major1 / ' ' group order=data preloadfmt
style(column)={font_weight=bold};
define academic_period / ' ' across order=data preloadfmt;
```

```
define residency / ' ' across order=data preloadfmt
style(column)={just=right};
define n / ' ' f=comma15.;
break after department1 / summarize;
run;

***********;

data example2b; set example2b;
if major1 = ' ' then major1 = 'ZZZ'; /* total lines */
run;

data example2b; set example2b;
by department1;
count + 1;
if first.department1 then count = 1;
run;

data example3b; set example2b;
if major1 = 'ZZZ' and count = 2 then delete;
run;

title 'Proc report - Deleting total rows in groups having only one category';
proc report data=example3b missing;
format academic_period $term_short. major1 $major. residency $residency.
department1 $dept. major1 $maj.;
column department1 major1
('Fall 2008' _c3_ _c4_)   ('Fall 2009' _c5_ _c6_)   ('Fall 2010' _c7_ _c8_)
('Fall 2011' _c9_ _c10_)
('Fall 2012' _c11_ _c12_) ('Fall 2013' _c13_ _c14_) ('Fall 2014' _c15_
_c16_);
define department1 / ' ' group order=data preloadfmt style(column)={font_weight=bold};
define major1 / ' ' group order=data preloadfmt style(column)={font_weight=bold};
define _c3_  / 'Res' analysis sum f=comma15.; define _c4_  / 'Non' analysis sum f=comma15.;
define _c5_  / 'Res' analysis sum f=comma15.; define _c6_  / 'Non' analysis sum f=comma15.;
define _c7_  / 'Res' analysis sum f=comma15.; define _c8_  / 'Non' analysis sum f=comma15.;
define _c9_  / 'Res' analysis sum f=comma15.; define _c10_ / 'Non' analysis sum f=comma15.;
define _c11_ / 'Res' analysis sum f=comma15.; define _c12_ / 'Non' analysis sum f=comma15.;
define _c13_ / 'Res' analysis sum f=comma15.; define _c14_ / 'Non' analysis sum f=comma15.;
define _c15_ / 'Res' analysis sum f=comma15.; define _c16_ / 'Non' analysis sum f=comma15.;
run;
```

5. User-defined functions

If we have the same code lines in several SAS runs, whenever we want to modify them to incorporate changes, we would have to go to all the relevant SAS runs to update the same code lines accordingly. This task of keeping all SAS runs up-to-date can be time-consuming, especially when we have many SAS runs scattering in the network drive. I way to overcome this problem is to use user-defined functions. These functions are saved in a central location and can be called in a data step or proc SQL. When we want to modify a function, we would have to do it once. Changes made to a function would take effect when we call it in a data step or proc SQL. See the examples below.

Hard coding:

```
data rb_b; set rb;
length visaschev $3.;
if schev_citizen = 1 then visaschev = '111'; /* US national */

if schev_citizen = 2 then visaschev = '222'; /* US permanent resident */

if schev_citizen = 3 then do;
        if visa_type in
('A1','A2','A3','B1','B2','DA','E1','E2','F1','F2','G1','G2','G4',
                          'H4','I'
,'J1','J2','K1','K2','K3','L2','M1','O1','P1','R1','R2',
                          'TD','TN','U1','V2') then visaschev=visa_type;
   else if visa_type='HB' then visaschev='H2B';
   else if visa_type='LA' then visaschev='L1A';
   else if visa_type='LB' then visaschev='L1B';
   else if visa_type='N'  then visaschev='N1';
   else if visa_type='N6' then visaschev='NA6';
   else if visa_type='O3' then visaschev='O31';
   else if visa_type='RF' then visaschev='REF';
   else if visa_type='TP' then visaschev='TPS';
   else                        visaschev='333'; /* non-resident alien */
end;
run;

title 'Visa Types produced by hard coding';
proc tabulate data=rb_b missing;
class schev_citizen visaschev;
table schev_citizen='SCHEV Citizen'*visaschev='Visa Type' all='TOTAL',
n=''*f=comma15.;
run;
```

Building a user-defined function based on the hard coding above:

```
proc fcmp outlib=work.khoi.functions;
FUNCTION visa_function(citizen_var,visa_var $) $3;
if citizen_var = 1 then visa_function = '111'; /* US national */

if citizen_var = 2 then visa_function = '222'; /* US permanent resident */

if citizen_var = 3 then do;
        if visa_var in
('A1','A2','A3','B1','B2','DA','E1','E2','F1','F2','G1','G2','G4',
                          'H4','I'
,'J1','J2','K1','K2','K3','L2','M1','O1','P1','R1','R2',
                          'TD','TN','U1','V2') then visa_function=visa_var;
   else if visa_var='HB' then visa_function='H2B';
   else if visa_var='LA' then visa_function='L1A';
   else if visa_var='LB' then visa_function='L1B';
   else if visa_var='N'  then visa_function='N1';
   else if visa_var='N6' then visa_function='NA6';
   else if visa_var='O3' then visa_function='O31';
   else if visa_var='RF' then visa_function='REF';
   else if visa_var='TP' then visa_function='TPS';
   else                       visa_function='333'; /* non-resident alien */
```

```
end;
return(visa_function);
ENDSUB;
run;
```

Call the user-defined function in a data step:

```
options cmplib=work.khoi;

data rb_c; set rb;
visa = visa_function(schev_citizen,visa_type);
run;

title 'Visa Types produced by a user-defined function in data step';
proc tabulate data=rb_c missing;
class schev_citizen visa;
table schev_citizen='SCHEV Citizen'*visa='Visa Type' all='TOTAL',
n=''*f=comma15.;
run;
```

Call the user-defined function in a proc SQL:

```
proc sql;
create table rb_d as
select distinct academic_period, person_uid, schev_citizen, visa_type
             , visa_function(schev_citizen,visa_type) as visa
          from rb
; quit;

title 'Visa Types produced by a user-defined function in proc SQL';
proc tabulate data=rb_d missing;
class schev_citizen visa;
table schev_citizen='SCHEV Citizen'*visa='Visa Type' all='TOTAL',
n=''*f=comma15.;
run;
```

6. Saving SAS logs to a permanent document for future reference

This can be achieved easily by using proc printto as in the following example.

```
proc printto
log="I:\2 Enrollment Management\Test\Khoi_test_&SYSDATE9..log"
/* this is for the log */
/*print="I:\2 Enrollment Management\Test\Khoi_test_&SYSDATE9..lst" */
/* this is for the output */
new
/* if "new" were commented out, the logs/outputs from different sessions
would keep appending */
; run;

**********************;
```

```
proc sql; create table macro_list_all as select distinct * from
dictionary.macros; quit;

%let user = %sysfunc(translate(&_clientusername.,' ',"'"));
%let time = %sysfunc(time(),time.);
%let path = %sysfunc(translate(&_clientprojectpath.,' ',"'"));
%let tab  = %sysfunc(translate(&_clienttasklabel.,' ',"'"));

title 'New first-time degree-seeking undergraduates';
footnote1 "Produced by &USER. on &SYSDAY., &SYSDATE9. at &TIME.";
footnote2 "(Project path: &PATH.\&TAB.)";
proc tabulate data=test missing;
format academic_period $term_short. student_classification1 $class.;
class academic_period student_classification1 / order=data preloadfmt;
class  / order=data preloadfmt;
table student_classification1='' all='TOTAL',
academic_period=''*n=''*format=comma15.;
run;

**********************;

* save SAS log after each run;

proc printto; run;
/* return to default output destination */
```

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Please contact me at:

Khoi Dinh To, PhD
Office of Planning and Decision Support, Virginia Commonwealth University
901 W Franklin Street
Richmond, VA 23284-2527
Email:todinhkhoi@yahoo.com, kdto@vcu.edu