Using Subroutines and Loops to Simplify Submissions Involving Several Datasets

When doing comparative research, it is natural to want to run the same routine across several different countries or across different waves. This approach can be simplified by the use of macros, subprograms/subroutines, and loops. These are easy ways to repeat program commands without having to retype them each time.

Activity

Compare the median (weighted) labour earnings for the United States and Mexico (2000), and the United Kingdom (1999) by looking separately at gross wage earnings (***pgwage***), net wage earnings (***pnwage***) and self-employment earnings (***pself***). Your sample should only include those with positive labour earnings (from either paid or self-employment) who are 16 years of age or older. For this comparison, you should write a subroutine to run for each dataset.

Now repeat the same analysis for the five new Latin American datasets (BR06, CO04, GT06, PE04 and UY04). In order to shorten the code, loop the subroutine created above over the 5 datasets.

Use the information from your output to answer the following questions:

1. Which country had the highest median wages from employment?

   _____

   _____

   _____

   _____

2. Are the income variables analysed above expressed in net or gross terms (i.e. before or after deduction of income taxes and mandatory social contributions)?

   _____

   _____

   _____

   _____

Guidelines

- For this comparison, proceed according to the following steps:
  - use a macro to define the variables you want to use;
  - use a subprogram to define a subroutine to calculate the estimates;
  - within the subroutine, i) use a where clause to select the adults for whom the relevant income measure is positive (`page > 15`) and (`incomemeasure>0`) on which you calculate your estimates and ii) find the median gross and net wage earnings and self-employment income for the subset of persons.
- An introduction to SAS macros
  - Macros allow you to substitute text in a program. A SAS program can contain any number of macros, and you can invoke a macro any number of times in a single program. The simplest way to repeat a series of commands several times in SAS, is to write those commands within SAS macro using the statement:

```
%MACRO yourmacroname <(yourparameter)> ;
 … your code …
%MEND <yourmacroname>;
```

  - A macro (*yourmacroname*) name must be a SAS name. Do not use macro reserved words as a macro name. The parameter-list (*<yourparameter>*) names one or more local macro variables whose values you specify when you invoke the macro. Parameters are local to the macro that defines them. You must supply each parameter name. A parameter list can contain any number of macro parameters separated by commas.
  - In order to invoke a macro statement do: `%yourmacroname <(yourparameter)>.` A macro definition must precede the invocation of that macro in your code.
  - Any parameter of a SAS macro as well as any macro-variable within a SAS program are preceded by an &. For instance:

```
%MACRO multi(pi);
  TITLE "pays: " &pi;
…
```

- The LIS "file names" (`&uk99p, &us00p`, etc.) are actually global macros that represent the full path of the data set you wish to use. By using macros instead of the full path names, it saves the user a lot of typing and also allows LIS to move and reorganize files without making the user update their programs. As LIS needs an `&` for declaring datasets, use the following code to open a dataset (`SET` statement) where the country code is a parameter of your SAS program:
`SET &&&pi.p`.
  - For more information about whether the LIS datasets report gross or net values, go to *Luxembourg Income Study (LIS)* → *List of net income datasets* (under the heading Information by Country).

- A simple way to get the median of an variable is to use the SAS procedure `PROC MEANS`:

```
PROC MEANS DATA=dataset MEDIAN MAXDEC = 0;
```

- You can control the output format of the estimates by applying the option **MAXDEC = n** .
- You can select the subset to perform your calculation using the SAS **WHERE** clause (that can be used for almost any SAS procedures) using the following statements:

  WHERE ((page > 15) AND (&var > 0)); where **&var** is a reference to the income measure on which you calculate the median

➢ You will need to keep exactly the same variables for each of the five datasets. Rather than repeating the list of variables each time you open a new dataset, you can assign to SAS macro variables the names of those variables and dataset names. Try to simplify your code for the second part of the exercise by using the following tips:

- You can then write a loop code to repeat the same commands for multiple data sets and to loop over variables as well using a combination of the **%DO %UNTIL, %SCAN(argument, n,<delimiters>) and %EVAL** macro functions. In the example below, SAS will loop (and run) five times your code. Since macro variables are strings, you need to use the **%EVAL** function to implicitly convert your index **(&i)** into a numeric variable on which you can perform calculation.

```
%LET I = 1;
%DO %UNTIL (&I > 5);
… yourcode …
  %LET I = %eval(&i+1);
%END;
```

- While using the %SCAN function SAS will read the nth argument of a list of name separated by a delimiter (by default a blank character). In the example below the first time the loop is invoked, SAS will use the string br06 within your program and so on …

```
%LET pi  = br06 co04 gt06 pe04 uy04;
%DO %UNTIL (&I > 5);
  …%scan(&pi,&i) … ;
  %LET I = %eval(&i+1);
%END;
```

Program

```
OPTIONS NOSOURCE NONOTES NOFMTERR NODATE NOCENTER LABEL NONUMBER LS=MAX
PS=MAX;

%MACRO param (pi,var) ;
   PROC MEANS DATA=&&&pi.p MEDIAN MAXDEC=0 ;
     WHERE ((page > 15) AND (&var > 0)) ;
     VAR &var ;
     WEIGHT pweight ;
   RUN ;
%MEND param ;

%param(uk99,pgwage)
%param(uk99,pnwage)
%param(uk99,pself)
%param(us00,pnwage)
%param(us00,pgwage)
%param(us00,pself)
%param(mx00,pgwage)
%param(mx00,pnwage)
%param(mx00,pself)


%LET var = pnwage pgwage pself ;
%LET pi  = br06 co04 gt06 pe04 uy04 ;
%MACRO single ;
%LET i = 1 ;
%DO %UNTIL (&i > 5) ;
      %LET country = %scan(&pi,&i) ;
      %LET j = 1 ;
      %DO %UNTIL( &j > 3) ;

            TITLE "Country :   %scan(&pi,&i)   -   %scan(&var,&j) " ;
            PROC MEANS DATA=&&&country.p MEDIAN MAXDEC=0 ;
              WHERE ((page > 15) AND (%scan(&var,&j)> 0)) ;
                  VAR %scan(&var,&j) ;
              WEIGHT pweight ;
            RUN ;

            %LET j = %eval(&j+1) ;
      %END ;
      %LET i = %eval(&i+1) ;
%END ;
%MEND single ;

%single
```

Results

	Gross wage earnings	Net wage earnings	Self-employment earnings
UK99	*13,513*	*10,628*	*10,400*
US00	*25,000*	*NA*	*12,500*
MX00	*NA*	*27,400*	*14,400*
BR06	*5,865*	*NA*	*6,000*
CO04	*4,320,000*	*NA*	*2,400,000*
GT06	*12,000*	*NA*	*6,000*
PE04	*NA*	*5,986*	*2,980*
UY04	*NA*	*54,042*	*42,577*

Answers to questions 1-2:

1. Country with the highest median wages from employment

   Because incomes are expressed in national currencies, it is not possible to compare values directly.

2. *Whereas it is obvious from the results above that MX00 contains only net incomes and US00 only gross incomes (as only one of the two wage variables is filled), for the UK99 dataset it is necessary to look at the documentation on-line to see that all other income variables are reported gross of taxes and contributions (with, in addition, the net wage variable).*

Comments

- Please note that all the results are given in nominal terms and in national currency. In order to make a direct comparison, convert these values to a common currency with the use of exchange rates (or PPPs) and deflators. LIS leaves it up to the researcher to choose the exchange rates and/or deflators that are best suited to his/her purpose.
- LIS detailed income variables are ideally filled with gross values (before taxes and mandatory social contributions are deducted), so that their overall sum (reported in summary income variable ***gi***) is equal to total gross income, from which taxes and contributions are subtracted to get to the final net disposable income figure (***dpi***). In some instances though, the original datasets only report net incomes: whereas total gross income is then not available (and the summary income variable ***gi*** is thus left empty), total final net disposable income figure is obtainable by aggregating all the net incomes (and this is exactly what ***dpi*** includes). In those cases, each LIS detailed income variable will contain net values instead of gross; only for wages there are two separate variables for those two amounts. For all other variables you need to take care when comparing across datasets that you are not comparing two different concepts of income.