Paper 176-25

Support System - A Web-Based SAS/IntrNet™ Tabulator

Stephen Heacock, U.S. Census Bureau

ABSTRACT

Do you want to develop interactive web pages using SAS® macro, a language with which you probably are already familiar? This paper discusses how, utilizing SAS/IntrNet™ and SAS® macro, the system puts much of the power of PROC tabulate into users' hands. This web-based application is simple to learn and is tailored for the Current Population Survey (CPS) data sets. It provides a mechanism for users, with or without any programming experience, to easily generate complex, but attractive, tables. This paper illustrates the key components for the developer: the HTML code required to use SAS/IntrNet™; validation of parameters provided by the user via an HTML form; joining data sets and building the tabulate code to meet the user's selected conditions.

The CPS is a monthly survey of about 50,000 households conducted by the U.S. Census Bureau for the Bureau of Labor Statistics. It is the primary source of information for many national economic and demographic indicators including poverty, income, unemployment, disability, and health insurance coverage. This application has improved the timeliness and ease with which analysts can create their own tables for reviewing data.

INTRODUCTION

Statisticians in our unit need to view the summarized CPS data in a multitude of cross tabulations. The Census Bureau developed the CPS Support System, a dynamic, web-based tabulator to put control in the hands of the user. It is easy to deploy web-based systems (an advantage over SAS/AF®); the user simply needs a browser.

SAS/IntrNet™ technology allows the developer to create interactive applications without learning new cgi-scripting languages. SAS® programmers can use tools they already know -- base SAS® macro. Users can generate tables in a fraction of the time it would take to do so by writing their own programs: no coding, no syntax to worry about, no testing and debugging. By being more efficient in their table generation, analysts can devote their energies to their specialized talent -- statistical review, rather than to programming.

You can apply the strategy used by the CPS Support System to your data sets to produce a tabulator to fit your organization's needs.

OVERVIEW OF THE PROCESS

- The user enters the CPS Support System via their browser; there are links to supporting documentation pages
- The user completes an HTML form (most fields have default values so tables can be generated in a few keystrokes) and presses a button to get the table
- SAS/IntrNet™ software converts the HTML form variables to SAS® macro variables and executes the developer's custom-designed SAS® program
- This customized SAS® program validates entries, builds formats, joins three input data sets (person, family, household data), creates variables to be derived from existing variables, runs a PROC tabulate, and redirects the output to the browser via the SAS® HTML-formatting macro, tab2htm.

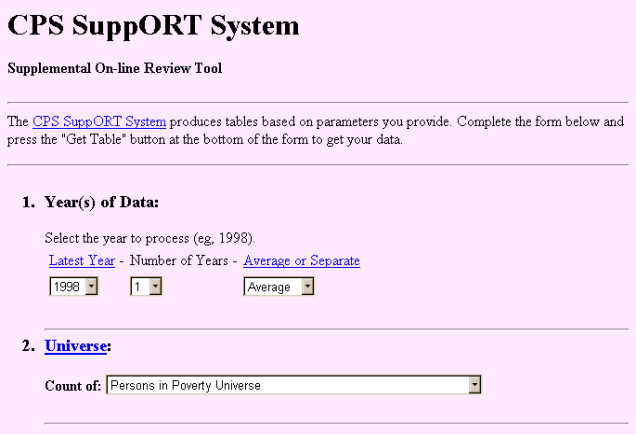
We are running SAS® 6.12 on a Sun Microsystems Solaris UNIX machine (inside the Census firewall) and accessing the CPS Support System using Netscape Navigator™ on PCs. The input data set sizes are approximately, 130,000, 57,000, and 50,000 records, respectively; response time varies depending upon the complexity of the table but is as short as 25 seconds for a simple cross-tabulation.

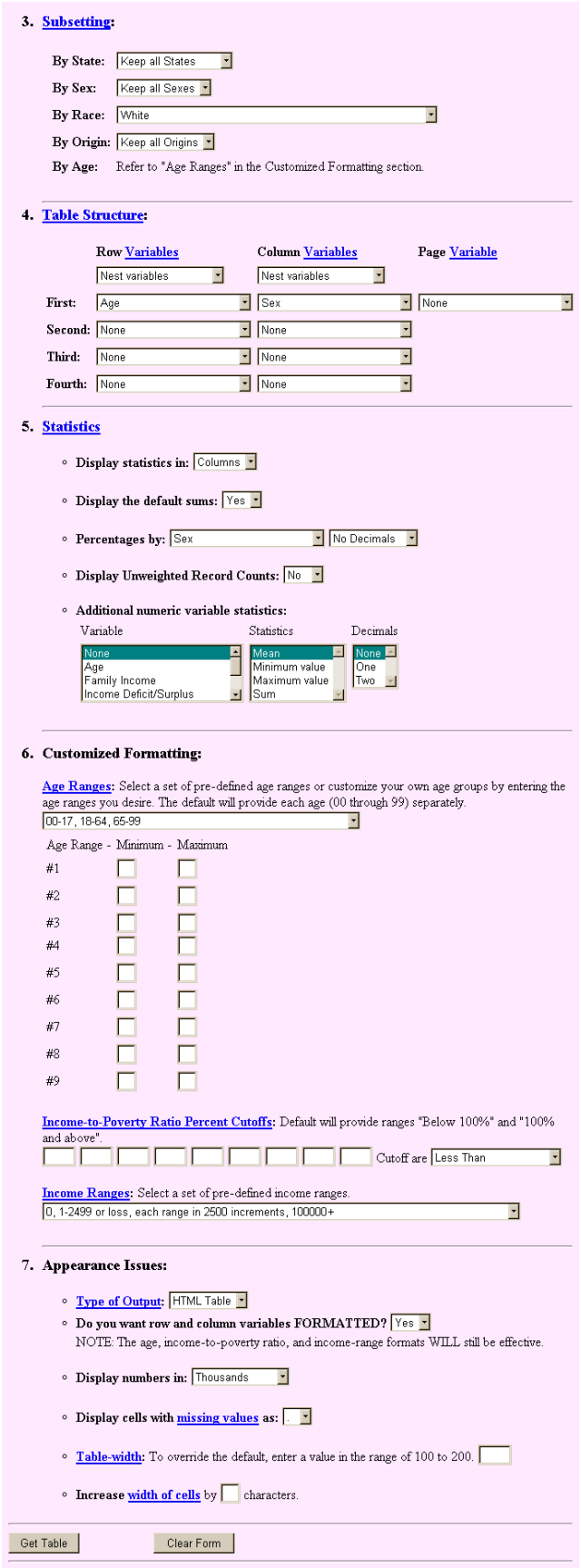
FUNCTIONALITY

- Provides access to the 10 most recent years of CPS data
- Tables can be generated for a single year's data or for 2- or 3-year averages
- Available are 13 different person or family universes and 28 different variables
- The selected universe can be further subsetted by up to five specific variables; for example, for a given person universe, the user can limit the table to white, Hispanic females between the ages of 18 and 55 in the state of California
- User customized-formatting is available for two variables
- The generated table can have row, column, and page dimensions, up to four different variables (nested or concatenated) for each of the row and column dimensions; effectively, the user can obtain a nine-way cross-tabulation
- Statistics available are weighted sums of persons or families (the default), percents for a single variable in the table, unweighted record counts, and up to four statistics (mean, minimum, maximum, sum) for one of six numeric variables (including age, family income, and number of persons in family). Any combination of these statistics can be generated.
- Page and cell widths can be changed to provide partial control over the parsing of column labels and columns across a page (for wide tables)

WHAT THE USER SEES IN THE BROWSER

Below is the CPS Support System's main page; there are links to other pages with a system overview, universe and variable descriptions, example SAS® code to join the input data sets and run a tabulate, and other tips to guide the CPS Support System user.





Below is an example of a generated table: age (using one of the pre-defined formats) by sex for persons in the poverty universe for the White subset displaying sums and percentages by sex.

U.S. Census Bureau

CPS Data for Reference Year: 1998

Persons in Poverty Universe

Percentages by Sex

(Numbers in Thousands)

NOTE: The Current Population Survey March Supplement is an annual survey of approximately 50,000 households nationwide. Therefore, use extreme caution when making inferences when the cell sizes are small.

White	Totals		Sex			
			Male		Female	
	Persons		Persons		Persons	
	Sum	PCT	Sum	PCT	Sum	PCT
Totals	222,837	100%	109,626	49%	113,210	51%
Age						
00 to 17	56,016	100%	28,714	51%	27,303	49%
18 to 64	138,061	100%	68,675	50%	69,386	50%
65 to 99	28,759	100%	12,238	43%	16,521	57%

Source: U.S. Census Bureau

Current Population Survey March 1999

LIMITATIONS

The system is limited to PROC tabulate's functionality. Ideally, the analysts would like Pareto-interpolated medians (for income variables) and standard errors (for which the Census Bureau has its own complicated algorithm). Note that simple medians are available in tabulate beginning with SAS® version 7.

HTML does not have SAS/AF®'s flexibility. For example, the system would be improved if the percentage selection box choices were limited to the user-selected variables for the row and column dimensions for the table. If the user selects a percentage variable that is not in the row and column dimensions, the data validation section of the customized program identifies the invalid combination of parameters and reports the conflict to the user. If the disallowable variables were excluded from the selection list, the user would have fewer possible combinations of invalid parameters.

HTML MAIN PAGE

Imbed the following key lines of code in the HTML to allow it to work with SAS/IntrNet™. The FORM tag is used for all HTML

forms. The ACTION parameter is the name of the cgi program: to use SAS/IntrNet™, always use the SAS® broker program. The two hidden input variables indicate the name of the developer's custom-designed SAS® program, for the CPS Support System, support.sas in the 'myhome' libname (defined in the SAS/IntrNet™ autoexec file) and the type of service (there are two options available) for running SAS/IntrNet™. The remaining code (not shown) is standard HTML form code. Note that this paper does not address SAS/IntrNet™ installation.

```
<FORM ACTION="/cgi-bin/broker" METHOD="POST">
<INPUT TYPE="HIDDEN" NAME="_PROGRAM"
      VALUE="myhome.support.sas">
<INPUT TYPE="HIDDEN" NAME="_SERVICE"
      VALUE="newsas">
```

PARAMETER VALIDATION

As with any interactive program, it is the programmer's responsibility to make it robust. This responsibility includes validating parameters to ensure that they will not cause the customized SAS® program to abort. When SAS® aborts due to an invalid parameter, a blank screen is generally displayed for the user. This gives the user no direction for resolving the problem. Instead, identify invalid parameters (or combinations of parameters), bypass the code that would otherwise abort, and display a message that tells the user what prevented the generation of the requested table.

The code used to validate the macro variable xcelwid, the number of characters to add to the default cell width to determine the final cell width, is shown below.

```
%*** set message to display if error;
%let mymsg=Invalid additional cell width
      (&xcelwid) provided -- must be in
      the range of 0 to 9.;
%let mylen=%length(%superq(xcelwid));
%*** value is null or blank - treat as zero;
%if &mylen = 0 or
    %quote(&xcelwid) = %quote( ) %then %do;
  %let xcelwid=0;
%end;
%*** value is > 9 - an error;
%else %if &mylen ne 1 %then %do;
  %goto badinput;
%end;
%else %do;
  %*** check each character for non numeric
       (invalid) entries. Length is always
       one, but this demonstrates how to
       validate variables with longer lengths;
  %do i=1 %to &mylen;
    %if(%substr(%superq(xcelwid),&i,1) < 0)
       or
       (%substr(%superq(xcelwid),&i,1) > 9)
                    %then %do;
      %goto badinput;
    %end;
  %end;
%end;
.
.
.
%badinput:
  data _null_;
    file _webout;
    put 'Content-type: text/html';
    put;
    put '<HTML>';
    put '<BODY>';
    put "<H2>&mymsg</H2>";
    put '</BODY>';
    put '</HTML>';
  run;
```

Note that when a parameter fails the validation check, the program reports the invalid parameter with simple HTML code generated with put statements. The keyword "_webout", used in the file statement, references the terminal screen. The first two "put" statements are required (in the format shown) to interface with the browser.

The CPS Support System only reports the first invalid parameter encountered. Thus, if the user enters a number of invalid parameters, it will take a few iterations to generate a table. More elaborate code could be developed to report all invalid parameters at one time.

JOINING THE DATA SETS

Simplified code used to join two of the three data sets is shown below. The author excluded code for some of the universes, variables, and subset conditions. This shows how macro code is used to build the DATA step with the proper observations and variables. For example, the "where clauses" to apply to the input data sets must vary by universe to extract the records that comprise that selected universe.

Note that the macro variables with names ending with "yn" are booleans to indicate whether or not the associated variable has been selected for any row, column, or page variable.

```
data ds1&year
     (drop=hhdrel a_famrel
         %*** if the work experience variable
              is included in the table, drop
              the intermediate variables used
              to calculate the recoded
              variable;
         %if &wrkexpyn = Y %then %do;
           hrswk wkswork
         %end;
     );
  merge cps.family
    (keep=fh_seq  ffpos ftype fkind
        %*** if the poverty ratio variable is
             included in the table, keep the
             input data set variables required
             to assign the recoded value;
        %if &povratyn = Y %then %do;
          fpovcut fincome
        %end;
     in=in_fam
  %*** apply to the input data set the where
       clause that corresponds to the selected
       universe;
  %if &univ = PPOV %then %do;
     where=(ftype ne '3')
  %end;
  %else %if &univ = PINPFAM or
            &univ = FPRIMARY %then %do;
     where=(ftype = '1')
  %end;
  %else %if &univ = PINMCPF or
            &univ = FMCPF %then %do;
     where=(ftype = '1' and fkind = '1')
  %end;
    )
```

```
      cps.person
    (keep=ph_seq pf_seq a_age cpswgt
          hhdrel a_famrel
      %if &raceyn = Y %then %do;
         a_race
      %end;
      %if &originyn = Y %then %do;
         a_origin
      %end;
      %if &sexyn = Y %then %do;
         a_sex
      %end;
      %if &wrkexpyn = Y %then %do;
         hrswk wkswork
      %end;
     in=in_per
     rename=(ph_seq=fh_seq pf_seq=ffpos)
     where=((a_age ge "&minage" and
             a_age le "&maxage")
       %if &univ = FPRIMARY or
           &univ = FMCPF %then %do;
            and (hhdrel = '1')
       %end;
           %*** close where clause;
           )
    %*** close person data set options;
    );
  by fh_seq ffpos;
  if in_fam and in_per then do;
    %if &univ ne PALL %then %do;
      *** remove unrelated individuals under
          15 - not in the poverty universe;
      if (ftype = '5' and a_age < '15') then
          delete;
    %end;
    *** assign constant 1 for tabulate var;
    onecnst=1;
    %*** if the poverty ratio variable is
         included in the table, create the
         variable with the proper value;
    %if &povratyn = Y %then %do;
      povratio = (fincome / fpovcut) * 100;
    %end;
    %*** if the work experience variable is
         included in the table, create the
         variable with the proper value;
    %if &wrkexpyn = Y %then %do;
      if a_age le '14' then
         wkexp_r = 'r4';
      else if wkswork = '00' then
         wkexp_r = 'r3';
      else if hrswk ge '35' and
              wkswork ge '50' then
         wkexp_r = 'r1';
      else if hrswk lt '35' or
              wkswork lt '50' then
         wkexp_r = 'r2';
      else
         %*** implies error in input data set;
         wkexp_r = '??';
    %end;
    *** output matches only;
    output;
  end;
run;
```

PROC TABULATE CODE

Simplified PROC tabulate code is shown below. (The author excluded code for the third and fourth rows and columns, the table options, the formats, labels, keylabels, and more.) Note that the FORMCHAR value shown below is required for the HTML-formatting macro, tab2htm, to display the data in an HTML table.

The macro code is parsed in a manner that is sometimes cumbersome to read but is designed to satisfy the PROC tabulate syntax for every possible combination of parameters provided by the user.

The variable in the var statement, onecnst, always contains a value of one. The weight, cpswgt, is applied to this variable. Optionally, a second variable is included in the var statement when statistics are requested for one of the numeric variables.

The variables listed in the HTML form selection lists are descriptive names. However, the actual data set variable names (which the user does not see) are returned from the HTML form and become the SAS® macro variables row1, row2, col1, etc.

```
proc tabulate data=ds1&year
           format=comma&cellwid..
           formchar='8283485868788898a8b8c'x;
  class
    %*** include row/column/page variables in
         the class statement. Note that if no
         row1 variable is requested, the row2
         variable is ignored. If there is no
         row1 or col1 variable requested, an
         invalid parameter message is provided
         to the user in the data validation
         section of the program.;
    %if &row1 ne NONE %then %do;
        &row1
      %if &row2 ne NONE %then %do;
          &row2
      %end;
    %end;
    %if &col1 ne NONE %then %do;
        &col1
      %if &col2 ne NONE %then %do;
          &col2
      %end;
    %end;
    %if &page1 ne NONE %then %do;
        &page1
    %end;
    %*** end of the class statement;
       ;
    var onecnst
    %if &nvar ne NONE %then %do;
      %*** additional numeric variable
           for which statistics will be
           displayed;
      &nvar
    %end;
    %*** end of the var statement;
           ;
  weight cpswgt;
  table
    %if &page1 ne NONE %then %do;
      (all='Totals' &page1),
    %end;
    %if &row1 ne NONE %then %do;
      %*** row variables are nested -
           show totals for each variable;
      %if &rnest = NEST %then %do;
        (all='Totals' &row1)
        %if &row2 ne NONE %then %do;
          *(all='Totals' &row2)
        %end;
      %end;
```

```
      %*** row variables are concatenated -
           show totals for first variable
           only;
      %else %if &rnest = CONCAT %then %do;
         (all='Totals' &row1
        %if &row2 ne NONE %then %do;
                              &row2
        %end;
                                  )
      %end;
      %*** comma between the tables row and
           column dimensions;
                                   ,
    %end;
    %if &col1 ne NONE %then %do;
      %*** column variables are nested -
           show totals for each variable;
      %if &cnest = NEST %then %do;
           (all='Totals' &col1)*
        %if &col2 ne NONE %then %do;
           (all='Totals' &col2)*
        %end;
      %end;
      %*** column variables are concatenated -
           show totals for first variable
           only;
      %else %if &cnest = CONCAT %then %do;
           (all='Totals' &col1
        %if &col2 ne NONE %then %do;
                               &col2
        %end;
                                    )*
      %end;
    %end;
   %*** display the requested statistics;
   (
    %*** statistics based on the onecnst
         variable (sum, percent, unweighted
         record count);
    %if &sum = YES or &pctg1 ne NONE or
        &ncount = YES %then %do;
      onecnst=' '*(
      %if &sum = YES %then %do;
                SUM
      %end;
      %if &pct1 ne NONE %then %do;
                PCTSUM<&pct1 all>*f=pctfmt.
      %end;
      %if &ncount = YES %then %do;
                N
      %end;
      %*** close parenthesis for the onecnst
           variable;
                 )
    %end;
    %*** statistics for the optional numeric
         variable;
    %if &nvar ne NONE %then %do;
       (&nvar*(&mystats))
    %end;
   %*** close parenthesis for the statistics;
   )
  ;
run;
```

SAS® HTML-FORMATTING MACRO CODE

Include the following statement prior to running the PROC tabulate to initiate the capture of the output:

```
%tab2htm(capture=on);
```

Below is simplified code for converting the output of the PROC tabulate to an HTML-table. Note that excellent documentation of the HTML-formatting macros is included on the SAS® web site.

```
%tab2htm(capture=off,         runmode=s,
         htmlfref=_webout,    encode=n,
         cspace=0,            tbbgcolr=white,
         clbgcolr=#ffeed4,    rlbgcolr=#ffeed4,
         bgtype=color,        bg=#d2ffff,
         bxcolor=red,         dvalign=BOTTOM,
         brtitle=CPS Data Table);
```

CONCLUSION

To provide the CPS Support System's users with the available functionality, the Census Bureau had to invest the resources to develop an elaborate macro. But it allows Census Bureau analysts (including those with no programming experience) to be more productive by empowering them to quickly and easily generate complex tables with eye-appeal. It supports their statistical review and research functions. It is used to generate ad-hoc tables for other Census Bureau users and to produce a subset of the special tabulation requests from users outside the Census Bureau.

From the SAS® programmer perspective, SAS/IntrNet™ provides a mechanism for developing interactive web pages primarily using skills we have previously developed. What is new is fairly easy to learn.

The most complex part of the CPS Support System is the custom-designed SAS® program. This is coded entirely in base SAS® code. The level of complexity of the code is determined by the complexity of your data and functionality that you provide your users. SAS/IntrNet™ is not the cause of the complexity of the CPS Support System; this complexity would also have been required had it been designed as a stand-alone macro to be run from the SAS® Display Manager System.

REFERENCES

DISCLAIMER

This paper is not an official report of the Census Bureau and does not necessarily reflect the views of the U.S. government or the Census Bureau. Responsibility for any errors remains with the author.

CONTACT INFORMATION

Stephen Heacock
U.S. Census Bureau
Housing and Household Economic Statistics Division
Mail Stop 8500
Washington, DC 20233

Work Phone: (301) 457-3204
Fax: (301) 457-3499

Email: stephen.j.heacock@ccmail.census.gov