Paper 027-2013

Census Retires PROC COMPUTAB

Christopher J. Boniface, U.S. Bureau of the Census

Hung Pham, U.S. Bureau of the Census
Nora Szeto, U.S. Bureau of the Census

ABSTRACT

PROC COMPUTAB is used to generate tabular reports in a spreadsheet like format. PROC COMPUTAB has been around a long time. It has served us well at Census, but it is time to replace it with reporting procedures that are more modern. This paper will show how we converted PROC COMPUTAB to PROC TABULATE and PROC REPORT to create complex Census tables. We will also discuss how we created hundreds of tabular Excel tables using ODS TAGSETS EXCELXP. Furthermore, we will show how we used PROC TABULATE as a computing engine to handle overlapping format ranges and PROC REPORT as the reporting tool to control the appearance of the tabular Excel tables including column widths, row heights and formats.

INTRODUCTION

Every year the Census Bureau reports data on income, poverty, and health insurance coverage in the United States based on information collected in the Current Population Survey Annual Social and Economic Supplements (CPS ASEC) conducted by the U.S. Census Bureau. We are in the Survey Processing Branch within the Social, Economic and Housing Statistics Division (SEHSD) at the Census Bureau responsible for the programming and producing hundreds of predefined complex crosstab tables using SAS®.

We started using SAS® to generate crosstab reports in 1995. We produce custom tabulations using data extracts from micro-data files, internal analysis files and other data sources, to meet the specifications of subject matter analysts in SEHSD and of external data users. The custom tabulations range from simple tallies to complex crosstab of multiple variables and universes with numerous iterations, with input data ranging from the current year back to 1968 CPS ASEC data. At that time, we found that PROC COMPUTAB was the only procedure suitable for producing tabular reports. PROC TABULATE was the alternative but the syntax was confusing and difficult to follow. In addition, PROC TABULATE could not handle the overlapping ranges and formats as easily as it does now. The nested-formats, MLF (multi-label format) overlapping ranges, preloadfmt, and notsorted options were not available in PROC TABULATE then.

We chose PROC COMPUTAB because it is column and row based. It can create rows and columns for any crossing of categorical and analysis variables. Furthermore, it can handle rows and groups of rows that are not mutually exclusive, a much-needed and desirable function that was not available in PROC TABULATE in previous versions of SAS®. As a result, PROC COMPUTAB worked well for us for last 18 years.

We used SAS® ODS (Output Delivery System) to create HTML tables using the data and listing outputs generated from PROC COMPUTAB. However, the ODS HTML tables output are not Section 508 compliance. Section 508 requires that all contents on federal agencies websites be accessible to people with disabilities. We wrote custom macro programs to convert HTML tables into Section 508 compliant web pages. More recently, we used SAS® DDE (Dynamic Data Exchange) to export data from SAS® datasets to Excel. We also wrote custom macro programs to make the Excel tables into 508 compliant Excel tables before posting them to the Web.

Needless to say, this was a very complex and tedious process to follow. As changes made to the record layout of rows and columns in PROC COMPUTAB, we also had to make changes to the custom macro programs. The macro programs had grown in complexity to handle all the tables; therefore we needed a new and better solution with SAS® to create hundreds of Excel tables without any extra post-processing.

Time has changed and PROC TABULATE and PROC REPORT have evolved. It is time for us to retire PROC COMPUTAB and migrate to PROC TABULATE and PROC REPORT. A noteworthy to mention is the fact that PROC

COMPUTAB is one of the financial forecasting procedures available in SAS/ETS® software and it is not part of BASE SAS®. As a result, PROC COMPUTAB is not in great demand nor heavily invested in as in the case of PROC TABULATE and PROC REPORT.

PROC TABULATE and PROC REPORT to the RESCUE!

When we started the mission to convert all of our Poverty programs from PROC COMPUTAB, we knew that PROC TABULATE and PROC REPORT were our top contenders for the job. Nevertheless, which one should we choose? One of the main issues with our Poverty reports is that within our tables we have several sub-tables many of which have overlapping ranges. For example, we might have one sub-table that has different age groups for each row. One row might be 18-64 years while another row is 18-24 years. With PROC COMPUTAB, we had two lines of code to get the totals for these two age groups. However, with PROC TABULATE we can take advantage of the multi-label format (MLF) options, mlf and preloadfmt.

```
proc format;
value agef (notsorted multilabel)
                0-17    = '~S={foreground=white}..~S={}Under 18 years'
                0-4     = '~S={foreground=white}....~S={}Under 5 years (1)'
                5-17    = '~S={foreground=white}....~S={}5 to 17 years'

                18-64   = '~S={foreground=white}..~S={}18 to 64 years'
                18-24   = '~S={foreground=white}....~S={}18 to 24 years'
                25-34   = '~S={foreground=white}....~S={}25 to 34 years'
                35-44   = '~S={foreground=white}....~S={}35 to 44 years'
                45-54   = '~S={foreground=white}....~S={}45 to 54 years'
                55-59   = '~S={foreground=white}....~S={}55 to 59 years'
                60-64   = '~S={foreground=white}....~S={}60 to 64 years'

        65-high         = '~S={foreground=white}..~S={}65 years and over'
                65-74   = '~S={foreground=white}....~S={}65 to 74 years'
                75-high = '~S={foreground=white}....~S={}75 years and over'

       ;

  ods listing close; * <- to suppress listing output;

  proc tabulate data=&target_data out=sums;
   where &wclause;
    class ftype / order=data preloadfmt;
    class &byvar / order=data preloadfmt mlf missing;
    var count poverty;
    weight marsupwt;
      table all &byvar
       ,
       (all ftype='' ) * (count*sum poverty*sum)
          /  printmiss;
   format age agef. age_hhld age2f. ftype $ftypef. a_sex $sexf.;
  run;
```

Here, the “preloadfmt” option indicates using all the format-defined values in the order specified by the format definition. The “mlf” option indicates using multi-label (overlapping) formats. This works in tandem with options (notsorted multilabel) defined in the PROC FORMAT for the age variable. Here, “multilabel” allows for overlaps in the age ranges. SAS® takes the sort order from the format definition with the “notsorted” option.

These few options solve the vast majority of issues we had with PROC COMPUTAB. If only these options were available back in the beginning. PROC TABULATE wins as our computing engine with these options.

Nevertheless, can PROC REPORT do this also? Well, the answer is no at this time. These multi-label options are not yet available with SAS® 9.2. However, PROC REPORT is planned to support the MLF format in SAS® 9.3.

The only hurdle's to us not using PROC TABULATE with a single PROC TABULATE is twofold.

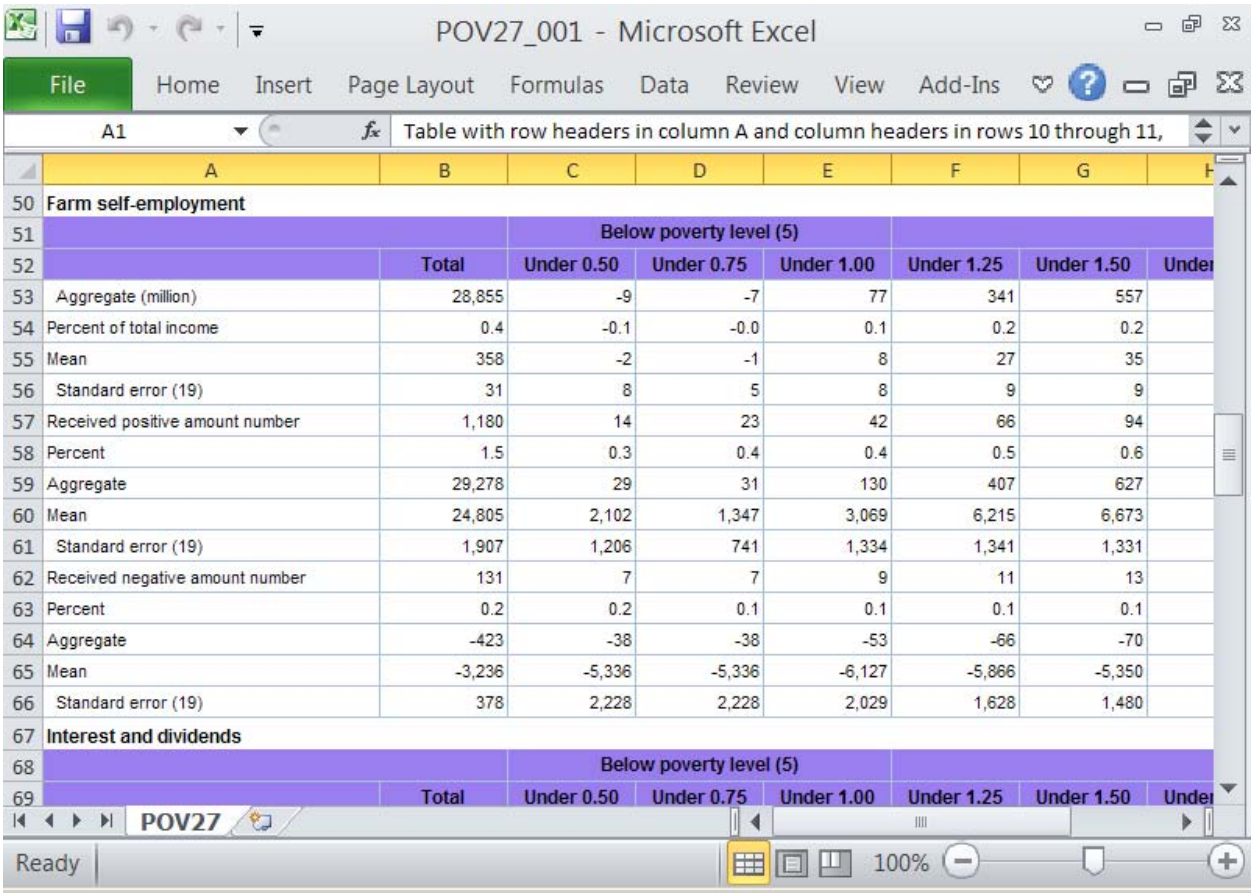
1. Various calculations , such as percentages and standard errors
2. Some tables have varying formats for values within the same column - some with decimals
3. Missing values are represented by different footnote values in the table

```
/* compute percents; re-code missing and low numbers: (X), (B) */
 data &out;
   set sums;
   /* caclculate percentages with special formatting */
   if (count_sum le 0 ) then poverty_pct = -2; else
   if (count_sum lt 75) then poverty_pct = -1; else
          poverty_pct = 100 * poverty_sum / count_sum;
   length fam_type $35 rowhead $130;
    fam_type = put(ftype,$ftypef.);
    if fam_type eq '' then fam_type = 'All people (1)';
     if &byvar eq '' then rowhead = "&header";
        else rowhead = &byvar;
     if index(rowhead,'living') then do;
          rowhead='~S={foreground=white}...~S={}'||rowhead;
                  end;
     rowhead2=tranwrd(rowhead,'~_',' ');
 run;
```

With this in mind, we knew that we would need to create an output data set from PROC TABULATE and pass it to a DATA step for the calculations and missing value representation for the footnotes. We also knew that we would need to have either a second PROC TABULATE or PROC REPORT to create the final table with all the calculations and footnotes.

We initially thought to use TABULATE as the reporting tool, but we chose PROC REPORT in the end for a couple of reasons:

1. Varying formats for values in a single column – some whole numbers and some with decimals (Display 1). Use COMPUTE DEFINE feature of PROC REPORT to solve this
2. Two or more footnote formats represent missing values in the same column. To solve this, we use the COMPUTE DEFINE feature of PROC REPORT (Display 2)
3. Some of the Poverty tables have multiple nested columns headers and the output from PROC REPORT looks nicer with these tables



	A	B	C	D	E	F	G
50	**Farm self-employment**						
51			Below poverty level (5)				
52		Total	Under 0.50	Under 0.75	Under 1.00	Under 1.25	Under 1.50
53	Aggregate (million)	28,855	-9	-7	77	341	557
54	Percent of total income	0.4	-0.1	-0.0	0.1	0.2	0.2
55	Mean	358	-2	-1	8	27	35
56	Standard error (19)	31	8	5	8	9	9
57	Received positive amount number	1,180	14	23	42	66	94
58	Percent	1.5	0.3	0.4	0.4	0.5	0.6
59	Aggregate	29,278	29	31	130	407	627
60	Mean	24,805	2,102	1,347	3,069	6,215	6,673
61	Standard error (19)	1,907	1,206	741	1,334	1,341	1,331
62	Received negative amount number	131	7	7	9	11	13
63	Percent	0.2	0.2	0.1	0.1	0.1	0.1
64	Aggregate	-423	-38	-38	-53	-66	-70
65	Mean	-3,236	-5,336	-5,336	-6,127	-5,866	-5,350
66	Standard error (19)	378	2,228	2,228	2,029	1,628	1,480
67	**Interest and dividends**						
68			Below poverty level (5)				
69		Total	Under 0.50	Under 0.75	Under 1.00	Under 1.25	Under 1.50

Display 1. Excel Table with Various Decimal Places/Formats in the same column

The compute define block below will compute the values for the variable depending upon the particular row and column in the table. PROC REPORT uses absolute column names, such as _c2_ or col2 to represent column 2 of the report. With a sub setting if statement and an index function on the group (or row) variable, we can define the different format for different cells of the table. This feature of PROC REPORT sold us, since our Poverty tables have many different formats for footnotes at different cells of each table. The first PROC REPORT code illustrates how we used multiple formats with COMPUTE DEFINE to define some cells in each column as a whole numbers and others with decimal places. The results are in Display 1 above.

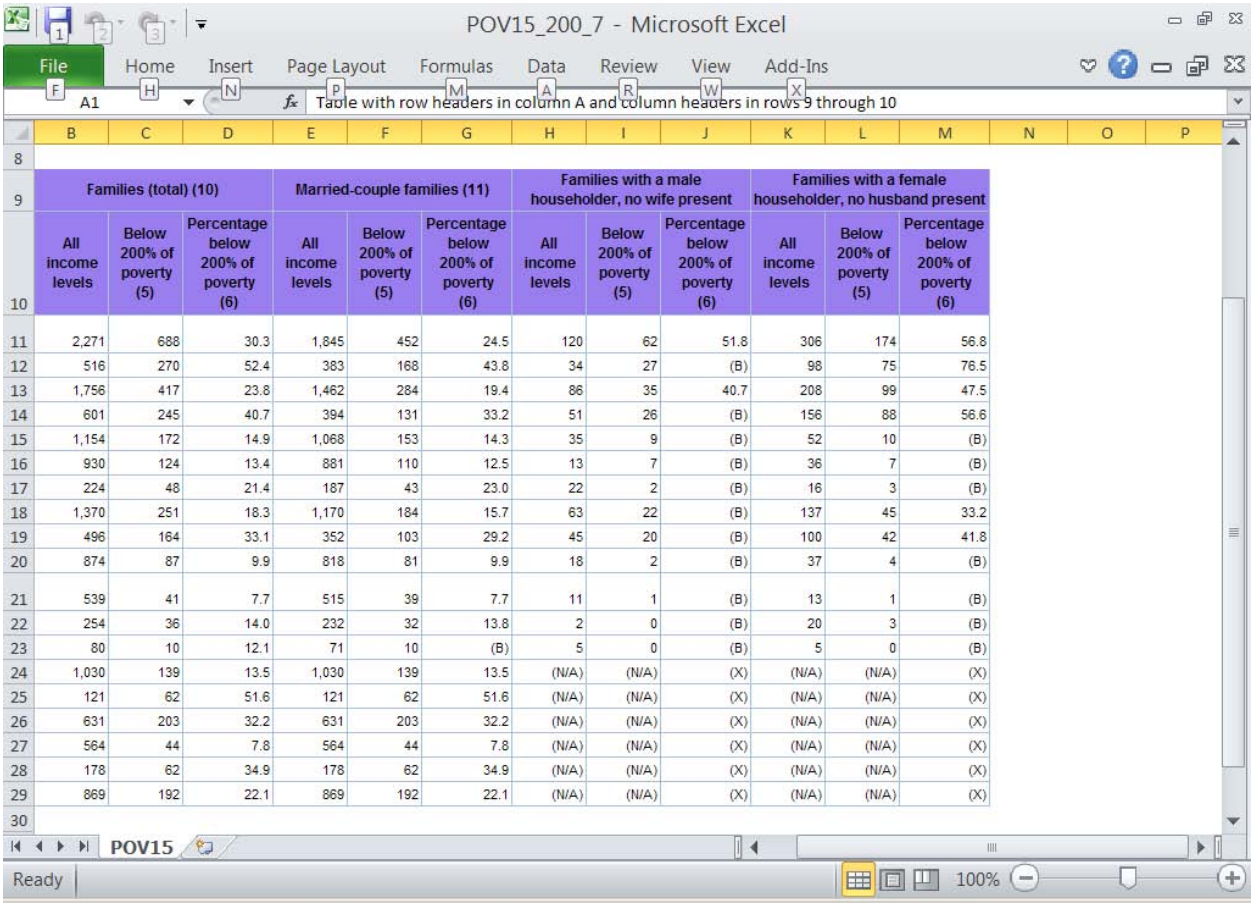
```
proc report data=section&i nowd spanrows split="|" ;
  column rowhead2 col1
    ('Below poverty level (5)' (col2 col3 col4))
    ('Above poverty level (5)' (col5 col6 col7 col8 col9 col10 col11)) ;
  define    rowhead2    /    ""    group    order=data    flow    style(column)    =
[protectspecialchars=off];
  define col1/ 'Total' ;
  define col2/ 'Under 0.50' ;
  define col3/ 'Under 0.75' ;
  define col4/ 'Under 1.00' ;
  define col5/ 'Under 1.25' ;
  define col6/ 'Under 1.50' ;
  define col7/ 'Under 1.75' ;
  define col8/ '1.00 and over' ;
  define col9/ '1.25 and over' ;
  define col10/ '1.50 and over' ;
  define col11/ '1.75 and over' ;

  %do c=1 %to 11; /* Set the Excel format based on whether the value is a percent or
not */
```

```
    compute col&c;
  if index(rowhead2,'Percent') and left(compbl(col&c))="-0.0" then
   call define(_col_,"style","style={tagattr=""format:-0.0"" just=right}");
    else if index(rowhead2,'Percent') and left(compbl(col&c)) ne "-0.0" then
     call define(_col_,"style","style={tagattr=""format:0.0"" just=right}");
      else if index(rowhead2,'Aggregate') and left(compbl(col&c))="-0" then
       call define(_col_,"style","style={tagattr=""format:-0"" just=right}");
        else if index(rowhead2,'Mean') and left(compbl(col&c))="-0" then
         call define(_col_,"style","style={tagattr=""format:-0"" just=right}");
          else        call        define(_col_,"style","style={tagattr=""format:#,##0""
just=right}");
           endcomp;
  %end;
  run;
```

The COMPUTE DEFINE block below will compute the values for the variable depending upon the particular row and column in the table again. This time we use a different user-defined format to represent missing values in Display 2 below. Missing values in display 2 are represented by either a '0' or 'N/A' in the table. We can set the appropriate format to use in the table with a compute define block in PROC REPORT. Once again, the format depends upon the row and column of the table.

```
proc report data=section&i nowd spanrows split='|' out=rept&i;
column rowhead2 fam_type, (count_sum poverty_sum poverty_pct);
define rowhead2 / "&race2 | Below &ratio2.%" group order=data flow style(column) =
[protectspecialchars=off];
define fam_type / '' across order=data preloadfmt missing ;
define count_sum / 'All | income | levels' style(column)=[tagattr="format:#,##0"];
define    poverty_sum    /    "Below    |    &ratio2.%    of    |    poverty    (5)"
style(column)=[tagattr="format:#,##0"];
define poverty_pct / "Percentage | below &ratio2.% of | poverty | (6)"
f=pctf. style(column)=[tagattr="format:##0.0"] ;

compute count_sum;
if scan(rowhead2,4,'')='(1)' then do;
call define('_c2_','format','num2f.');  /* Use 0 (zero) as format for column 2 */
call define('_c5_','format','num2f.');
call define('_c8_','format','num2f.');
call define('_c11_','format','numf.'); /* Use NA as format for missings in column 11
for Under 5 years */
end;

else do;
call define('_c2_','format','num2f.');
call define('_c5_','format','num2f.');
call define('_c8_','format','num2f.');
call define('_c11_','format','num2f.');
end;
endcomp;

compute poverty_sum;
if scan(rowhead2,4,'')='(1)' then do;
call define('_c3_','format','num2f.');  /* Use 0 (zero) as format for column 2 */
call define('_c6_','format','num2f.');
call define('_c9_','format','num2f.');
call define('_c12_','format','numf.'); /* Use NA as format for missings in columns 12
for Under 5 years */
end;

else do;
call define('_c3_','format','num2f.');
call define('_c6_','format','num2f.');
call define('_c9_','format','num2f.');
call define('_c12_','format','num2f.');
end;
endcomp;
```

```
run
```



	Families (total) (10)			Married-couple families (11)			Families with a male householder, no wife present			Families with a female householder, no husband present		
	All income levels	Below 200% of poverty (5)	Percentage below 200% of poverty (6)	All income levels	Below 200% of poverty (5)	Percentage below 200% of poverty (6)	All income levels	Below 200% of poverty (5)	Percentage below 200% of poverty (6)	All income levels	Below 200% of poverty (5)	Percentage below 200% of poverty (6)
	2,271	688	30.3	1,845	452	24.5	120	62	51.8	306	174	56.8
	516	270	52.4	383	168	43.8	34	27	(B)	98	75	76.5
	1,756	417	23.8	1,462	284	19.4	86	35	40.7	208	99	47.5
	601	245	40.7	394	131	33.2	51	26	(B)	156	88	56.6
	1,154	172	14.9	1,068	153	14.3	35	9	(B)	52	10	(B)
	930	124	13.4	881	110	12.5	13	7	(B)	36	7	(B)
	224	48	21.4	187	43	23.0	22	2	(B)	16	3	(B)
	1,370	251	18.3	1,170	184	15.7	63	22	(B)	137	45	33.2
	496	164	33.1	352	103	29.2	45	20	(B)	100	42	41.8
	874	87	9.9	818	81	9.9	18	2	(B)	37	4	(B)
	539	41	7.7	515	39	7.7	11	1	(B)	13	1	(B)
	254	36	14.0	232	32	13.8	2	0	(B)	20	3	(B)
	80	10	12.1	71	10	(B)	5	0	(B)	5	0	(B)
	1,030	139	13.5	1,030	139	13.5	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)
	121	62	51.6	121	62	51.6	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)
	631	203	32.2	631	203	32.2	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)
	564	44	7.8	564	44	7.8	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)
	178	62	34.9	178	62	34.9	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)
	869	192	22.1	869	192	22.1	(N/A)	(N/A)	(X)	(N/A)	(N/A)	(X)

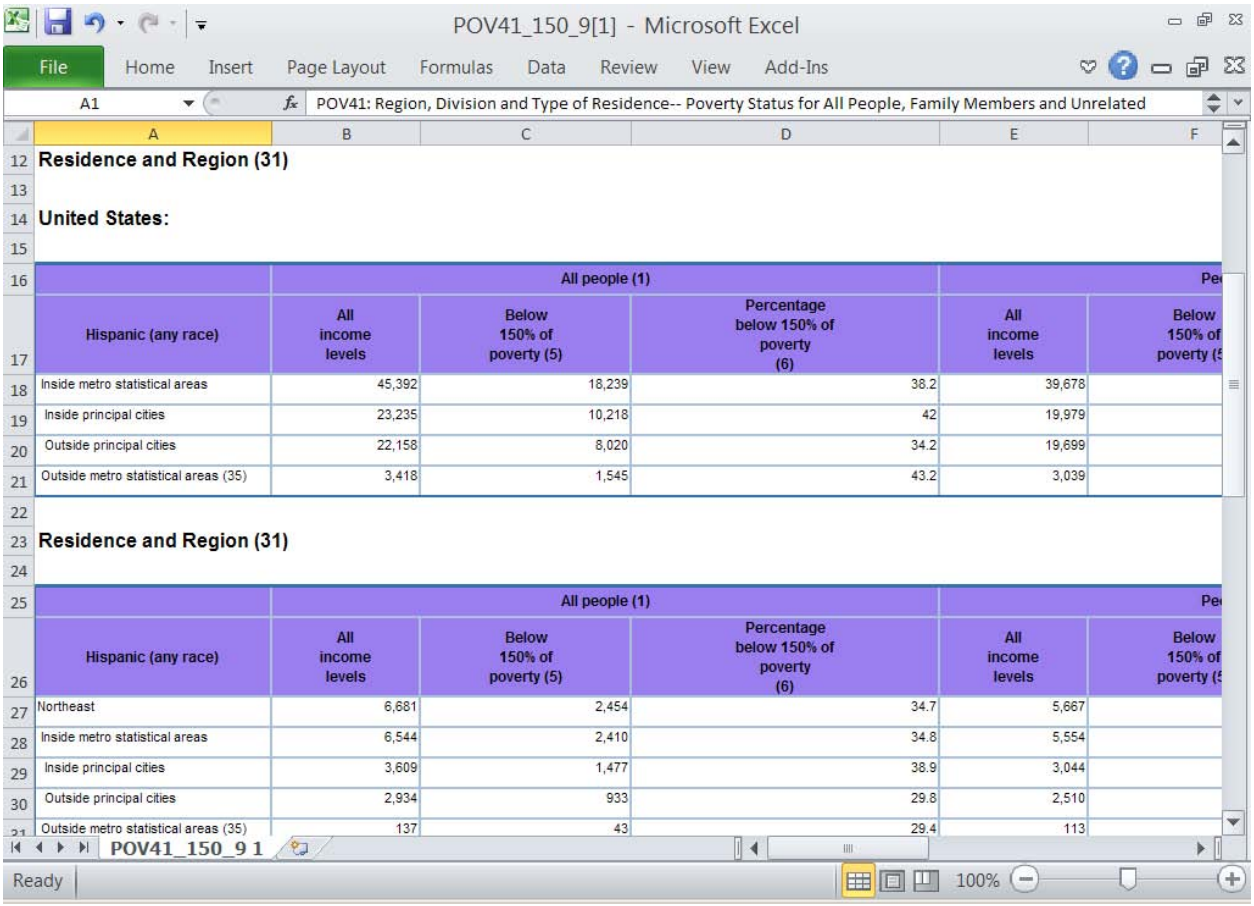
Display 2. Excel Table with Various Footnotes for Missing Values

For all the SAS® Papers and presentations over the years touting either PROC TABULATE or PROC REPORT as the way to go, we prove that you can use both of them in perfect harmony. Now, we just need to wrap our PROC REPORT in ODS to create the excel files.

CREATE HUNDREDS of EXCEL FILES

Now we need to add the final piece to the puzzle. We need to add the ODS code to create the hundreds of Excel files. In order to do it, we need to wrap all of the code in a macro loop to create the hundreds of excel files. We read several SAS® papers on ODS MSOFFICE2K and ODS TAGSETS EXCELXP destinations. Yet, which one is best for us? In order to find out, we have to explore both and see what the Excel tables look like.

We started with ODS MSOFFICE2K and liked what we saw at first, but then we ran into a snag with it. Each of our Poverty tables (or Excel worksheets) has multiple sub-tables within them. We quickly noticed that with MSOFFICE2K we had some very long column widths in the excel files and the widths of columns varied. Excel was not honoring the column widths that we specified in PROC REPORT (Display 3).



Display 3. Output using MSOffice2K – with multiple sub-tables there are varying column lengths

We realized that Excel would not allow any cell width settings when using multiple sub-tables with MSOFFICE2K. The MSOFFICE2K destination generates output in the HTML file format. When Microsoft Excel comes across multiple table tags on a single worksheet with the HTML file format it resets the cell widths. This situation left us with three choices:

1) Live with column widths imposed by Excel

2) Post-process the output Excel file from MSOFFICE2K

3) Explore ODS TAGSETS.EXCELXP destination to see if it can do better

ODS TAGSETS.EXCELXP generates output in Microsoft's SpreadsheetML XML format. On our first attempt with EXCELXP tagsets, we experienced the same column width problem as with MSOFFICE2K. However, ODS TAGSETS.EXCELXP has a bevy of options and one of them named "absolute_column_width" solved our varying column width problem. With this option, we can specify a column width for every column in the table. The widths stay the same for all sub-tables. With EXCELXP tagsets, the width in the second sub- table will override the width that we calculate for each column in the first sub-table. This happens because there is no such thing as adding multiple tables to a worksheet with the XML spreadsheet format. Thus, when we set the absolute column width of the first table the column widths remain in effect for the entire table.

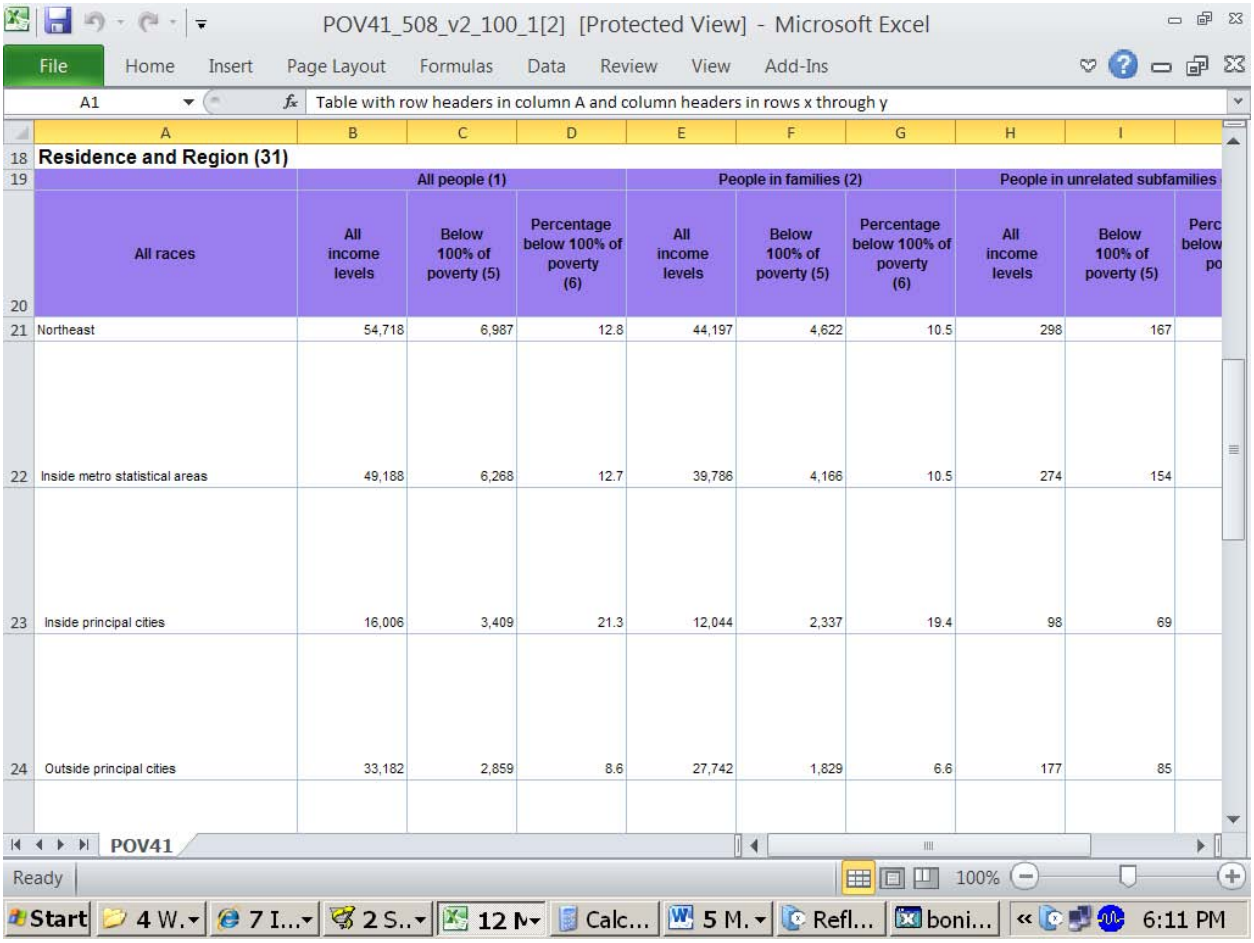
```
ods tagsets.excelxp path="&outp" file="POV01_&ratio2._&raceit..xls"
    style=styles.newstyle options(autofit_height='yes')
options(embedded_titles="yes" embedded_footnotes='yes' sheet_interval='none'
       absolute_column_width="24,10,10,10,10,10,10,10,10,10,10,10,10"
         sheet_name='POV01' skip_space="0,0,0,1,1,0,0");
```

```
title1 j=left color=white "Table with row headers in column A and column headers in
rows 9 through 10, 26 through 27, 43 through 44, and 60 through 61";
title2 j=left bold height=11pt "POV01: Age and Sex of All People, Family Members and
Unrelated Individuals Iterated by Income-to-Poverty Ratio and Race: &incyear";
title3 j=left bold height=11pt "Below &ratio2.% of Poverty -- &race4";
title4 j=left height=9pt underlin=1
link="http://www.census.gov/apsd/techdoc/cps/cpsmar&yr..pdf" "For information on
confidentiality protection, sampling error, nonsampling error, and definitions, see
www.census.gov/apsd/techdoc/cps/cpsmar&yr..pdf";
title5 j=left height=9pt "Source: U.S. Census Bureau, Current Population Survey,
&marchcps Annual Social and Economic Supplement.";
title6 j=left height=9pt "(Numbers in thousands)";
```

Additionally, the “embedded_titles” and “embedded_footnote” options allow us to use title and footnote statements. Furthermore, they tell Excel to put them in the worksheet. The link option on the title statement allows us to embed an html hyperlink into the title. The “autofit_height” option helped us with another problem that we encountered with our Excel tables. Many of the rows in our tables are indented. The row heights increased for the rows that were indented. The indented rows had a larger row height than the rows without indentation. Initially, we used   in the format code to generate blank spaces to get the indentations. Later, we changed these over to periods or “white dots” in order to make the Excel tables 508 compliant. We will not discuss that in depth here, since that is a topic for another paper. The point here is that in creating the “white dots” for indentation in the PROC FORMAT, we increased the length of the format.

```
proc format;
  value $sexf (notsorted)
   '1' = '~S={foreground=white}....~S={}Male'
   '2' = '~S={foreground=white}....~S={}Female'
  ;
```

The length of the format text started to override the in-line formatting and it caused the row heights to increase for the rows that were indented (Display4). This was clearly unacceptable. We had to have the rows all the same height. In this case, the “autofit_height” option came to the rescue and forced all rows to be the same height. Additionally, the “skip_space” option was helpful to us for Section 508 compliance, which is the topic for another paper.



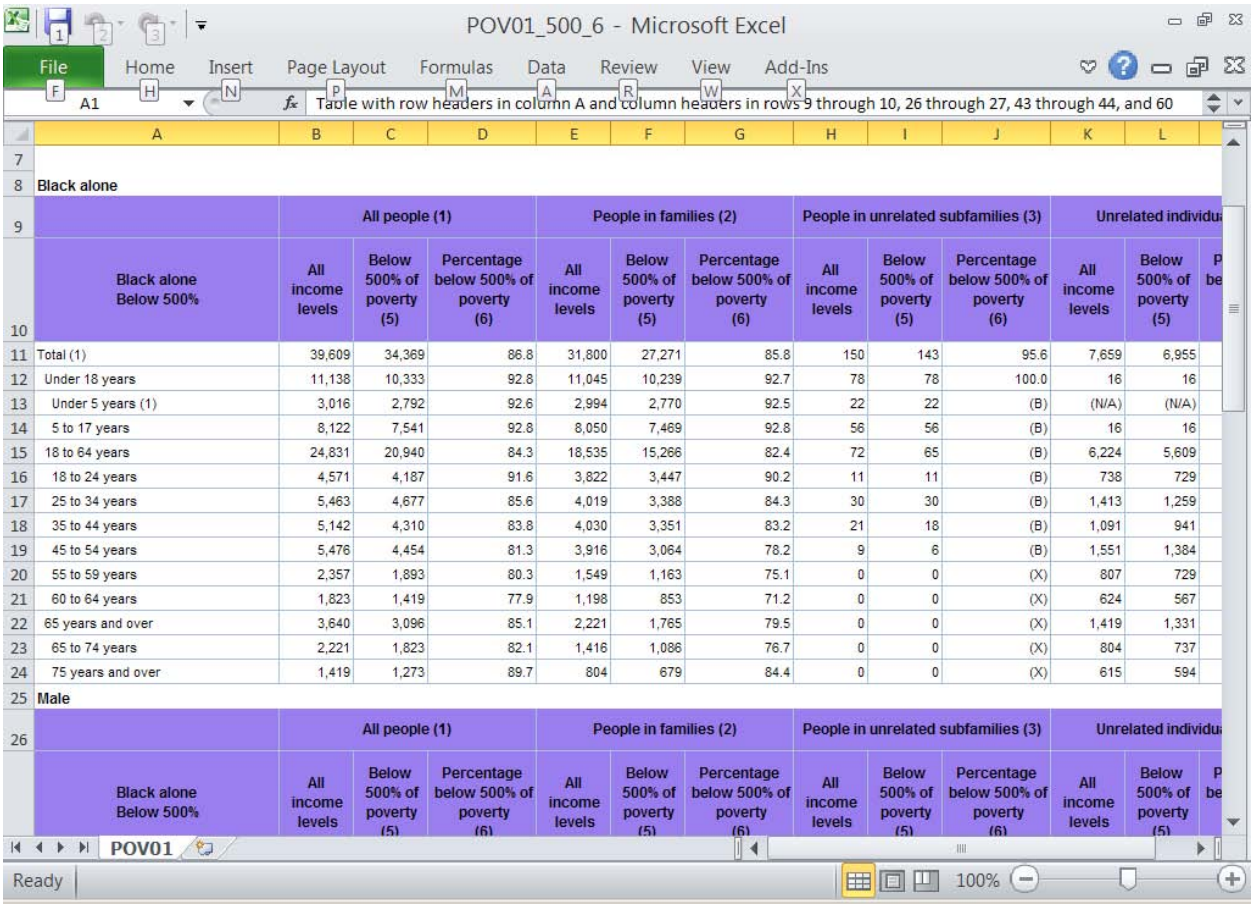
Display 4. Output using TAGSETS.ExcelXP– varying/ longer row heights for row headers with indentation

Lastly, we utilize the TAGATTR formats in the PROC REPORT code. As mentioned earlier, Microsoft Excel does not honor the formats that you use in your SAS® code. It has a mind of its own and likes to impose its formats. One such problem is with zeroes including decimals with leading and trailing zeroes. Microsoft Excel likes to strip these out. The solution to this is to use TAGATTR formats in PROC REPORT. Here is the main macro code that creates hundreds of great looking Excel tables.

```
%macro iterations;  /* Create 135 Excel tables by race and poverty group */
%do raceit=1 %to 9;  /* Testing Nine Race Groups */
     %do percentit=1 %to 15;  /* Fifteen Poverty Group Levels */

%macro print_report_excel;
ods escapechar='~';
ods tagsets.excelxp path="&outp" file="POV01_&ratio2._&raceit..xls"
style=styles.newstyle options(header_data_associations="yes" autofit_height='yes')
options(embedded_titles="yes"       embedded_footnotes='yes'       sheet_interval='none'
absolute_column_width="24,10,10,10,10,10,10,10,10,10,10,10,10"
sheet_name='POV01' skip_space="0,0,0,1,1,0,0");

title1 j=left color=white "Table with row headers in column A and column headers in
rows 9 through 10, 26 through 27, 43 through 44, and 60 through 61";
title2 j=left bold height=11pt "POV01: Age and Sex of All People, Family Members and
Unrelated Individuals Iterated by Income-to-Poverty Ratio and Race: &incyear";
title3 j=left bold height=11pt "Below &ratio2.% of Poverty -- &race4";
title4                  j=left                  height=9pt                  underlin=1
link="http://www.census.gov/apsd/techdoc/cps/cpsmar&yr..pdf"   "For   information   on
```

```
confidentiality protection, sampling error, nonsampling error, and definitions, see
www.census.gov/apsd/techdoc/cps/cpsmar&yr..pdf";
title5 j=left height=9pt "Source: U.S. Census Bureau, Current Population Survey,
&marchcps Annual Social and Economic Supplement.";
title6 j=left height=9pt "(Numbers in thousands)";


%do i=1 %to 4; /* There are 4 sub-tables in the Excel worksheet */

/* generate output */

%if &i=1 %then %do;
 title7 " ";
 title8 j=left bold height=9pt "&&title&i";
%end;
%else %do;
 title1;title2;title3;title4;title5;title6;title7;title8;
  title1  j=left bold height=9pt "&&title&i";
%end;

proc report data=section&i nowd spanrows split='|' out=rept&i;
column rowhead2 fam_type, (count_sum poverty_sum poverty_pct);
define rowhead2 / "&race2 | Below &ratio2.%" group order=data flow style(column) =
[protectspecialchars=off];
define fam_type / '' across order=data preloadfmt missing ;
define count_sum / 'All | income | levels' style(column)=[tagattr="format:#,##0"];
define    poverty_sum    /    "Below    |    &ratio2.%    of    |    poverty    (5)"
style(column)=[tagattr="format:#,##0"];
define poverty_pct / "Percentage | below &ratio2.% of | poverty | (6)"
f=pctf. style(column)=[tagattr="format:##0.0"] ;



%end;
%print_report_excel;
footnote;
ods _all_ close;

  %end;
 %end;
%mend iterations;
%iterations;
```

The tagattr=”format:##0.0” preserves the leading and trailing zeroes in the final excel file. It is clear to see why we chose ODS TAGSETS EXCELXP over MSOFFICE2K to create hundreds of great looking complex excel files on the file. The TAGSET options solved our column width and row height issues. Moreover, the TAGATTR formats allowed us to get those leading and trailing zeroes into the Excel file. Display 5 shows part of one final table with leading and trailing zeroes preserved.



	A	B	C	D	E	F	G	H	I	J	K	L
7												
8	Black alone											
9		All people (1)			People in families (2)			People in unrelated subfamilies (3)			Unrelated individu	
10	Black alone Below 500%	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)
11	Total (1)	39,609	34,369	86.8	31,800	27,271	85.8	150	143	95.6	7,659	6,955
12	Under 18 years	11,138	10,333	92.8	11,045	10,239	92.7	78	78	100.0	16	16
13	Under 5 years (1)	3,016	2,792	92.6	2,994	2,770	92.5	22	22	(B)	(N/A)	(N/A)
14	5 to 17 years	8,122	7,541	92.8	8,050	7,469	92.8	56	56	(B)	16	16
15	18 to 64 years	24,831	20,940	84.3	18,535	15,266	82.4	72	65	(B)	6,224	5,609
16	18 to 24 years	4,571	4,187	91.6	3,822	3,447	90.2	11	11	(B)	738	729
17	25 to 34 years	5,463	4,677	85.6	4,019	3,388	84.3	30	30	(B)	1,413	1,259
18	35 to 44 years	5,142	4,310	83.8	4,030	3,351	83.2	21	18	(B)	1,091	941
19	45 to 54 years	5,476	4,454	81.3	3,916	3,064	78.2	9	6	(B)	1,551	1,384
20	55 to 59 years	2,357	1,893	80.3	1,549	1,163	75.1	0	0	(X)	807	729
21	60 to 64 years	1,823	1,419	77.9	1,198	853	71.2	0	0	(X)	624	567
22	65 years and over	3,640	3,096	85.1	2,221	1,765	79.5	0	0	(X)	1,419	1,331
23	65 to 74 years	2,221	1,823	82.1	1,416	1,086	76.7	0	0	(X)	804	737
24	75 years and over	1,419	1,273	89.7	804	679	84.4	0	0	(X)	615	594
25	Male											
26		All people (1)			People in families (2)			People in unrelated subfamilies (3)			Unrelated individu	
	Black alone Below 500%	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)	Percentage below 500% of poverty (6)	All income levels	Below 500% of poverty (5)

Display 5. Final table with leading and trailing zeroes preserved . Row heights are the same. Indentation is preserved. Column width are the same.

CONCLUSION

We completed our main objectives to get rid of PROC COMPUTAB and extra DDE post-processing. We eliminated PROC COMPUTAB and its limitations by using the power of both PROC TABULATE and PROC REPORT. We took advantage of TABULATE's multi-label options (mlf, preloadfmt) to deal with overlapping format ranges. Additionally, we used TABULATE as our main computing engine. PROC REPORT was our main reporting tool. Moreover, we took advantage of its compute define functionality to use more than one format for the same column depending upon the row and column of the table. Lastly, we developed great looking and complex Excel tables with the power of ODS TAGSETS.EXCELXP. The TAGSETS.EXCELXP options solved our problems with varying column widths and row heights. The TAGATTR formats put the leading and trailing zeroes into the Excel files.

The multilabel (MLF) format is planned to be supported in PROC REPORT in SAS® 9.3. This enhancement was a previous SASware Ballot item. PROC REPORT will be able to use the format label for overlapping ranges to create subgroup combinations. The MLF will be supported in the DEFINE statements for both group and across variables. It may be possible to remove PROC TABULATE as the computing engine in the future and use only PROC REPORT. This will be a consideration for us at Census as we plan to create hundreds of Excel tables for our ASEC Income tables in 2013.

ACKNOWLEDGMENTS

The authors would like to thank Leonid Batkhan of SAS® Institute for his consulting service on a proof-of-concept prototype converting PROC COMPUTAB programs using PROC TABULATE and PROC REPORT. We would also

like to thank Jane Eslinger and Chevell Parker of SAS® Technical Support Staff for their valuable programming solutions on using ODS TAGSETS.EXCELXP, ODS MSOFFICE2K, PROC TABULATE and PROC REPORT.

CONTACT INFORMATION

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

Name: Christopher J. Boniface
U.S. Census Bureau
Washington D.C., 20233
Work Phone: (301)763-5769
E-mail: christopher.j.boniface@census.gov

Name: Hung X. Pham
U.S. Census Bureau
Washington D.C., 20233
Work Phone: (301)763-5909
E-mail: hung.xuan.pham@census.gov

Name: Nora P. Szeto
U.S. Census Bureau
Washington D.C., 20233
Work Phone: (301)763-5920
E-mail: nora.p.szeto@census.gov

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.