Self-Teaching Package

Version 2012

STATA version - Part I

Part I

Inequality, poverty, and social policy

Overall Plan and Structure of the Exercise

The next seven exercises demonstrate the use of the LIS data. These exercises will lead you through the process of developing a comparative research project that examines inequality and poverty across countries. Each of the exercises introduces new concepts related to the datasets and the programming techniques needed to make use of the data. By the end of the last exercise, you will produce a complete program that returns results on poverty and inequality for a selection of six LIS countries.

Each exercise builds on the one that comes before it. It is intended that you will begin each new activity by returning to the program you have written in the previous exercise, and modifying it to satisfy the requirements of the new exercise. Each exercise contains questions that you can answer with the new results you produce. The solutions included with each exercise include an example program, with **bolded** sections indicating code that has been added for that exercise.

Unless otherwise instructed in the lesson, you should not remove variables or lines of code from your program, even if it is not needed for the current exercise. Variables and procedures that are introduced in earlier lessons may be needed later. In particular, all of the variables that are introduced in Exercise 2 will eventually be needed to complete the final exercise, even if they are not all required for the lessons in between.

The analysis shown here is simplified somewhat, compared to what might be done in an actual LIS Working Paper. Some choices have also been made in order to demonstrate particular aspects of the LIS data. However, these exercises provide a starting point for researchers who want to develop an analysis of the data based on their own research questions.

Research Questions

Since the beginning of the LIS project, one of the most prominent objects of research using the data has been the effect of government tax and transfer programs on poverty and inequality. The first substantive paper in the LIS Working Papers series, published in 1985, analyzed the pre- and post-transfer poverty rate in Sweden, the United Kingdom, Israel, the United States, Norway, Canada,

and West Germany. [1] The second substantive paper compares the distribution of income across these same seven countries.[2]

States affect the income distribution through several different types of policy, which are captured in the LIS data:

- Progressive taxation
- Social insurance programmes linked to employment, such as public pensions, unemployment insurance, and sickness pay
- Universal benefits provided irrespective of employment, income or assets
- Social assistance benefits for especially needy individuals or households

The exercises in this section assess the impact of these policies in different countries on both poverty and income inequality. We will measure incomes due to labour market income, capital income, and private transfers, and compare this to incomes after income taxes and social insurance contributions as well as government transfers are accounted for. Within the category of government policies, we will separate the effect of payroll taxes, social insurance, and universal benefits on the one hand, and social assistance benefits on the other. We will measure the proportion of the population that is poor according to these different income measures, where poverty is defined relative to median level of income within a country. We will also compare income inequality using one of the most popular and longstanding inequality measures, the Gini coefficient.

As LIS has grown, the analysis of government policy, poverty, and inequality has been updated for more countries and more recent years.[3] Recently, LIS expanded beyond the rich countries that have long made up the core of the project, and began adding data from middle income countries. This gives researchers the opportunity to compare income and poverty in these countries to the patterns seen in rich countries that have been much more heavily studied.

For this exercise, we will begin by analyzing the data from Guatemala, one of the recently-added

---

1 Richard Hauser, Lee Rainwater, Martin Rein, Gaston Schaber, Timothy Smeeding. "Poverty in Major Industrialized Countries". LIS Working Paper No. 2 - Jul 1985. http://www.lisdatacenter.org/wps/liswps/2.pdf

2 Michael O'Higgins, Gunther Schmaus, Geoffrey Stephenson. "Income Distribution and Redistribution". LIS Working Paper No. 3 - Jun 1985. http://www.lisdatacenter.org/wps/liswps/3.pdf

3 For a recent example, see Timothy Smeeding, "Government Programs and Social Outcomes: The United States in Comparative Perspective". LIS Working Paper No. 426 - May 2005.http://www.lisdatacenter.org/wps/liswps/426.pdf

middle income countries. We will then compare Guatemala to five other countries: the United States, Denmark, Poland, Slovenia, and Israel. These specific countries have been chosen in part for pedagogical reasons. However, they will also produce substantively interesting comparisons, because they represent a wide range of national income levels and welfare state regime types. After completing the final exercise, you will be able to answer the following questions:

- Which of the six countries in the study have the highest levels of income inequality and poverty before taxes and transfers are accounted for? Does this change when taxes and transfers are included?

- Which type of government policy has a larger impact on inequality and poverty in each country: taxes, social insurance, and universal benefits, or targeted social assistance?

Contents

1. Accessing the LIS Database: the Job Submission Interface (JSI)

Goal

This exercise introduces the Job Submission Interface (JSI), which we will be using to work with LIS data in all of the subsequent exercises.

The JSI is a secure Java application that allows researchers to:

- write, submit and view job requests (and corresponding outputs);
- track the status of the job requests in process ('received', 'processing', 'set for review', 'refused', etc.); and
- access the history of all job requests ever sent.

In this exercise we will use the JSI to open a dataset and produce basic descriptive statistics.

Activity

Launch the Job Submission Interface (JSI) application and logon to it with your LIS account.

Submit a simple program to display descriptive statistics (number of valid observations, mean, minimum, maximum) for the household-level income variable **dhi**, for Guatemala 2006. **dhi**, or *disposable household income*, contains the total monetary and non-monetary current income for the household, net of income taxes and social security contributions. It is a harmonised variable that is available for all datasets.

Track the status of your job.

View the resulting listing.

Go to the Job Library window and discard the job; use the advanced search tool to get it back.

Question: How many observations are in this dataset?

Guidelines

- Once connected to the Job submission Interface, there are three main tasks that may be carried out:

1. Submit jobs through the Job Session window.
   - Select a project (LIS, LWS or LES).
   - Select a statistical package (SAS, SPSS or Stata).
   - When submitting a job (Job Session window), always add a subject line.
   - Write your code.
   - Click on the submit button.
2. Work with Today's Jobs (Today Jobs window.)
   - Watch the status of jobs currently sent to LISSY in the 'jobs in process' panel (top-left).
   - View the jobs returned by LISSY.
   - Click on a job in the 'jobs returned' panel (bottom-left).
   - Click on the 'view job' button.
   - Click on the 'job text' or 'listing' tabs, respectively, of the right panel to see the request and its output.
   - Re-submit a selected job by clicking on the 'edit in job submission' button at the bottom-right of the window.
3. Manage (view, clean and search) all job requests ever sent in the Job Library window.
   - View jobs sent over a specific time period.
   - Clean the library by discarding useless job requests ('discard' button).
   - Search jobs by keywords.
   - Re-submit a selected job by clicking on the 'edit in job submission' button at the bottom-right of the window.

➢ When you open a LIS dataset, use the correct file reference for the country/year you wish to use. For example:

```
use $gt06h
```

For more information about the syntax of country/year file reference, see the job submission instructions on the LIS web site (*Data Access → Job Submission*). For a list of available data sets and their 2digit country codes, go to:

Our Data → LIS Database → Documentation → List of Datasets.

➢ Stata reminder: to run descriptive statistics, use **summarize <varlist>** (abbreviated by **sum**).

➢ If you do not receive your job in the expected amount of time, it means that there is a long

queue of jobs on LISSY. In that case, resending your job, or sending several other ones while waiting to receive the first one, will only increase the queue and hence your waiting time. Remember to wait to get your results before sending a new job!

Program

```
use $gt06h, clear
sum dhi
```

Results

	Number of valid observations	Mean	Minimum	Maximum
dhi	13,664	39,468	-184,160	2,727,846

Question: How many observations are in this dataset?

- There are 13,664 observations in this dataset.

Comments

- It is important to pay close attention to sample sizes in order to make sure you have enough data to make stable estimates. When working with small datasets, or small sub-samples of datasets, always check the size of the sample underlying each statistic you have computed.
- As you can see in the results, the disposable household income **dhi** can be negative. This happens in cases where the data provider kept losses as such rather than applying bottom coding techniques. This typically happens with incomes from self-employment or capital income; in rare circumstances it happens that taxes are higher than gross income due to different income reference periods or miscalculation of taxes.

2. Sample selection and weighting

Goal

This exercise introduces the variables that we will be using in the rest of our analysis and concentrates on sample selection and the use of weights.

Sample selection - The final objective of this exercise is to compare incomes before and after government intervention. In order to be sure that the comparison is correct, users should ensure that they use exactly the same sample when calculating statistics for the pre- and post-government intervention. It is thus important to begin by selecting a sample which can be used for the entire analysis. For this reason, we will drop from the analysis not only cases which have missing values in the variable of interest to the specific statistic being calculated at each step, but all the cases that have a missing value in any of the variables of interest for the whole analysis.

Use of weights - Comparative researchers are typically interested in the characteristics of national populations, not the samples provided. It is very important to understand and use sample weights correctly in order to get representative results for the total underlying population. This exercise shows the differences in statistics between the unweighted sample and the weighted population.

Activity

As in the previous exercise, we will continue working with the Guatemala 2006 (GT06) data. Modify your previous program to select only the following variables in addition to **dhi**: household weight (**hpopwgt**), number of household members (**nhhmem**), gross or net income information (**grossnet**), factor income (**factor**), work-related insurance transfers (**hitsi**), universal benefit income (**hitsu**), assistance benefit income (**hitsa**), private transfers (**hitp**) and tax and social security contribution expenditures (**hxit**).

Now create an indicator variable that is equal to 1 if **dhi** or any of its income and expenditure sub-components is missing, and equal to zero otherwise. Produce a frequency table for this variable to see how many cases have missing data.

Then produce two different sets of descriptive statistics for the variables you have selected: unweighted, and weighted by person. For continuous variables, the statistics should include the number of observations, the mean, the median, the minimum and the maximum. For categorical variables, you should produce a frequency table.

Next, drop all cases for which the indicator variable is equal to 1 and produce again the set of

weighted descriptive statistics for all variables selected.

Question: What currency unit is used for the income variables in this dataset?

Question: What effect does applying the weights have on the median of disposable household income?

Question: What percentage of cases are being dropped from the dataset?

Question: In subsequent exercises, we will be comparing incomes before and after accounting for different kinds of government transfers. What is the difference between *work-related insurance transfers, universal benefits*, and *assistance benefits*?

Question: In subsequent exercises, we will separate *social assistance* transfers from social insurance/universal benefits. Based on the statistics you have produced, which type of transfer do you think has a larger effect on inequality and poverty in Guatemala?

Question: What specific government programs are included within the "social assistance" variable for Guatemala?

Question: How many different values does the variable **grossnet** take in this dataset? How might this variable be useful?

Guidelines

- To keep only the variables you will be using, use the **keep** command:

```
keep <varlist>
```

  This avoids unnecessary burden on the machine so that submitted jobs will run faster. For even more savings on space and time, combine the last two commands:

```
use <varlist> using $us04h
```

- Stata reminder: to run frequencies use the **tabulate** command (abbreviated by **tab**):

```
tab <varname>
```

- Stata reminder: a simple way to get to get both the median and the mean at once is by using the **summarize** command with the **detail** option (abbreviated by **de**), which also produces additional statistics including skewness, kurtosis, the four smallest and four largest values, and various percentiles:

```
sum <varlist>, de
```

- LIS Weights:
  - LIS records the household level weights in the variable **hpopwgt.** The person-level file contains a person-level weight variable, **ppopwgt.** When we are using the household file but want to weight by person rather than household, we can multiply the household weight by the number of household members, contained in the variable **nhhmem**.
  - Both of these are *inflated* weights. This means that for this dataset, the weight inflates to the total population in Guatemala in 2006. You can find the population size by looking at the weighted number of observations. There are also *normalized* weights, **hwgt** and **pwgt**, which can be useful when performing pooled analysis with multiple countries. These will be covered in a later exercise (see Ex. 8 in Part II).
- Stata reminder: to get weighted descriptive statistics with the **summarize** command, you need to add **[w=<varname>]** after the variable list, but before the options (if any). Your final command should look something like this:

```
sum <varlist> [w=<varname>], de
```

- Stata reminder: to run weighted frequencies you need to add **[aw=<varname>]** after the variable name, but before the options (if any). Note that the analytic weight **[aw]** must be specified in order to be able to use the **tabulate** command with non-integer weights:

```
tab <varname> [aw=<varname>]
```

Program

```
use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $gt06h, clear
gen miss_comp = 0
replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. | hxit==.
tab miss_comp
sum dhi factor hitsi hitsu hitp hxit hpopwgt, de
tab nhhmem
tab grossnet
sum dhi factor hitsi hitsu hitp hxit hpopwgt [w=hpopwgt*nhhmem], de
tab nhhmem [aw=hpopwgt*nhhmem]
tab grossnet [aw=hpopwgt*nhhmem]
drop if miss_comp==1
sum dhi factor hitsi hitsu hitp hxit hpopwgt [w=hpopwgt*nhhmem], de
tab nhhmem [aw=hpopwgt*nhhmem]
tab grossnet [aw=hpopwgt*nhhmem]
```

Results

miss_comp	number of observations	percent	cumulative percent
0	13,664	99.84	99.84
1	22	0.16	100
Total	13,686	100	

Unweighted, all cases

	number of observations	mean	median	minimum	maximum
hpopwgt	13,686	193.85	119	2	2,657
dhi	13,664	39468	25,388	−184,160	2,727,846
factor	13,664	35,863.51	20,722	0	3,865,192
hitsi	13,664	1,099.26	0	0	160,000
hitsu	13,664	48.34	0	0	46,800
hitsa	13,664	995.3	0	−756	129,120
hitp	13,664	3,653.03	0	0	468,400
hxit	13,664	2,191.45	0	0	1,137,346

hhmem	number of observations	percent	cumulative percent
1	646	4.72	4.72
2	1,345	9.83	14.55
3	2,004	14.64	29.19
4	2,376	17.36	46.55
5	2,277	16.64	63.19
6	1,738	12.7	75.89
7	1,218	8.9	84.79
8	856	6.25	91.04
9	538	3.93	94.97
10	284	2.08	97.05
11	183	1.34	98.39
12	93	0.68	99.06
13	50	0.37	99.43
14	44	0.32	99.75
15	20	0.15	99.9
16	6	0.04	99.94
17	3	0.02	99.96
18	1	0.01	99.97

19	2	0.01	99.99
21	2	0.01	100
Total	13,686	100	

grossnet	number of observations	percent	cumulative percent
[110] taxes and contributions collected	13,686	100	100
Total	13,686	100	

Weighted, all cases

	number of (weighted) observations	mean	median	minimum	maximum
hpopwgt	12,964,197	443.94	295	2	2,657
dhi	12,938,575	48,004.49	31,160	-184,160	2,727,846
factor	12,938,575	44,593.14	26,400	0	3,865,192
hitsi	12,938,575	1,211.93	0	0	160,000
hitsu	12,938,575	65.58	0	0	46,800
hitsa	12,938,575	1,191.59	300	-756	129,120
hitp	12,938,575	3,894.83	0	0	468,400
hxit	12,938,575	2,952.59	0	0	1,137,346

Weighted, missing income cases dropped

	number of (weighted) observations	mean	median	minimum	maximum
hpopwgt	12,938,575	443.71	295	2	2,657
dhi	12,938,575	48,004.49	31,160	-184,160	2,727,846
factor	12,938,575	44,593.14	26,400	0	3,865,192
hitsi	12,938,575	1,211.93	0	0	160,000
hitsu	12,938,575	65.58	0	0	46,800
hitsa	12,938,575	1,191.59	300	-756	129,120
hitp	12,938,575	3,894.83	0	0	468,400
hxit	12,938,575	2,952.59	0	0	1,137,346

Question: What currency unit is used for the income variables in this dataset?

- The currency unit in this dataset is the Guatemalan Quetzal. This information can be found on the LIS web site at *Our Data → Documentation → List of Dataset Information*

Question: What effect does applying the weights have on the median of disposable household income?

- The median of unweighted **dhi** is 25,388 Quetzals, while the median of weighted **dhi** is 31,160 Quetzals.

Question: What percentage of cases are being dropped from the dataset?

- There are 22 cases with missing data, or 0.16 percent of the total number of cases in the dataset. Note that you should always be careful if you see a large amount of missing data, as it could bias your estimates.

Question: In subsequent exercises, we will be comparing incomes before and after accounting for different kinds of government transfers. What is the difference between *work-related insurance transfers, universal benefits*, and *assistance benefits* ?

- These concepts are defined in the *LIS Variable Definitions* document. Look in the tab marked "Current income", under the headings for variables ITSI, ITSU, and ITSA.

    - Work-related insurance transfers: Monetary transfers stemming from systems where the eligibility is based on the existence and/or the length of an employment relationship; in most cases the benefits are financed by contributions paid by employers, workers or both, and their amount is usually dependent on either the previous earnings or the previous contributions;

    - Universal benefits: Monetary transfers stemming from public programmes that provide flat-rate benefits to certain residents or citizens, provided that they are in a certain situation, but without consideration of income, employment or assets; note that in some cases the benefit amount may also depend on the other incomes of the individuals, which at the limit may result on some proportion of the population at the upper end of the income distribution to be excluded from receipt.

    - Assistance benefits: Monetary and non-monetary transfers stemming from public programmes that provide benefits especially targeted to needy individuals or households (i.e. with a strict income or assets test); the amount of the benefits is either flat rate or based on the difference between the recipient income and a standard amount

representing the minimum subsistence needs as guaranteed by the government.

Question: What specific government programs are included within the "social assistance" variable for Guatemala?

- If you look at the Institutional documentation for Guatemala 2006 on the LIS web site (*Our Data → LIS Database → By country → Guatemala → 2006* -under the heading Institutional Information), by selecting on the LIS variables pertaining to Social Assistance in the column "LIS Variables - Main set", you will see that the overall Social assistance variable for this dataset includes the following programmes: School transport allowance *(Bono de transporte escolar)*, study grants *(Becas escolares)*, scholarships from Child attention programme *(Programa de atención a la niña)*, School material kit *(Bolsa de útiles escolares)*, Food-related benefits from Social Assistance *(leche en polvo, vaso de leche, vaso de atol, alimentación escolar)*, Health programme from Social Assistance and donations of in-kind goods from public institutions.

Question: How many different values does the variable **grossnet** take in this dataset? How might this variable be useful?

- The variable **grossnet** has the same value for every case (110, "taxes and contributions collected".) By looking at this variable, you can learn what type of gross or net income information is contained in this variable, even without looking at the documentation. This variable will also be useful when working with multiple datasets that may contain either gross or net income.

3. Working with household income variables: top and bottom coding and equivalence scales

Goal

In order to compare incomes across countries, we need to make sure that our variables are fully comparable. In this exercise, you will apply top- and bottom-codes to remove extreme values. You will then create an *equivalised* income variable that adjusts for household size.

Top and bottom coding – Many inequality measures are sensitive to the values at the bottom and/or top of the income distribution, and some are not defined for non-positive values of income (e.g., any measure that calculates a logarithm). Applying top and bottom-codes (often referred to as 'winsorising') will avoid this problem, as well as ensuring comparability between datasets that may have originally had different top- and bottom-codings.

Equivalence scales – In order to get measures of poverty and/or income inequality in a population, it is necessary to compare income across different types of households. It is not logical to directly compare total household income between households of different sizes and composition.

Suppose you observe three levels of income (A, B, and C), where A>B>C. You cannot state that a household earning A is better off than one earning B unless you know the two households are similar in composition. For example, a family of four adult members receiving A is not necessarily better off than a couple with two children who receive B, and the family receiving B may not be better off than the childless couple receiving C.

For this reason, total household income needs to be adjusted to make it comparable across different households. This exercise gives one example of "equalising" households using one specific equivalence scale.

Activity

Keep the code from your previous exercise that you used to drop cases with missing data.

Create a new variable, **dhi_tb**, a top- and bottom-coded version of **dhi**. Bottom-code by setting all values less than zero to zero. Top-code by setting all values greater than ten times the median of **dhi** to ten times the median of **dhi**.

Now, create another new variable, **edhi_tb**, an equivalised version of top- and bottom-coded

disposable household income. We will correct for household size by applying the "LIS equivalence scale" (i.e., the square root of the number of household members).

Next, create a measure of per-capita income, **pcdhi_tb**, by dividing household income (un-equivalised, but top- and bottom-coded) by the number of household members.

Produce summary statistics showing the mean, median, minimum, and maximum of the four income variables: **dhi**, **dhi_tb**, **edhi_tb**, and **pcdhi_tb**.

Using the results returned from LISSY, fill out the following table:

	Household income (no top or bottom codes)	Household income	Per capita income	Equivalised income
Mean				
Median				
Minimum				
Maximum				

Question: Which of these four versions of the income variable contain *negative* values?

Question: Relative to household income and per capita income, how large are the mean and median of equivalised income?

Question: How does applying the top and bottom code affect the mean and median of household income?

Guidelines

- As we continue to build up to our final program, some of the code from the previous exercises will no longer be necessary. (For example, the code that produced the summary statistics in the previous exercise). You can choose to delete this code from your program in order to make it shorter. However, if you would like to keep a line code but stop it from being executed, simply place a * before it.
- This exercise does not use all of the variables that were used in the previous section. You should continue to keep all of those variables when you open the dataset, however, because they will be needed in future exercises.
- To equivalise income, divide the total household income by the value of the equivalence scale for each observation. To generate LIS equivalised income:

```
gen edhi_tb = dhi_tb /(nhhmem^0.5)
```

- Be careful when using weights. Make sure that the weight matches your unit of analysis. Weigh by **hpopwgt** for variables which are intrinsically at the household level (e.g., **dhi**) and by **hpopwgt*nhhmem** (to account for household size) for variables that are conceptually meaningful at the person level (e.g., per capita and equivalised income).

- In order to recall any of the results calculated by the **summarize** (or other) command(s), use the return codes automatically created by Stata. After you have summarized a variable, you can call **r(mean)** for the mean; **r(p50)** for the median; **r(p10)** for the first decile; **r(p20)** for the second decile, and so on. (In order to find out what statistics are available for each command, you can look in the Stata manuals, available online. To find the Stata help for a particular command, go to http://www.stata.com/help.cgi?*<command>*.) As a result, you can create a new variable based on the saved results of another variable used in a previous command in the following way:

```
sum <varname1>, de

gen <varname2> = r(p50)*10 if <varname1> >= r(p50)*10
```

- If you need information from a command, but do not need to see the results, precede your command by "**quietly**" (abbreviated by **qui**). This can save you from potentially lengthy log files that could be sent to the manual review queue by LISSY.

Program

```
use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $gt06h, clear
gen miss_comp = 0
quietly replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. | hxit==.
quietly drop if miss_comp==1
gen dhi_tb = dhi
quietly sum dhi [w=hpopwgt], de
*Apply top and bottom codes
replace dhi_tb = 0 if dhi<0
replace dhi_tb = 10*r(p50) if dhi>10*r(p50)
gen edhi_tb = dhi_tb/(nhhmem^0.5)
gen pcdhi_tb = dhi_tb/nhhmem
sum dhi dhi_tb [w=hpopwgt], de
sum edhi_tb pcdhi_tb [w=hpopwgt*nhhmem], de
```

Results:

	Household income (no top or bottom codes)	Household income	Per capita income	Equivalised income
Mean	46,570	44,741	9,153	20,007
Median	29,520	29,520	5,567	13,321
Minimum	-184,160	0	0	0
Maximum	2,727,846	295,200	289,510	289,510

Question: Which of these four versions of the income variable contain *negative* values?

- Only the income variable without top or bottom codes contains negative values, which are removed by applying a bottom-code. This will be important in the next exercise on inequality. The measure of inequality we will be using, the Gini coefficient, does not allow negative values. Removing negative values also allows for the commonly-used logarithmic transformation of income.

Question: Relative to household income and per capita income, how large are the mean and median of equivalised income?

- The mean and median values for equivalised income fall between those for household income and those for per capita income. The equivalising formula of **dhi**/(**nhhmem**^0.5) is a compromise between assigning all individuals their household income (**dhi**/**nhhmem**^0) and assigning them a per capita income (**dhi**/**nhhmem**^1).

Question: How does applying the top and bottom code affect the mean and median of equivalised income?

- Applying top and bottom codes makes the mean lower but does not affect the median. Means can be very sensitive to extreme values, so median values are often preferred as a measure of central tendency.

4. Inequality: the Gini Index

Goal

This exercise introduces the Gini index, which is one of the most commonly used income inequality indicators. We will be using the Gini coefficient as our measure of inequality in subsequent exercises, in order to compare inequality across countries and across different concepts of income.

Activity

Calculate the Gini index on total disposable income for Guatemala in 2006, using variables created in the previous exercise. Start with your program from the previous exercise, which will drop observations with missing data, apply top and bottom codes, and create variables containing equivalised disposable income and disposable income per capita.

Calculate the Gini coefficient for the winsorised (or bottom- and top-coded) versions of household income, per capita income, and equivalised income, and fill out the following table:

	Household income	Per capita income	Equivalised income
Gini coefficient			

Question Which shows greater inequality: household income, per capita income or equivalised income? What does this suggest about the possible relationship between income and household size?

Guidelines

- Stata provides ado files that will calculate the Gini coefficient as well as several other inequality indices. One such command is:

  ```
  ineqdec0 [varname] [w=<weight>]
  ```

  Note that this command, **ineqdec0**, ends in a zero, and is different from the command **ineqdeco**, ending in the letter "o". **ineqdec0** includes zero values when it calculates the Gini coefficient, which is the result we want for this exercise.

Program

```
use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $gt06h, clear

gen miss_comp = 0

quietly replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. |
hxit==.

quietly drop if miss_comp==1

gen dhi_tb = dhi

quietly sum dhi [w=hpopwgt], de

*Apply top and bottom codes

replace dhi_tb = 0 if dhi<0

replace dhi_tb = 10*r(p50) if dhi>10*r(p50)

gen edhi_tb = dhi_tb/(nhhmem^0.5)

gen pcdhi_tb = dhi_tb/nhhmem

ineqdec0 dhi_tb [w=hpopwgt]

ineqdec0 pcdhi_tb [w=hpopwgt*nhhmem]

ineqdec0 edhi_tb [w=hpopwgt*nhhmem]
```

Results

	Household income	Per capita income	Equivalised income
Gini coefficient	0.495	0.527	0.489

Question: Which shows greater inequality: household income, per capita income or equivalised income? What does this suggest about the possible relationship between income and household size?

- The Gini is lower for equivalised income, than the Gini for household income, which in turn is lower than the Gini for per-capita income. This reflects the fact that poorer households in Guatemala tend to be larger than richer households. Because the LIS equivalence scale assumes some economies of scale in large households, it produces a lower estimate of inequality than a per-capita measure.

5. Relative poverty rates

Goal

In order to get any measure of poverty, it is essential to make some assumptions concerning the criteria based on which to define poverty. The approach used by LIS (and most commonly adopted in the literature), is that of creating a relative poverty line based on the level and distribution of household disposable (equivalised) income in the total population. Households are classified as poor or non-poor on the basis of whether their income is lower or higher than the relative poverty line.

Once poor households are identified, you can create an indicator to help identify the proportion of poor households (or individuals) and to measure the level of poverty. The choice of the indicator used will mainly depend on the purpose of the research. In this exercise, we will calculate the relative poverty rates of households and individuals in the Guatemala 2006 data.

Activity

Add code to your program to produce an indicator for poverty. Define the poverty line as 50% of the median equivalised income. Calculate both the percentage of households in poverty and the head count ratio (defined as the percentage of individuals living in poor households), and complete the following table.

	Households	Individuals
Relative poverty rate		

Question: Are there more poor *households* or more poor *individuals?* What can you infer from this?

Guidelines

- From this point forward, we will be working exclusively with equivalised income, so the sections of your code relating to per-capita income can now be commented out or removed. The code for producing the Gini coefficient of equivalised income is not needed for this exercise, but will be required again in the next exercise.
- You can create an indicator variable indicating that an individual is poor (**poor** = 0 or = 1). The mean of this variable will give you the proportion in poverty.

- Whether this is the proportion of *individuals* or *households* in poverty depends on which weighting you use. Use **hpopwgt** if you want to measure household poverty, and **hpopwgt*nhhmem** if you are interested in individual poverty. If you use the individual-level weighting, you will produce the Head Count Ratio (HCR), which is the percentage of poor individuals in the total population.

- Stata reminder: to get the median equivalised income, you can use the command:

```
sum edhi_tb [w=hpopwgt*nhhmem], de
```

  The median will be stored in **r(p50)**.

- Stata reminder: to create a dummy variable which takes the value of 1 if a certain condition is satisfied, you can simply generate a new variable equal to the condition using the Stata command **generate** (abbreviated by **gen**). If the condition is satisfied, the variable takes the value 1, otherwise it takes the value 0. Your final command should look something like this:

```
gen byte <newvarname> = <condition>
```

Program

```
use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $gt06h, clear
gen miss_comp = 0
quietly replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. | hxit==.
quietly drop if miss_comp==1
gen dhi_tb = dhi
quietly sum dhi [w=hpopwgt], de
*Apply top and bottom codes
replace dhi_tb = 0 if dhi<0
replace dhi_tb = 10*r(p50) if dhi>10*r(p50)
gen edhi_tb = dhi_tb/(nhhmem^0.5)
sum edhi_tb [w=hpopwgt*nhhmem], de
gen byte poor=(edhi_tb<r(p50)*0.5)
sum poor [w=hpopwgt*nhhmem]
sum poor [w=hpopwgt]
```

Results

	Households	Individuals
Relative poverty rate	23.5%	24.9%

Question: Are there more poor *households* or more poor *individuals?* What can you infer from this?

- There is a greater proportion of poor individuals than poor households. This is because poor households are larger on average than non-poor households. This is another reason why the use of the equivalence scale is important.

Comments

- The head count ratio (HCR) measures poverty incidence (i.e., the number or proportion of poor people), but gives every person equal weight no matter how far they fall from the poverty line.
- Another measure, the Income Gap Ratio (IGR) measures poverty intensity or depth (how poor are the poor), but one poor person with an income of an amount x counts the same as two poor people each with an income of x/2. That is, the IGR measures the average income gap, but not its distribution among the poor).
- There are many other indicators of poverty that may be useful for different purposes. These include, among the most common, the whole family of Foster-Greer-Thorbecke indicators (of which the HCR is only one), the Sen index, the Takayama index, the Clark index, and the Thon index. It is important to note that a country may score better in comparison to a second country when using a particular index, but could score worse if another index was used instead.

6. Comparing income concepts

Goal

Now that we have calculated Gini coefficients and poverty rates based on disposable income, we can easily apply this same code to two other income concepts: income before any taxes and government transfers, and income after taxes, social insurance, and universal benefit transfers (but before social assistance transfers). Starting from this exercise, we will also introduce some programming techniques which will make it easier to repeat a series of commands several times without having to repeat the code.

Different income concepts – The income variable we have been working with, **dhi**, combines multiple income and expenditure flows. It is the sum of labour and capital income, private transfers, work-related insurance transfers, universal benefits, and social assistance benefits, minus any taxes and social insurance contributions paid. We will now define two new concepts of income. One of them is income before *any* government redistribution. The second is income *after* taxes, social insurance, and universal benefits, but *before* social assistance is included.

By calculating the Gini coefficient and the poverty rate using each of these three income concepts (income before government intervention, income after non-assistance government redistribution, and income after all government redistribution, i.e. our original disposable household income variable, **dhi**), we gain some insight into the effect of government programmes on inequality and poverty.

Efficient programming techniques – This exercise also introduces some programming techniques that allow to loop the same code over several variables.

Activity

As always, begin with the program you developed for the last exercise. Modify it to create two new income variables. The first, **mi**, is the sum of factor income (**factor**) and private transfers (**hitp**). Because we are specifically interested in the role of *government* transfers, we add private transfers to our measure of "market income" from labor and capital.

The second, **siti**, adds **mi** together with social insurance transfers (**hitsi**) and universal benefits (**hitsu**), while subtracting taxes and social contributions paid (**hxit**).

The income variable we have been using up to now, disposable household income, adds together the

variables contained in **siti** along with social assistance transfers (which are also contained in the variable **hitsa**).

Make sure you apply top codes, bottom codes, and the equivalence scale to the new variables, producing the final variables **emi_tb, esiti_tb,** and **edhi_tb.** You should apply the same topcode value of ten times median **dhi** to both of the other two variables.

Write a loop to calculate the Gini coefficient and the poverty rate for all three income variables, based on the code from the previous two exercises. Use it to fill out the table below. Make sure you use the *same* poverty line for all three income variables. That is, the poverty line should be defined as 50 percent of the median equivalised disposable household income, and that same poverty definition should be applied to the other two income variables.

	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
Gini coefficient			
Poverty rate			

Question: What has the greater impact on inequality and poverty in Guatemala: taxes/social insurance/universal benefits, or social assistance?

Guidelines

- An introduction about macros in Stata

  Macros are names (up to 31 characters) that can stand for strings, program-defined results, or user-defined values. A local macro exists only within the program that defines it, and cannot be referred to in another program. To refer to the contents of a local macro, place the macro name within left and right single quotes (`` `localname' ``). Global macros are similar to local macros, but once defined, they remain in memory and can be used by other programs. To refer to a global macro's contents, we preface the macro name with a dollar sign (**`$globalname`**).

  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 reorganise files without making users update their programs.

- Using macros in this analysis

  We want to use the same cut-point for the top and bottom code for each of the three income

variables. Rather than calculate the top-code value repeatedly, we can store it in a global macro by using the Stata command **global**. This macro assigns strings to specified global macro names, so that, once it has been created, each time the macro name is typed (preceded by the dollar sign) during the Stata session, Stata reads the string which was assigned to it. When you create a global macro, you will write **global <mname> "<varlist>":**

```
qui sum dhi [w=hpopwgt]
global topline = 10*r(p50)
```

When you recall it later on, you will write **$<mname>**.We can insert references to these macros when we apply the top and bottom codes:

```
replace edhi_tb = $topline if dhi>$topline
```

- Stata reminder on the **foreach** loop
  - Writing the **foreach** loop: you can use the **foreach** command to repeat the same commands for multiple variables (in the next exercise, we will loop over data sets as well). The **foreach** statement must be immediately followed on the same line by an open bracket (**{**), while the closing bracket (**}**) has to be on the last line (after the series of commands) on its own:

```
foreach <loopname> in <arguments over which to loop> {
   <commands over several lines>
}
```

  - Calling the arguments within a **foreach** loop: in the **foreach** statement, the **<loopname>** is a local macro that represents the list that follows the word **in**. and hence can be called with the standard way for local macros (**`<loopname>'**). For example, the following loop would apply an equivalence scale and top/bottom codes to the three different income variables:

```
foreach var in mi siti dhi {
  gen e`var'_tb = `var'
  replace e`var'_tb = 0 if `var'<0
  replace e`var'_tb = $topline if `var'>$topline
  replace e`var'_tb = (e`var'_tb/(nhhmem^0.5))
}
```

Program

```
use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $gt06h, clear

gen miss_comp = 0

quietly replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. |
hxit==.

quietly drop if miss_comp==1

quietly sum dhi [w=hpopwgt], de

global topline = 10*r(p50)

gen mi = factor + hitp

gen siti = factor + hitp + hitsi + hitsu - hxit

foreach var in mi siti dhi {

      gen e`var'_tb = `var'

      replace e`var'_tb = 0 if `var'<0

      replace e`var'_tb = $topline if `var'>$topline

      replace e`var'_tb = (e`var'_tb/(nhhmem^0.5))

}

quietly sum edhi_tb [w=hpopwgt*nhhmem], de

global povline = r(p50)*0.5

foreach var in mi siti dhi {

      quietly gen byte poor`var'=(e`var'_tb<$povline)

      ineqdec0 e`var'_tb [w=hpopwgt*nhhmem]

      sum poor`var' [w=hpopwgt*nhhmem]

}
```

Results

	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
Gini coefficient	0.514	0.502	0.489
Poverty rate	27.7%	27%	24.9%

Question: What has the greater impact on inequality and poverty in Guatemala: taxes/social insurance/universal benefits, or social assistance?

- For both inequality and poverty, assistance benefits play a larger role than social insurance and universal benefits.

Comments

- When comparing incomes before and after taxes and transfers, take care in interpreting the meaning of the pre-tax and transfer figure. It is tempting to interpret this number as a representation of the income distribution that would exist in the absence of government programs. However, since outcomes in the private sector are conditioned by the presence or absence of government programs, this is not generally a reasonable inference.

7. Comparing multiple countries

Goal

Now that we have written code to compute all of our statistics of interest, it is time to calculate these quantities for multiple countries. Building up on the previous exercise, we will introduce different programming techniques that break the code into logic sub-routines and generalise our program to loop through multiple datasets.

Adding more countries - Before adding a new country/year to an analysis, it is important to check that the dataset in question has all information necessary for the analysis you are performing. In this case, one should carefully check that the income sub-components variables used in the previous exercise are filled for all the new datasets to be introduced (and if not, whether the analysis can be slightly modified to take into account a different situation).

Efficient programming techniques - In the previous exercise we have introduced some programming techniques that allowed to loop the same code over several variables. In this exercise, we will introduce some other techniques that allow to organise the code in an efficient way and easily loop over both variables and datasets.

Activity

Take the code from the previous exercise and modify it so that it loops through six datasets: Guatemala 2006 (**gt06**), United States 2004 (**us04**), Denmark 2004 (**dk04**), Poland 2004 (**pl04**), Slovenia 2004 (**si04**) and Israel 2005 (**il05**).

The code that creates the income variables can be placed in a subroutine that is called from the main loop, as can the code that applies the equivalence scale and the top and bottom codes.

One of these countries is a net income dataset. This means that the tax contribution variable **hxit** is missing for all cases. Because our current program drops cases that are missing any income subcomponent variable, all the cases will be dropped for net income datasets. Using the **grossnet** variable, you can modify your code to prevent this from happening.

Use your results to fill in the following tables:

Gini Coefficient

Dataset	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
GT06			
US04			
DK04			
PL04			
SI04			
IL05			

Poverty Rate

Dataset	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
GT06			
US04			
DK04			
PL04			
SI04			
IL05			

Keep in mind that even if all cells can technically be constructed, the result may not necessarily be comparable conceptually! Think carefully about whether the dataset you are looking at contains the necessary information to calculate the quantity in each column.

Question: In what cells does the figure you produced not match the income concept described in the column header?

Question: Comparing Guatemala and Poland, which country has higher inequality before taxes and transfers? After?

Question: Which country has the highest poverty rate *before* taxes and transfers? After?

Question: In which country do government programmes do the most to reduce inequality and poverty, in percentage terms? In which country do they do the least?

Question: In which countries do social assistance benefits do more to reduce poverty than social insurance and taxes?

Guidelines

- Looping through datasets

In the last exercise, we created a loop that performs the same recoding on multiple variables. We can put that loop inside another loop that repeats the same analysis on multiple countries. When opening multiple datasets in a loop, you can refer to the dataset name using a macro:

```
foreach ccyy in <list of datasets to use> {
      use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet
          using $`ccyy'h, clear
}
```

Make sure to include the **clear** option, so that any dataset currently in memory is replaced with the new one.

In order to make your output more readable, you can use the **display** command to print information in the log file. When placed inside the loop, the following command will display the dataset that the loop is currently processing:

```
display "`ccyy'"
```

- Programs

The simplest way to repeat a series of commands several times in Stata, is to write those commands within a Stata program using the commands **program define *<progname>*** before the first of the commands of the series, and **end** after the last one, and then type the program name each time you want to run the series of commands.

```
program define <progname>
      <series of commands>
end
```

- Breaking your code into programs

Programs are also a useful way to separate the data-preparation parts of your code from the sections which produce output. A suggestion for moving parts of your code into programs is to create a program that performs all of the recoding and variable creation, which is then called from the main loop. The rest of the main loop will then contain only the code that produces our desired output. Since all of the data processing code is in a program, you can

easily suppress unnecessary output by calling the program with **quietly**.

The variable **grossnet** reports whether the incomes in a dataset are *gross* income, before taxes, or whether the only report post-tax values. Within a single dataset, all cases will have the same value for this variable. In a purely net income dataset, **grossnet** will be between 200 and 299. For more information about whether the LIS datasets report gross or net values, go to *Our Data* → *Documentation* (under the LIS DATABASE heading) → *List of Dataset Information*.

You can refer to **grossnet** in your loop in order to tell Stata to perform different calculations for net income datasets. For example, this code will prevent our program from dropping all observations in net income datasets:

```
gen miss_comp = 0

replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==.
  | hxit==. & inrange(grossnet,100,199)

replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==.
  & inrange(grossnet,200,299)
```

Before opening a new dataset, you need to drop the macro containing the top coding threshold and the poverty line. You can do that by placing the following command inside your loop:

```
macro topline povline
```

If you do not include this line, the program will give an error when it tries to define a macro that already exists.

Program

```
program define make_variables
  gen miss_comp = 0
  replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. |  hxit==. & inrange(grossnet,100,199)
  replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. & grossnet==200
  drop if miss_comp==1
  sum dhi [w=hpopwgt], de
  global topline = 10*r(p50)
  gen mi = factor + hitp
  gen siti = factor + hitp + hitsi + hitsu - hxit
  *For net income datasets, omit tax variable from second income measure
  replace siti = factor + hitp + hitsi + hitsu if grossnet==200
  foreach var in mi siti dhi {
    gen e`var'_tb = `var'
    replace e`var'_tb = 0 if `var'<0
    replace e`var'_tb = $topline if `var'>$topline
    replace e`var'_tb = (e`var'_tb/(nhhmem^0.5))
  }
  quietly sum edhi_tb [w=hpopwgt*nhhmem], de
  global povline = r(p50)*0.5
end

foreach ccyy in gt06 us04 dk04 pl04 si04 il05 {
  di "`ccyy'"
  use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $`ccyy'h,
  clear
  quietly make_variables
```

```
  foreach var in mi siti dhi {
    quietly gen byte poor`var'=(e`var'_tb<$povline)
    ineqdec0 e`var'_tb [w=hpopwgt*nhhmem]
    sum poor`var' [w=hpopwgt*nhhmem]
  }
  macro drop topline povline
}
```

Results

Gini Coefficient

Dataset	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
GT06	0.514	0.502	0.489
US04	0.492	0.397	0.374
DK04	0.447	0.264	0.228
PL04	0.531	0.327	0.315
SI04	*0.397**	0.235	0.231
IL05	0.518	0.394	0.369

*Calculation based on post-tax income

Poverty Rate

Dataset	Before taxes and government transfers	After taxes, social insurance, and universal benefits	After taxes and all transfers
GT06	27.7	27	24.9
US04	28	21.3	17.4
DK04	29.2	13.4	5.6
PL04	42	13.2	10.8
SI04	*30.4**	7.7	7.1
IL05	32.8	22.5	19.4

*Calculation based on post-tax income

Question: In what cells does the figure you produced not match the income concept described in the column header?

- Because Slovenia 2004 is a net income dataset, the Gini before taxes and transfers cannot be included. While the example program below does produce a result for Slovenia, it is not actually comparable to the other two countries because it is post-tax. That cell should therefore be left blank.

Question: Comparing Guatemala and Poland, which country has higher inequality before taxes and transfers? After?

- Poland has higher inequality before taxes and transfers are accounted for. After taxes and transfers, however, inequality is substantially higher in the Guatemala.

Question: Which country has the highest poverty rate *before* taxes and transfers? After?

- Poland has the highest poverty rate before taxes and transfers, but Guatemala has the highest poverty rate after taxes and transfers.

Question: In which country do government programmes do the most to reduce inequality and poverty, in percentage terms? In which country do they do the least?

- Government programmes have the largest impact on in Denmark, where they reduce the Gini coefficient by 46 percent, and poverty by 80 percent.

Question: In which countries do social assistance benefits do more to reduce poverty than social insurance and taxes?

- In Guatemala, social assistance reduces poverty more than taxes and social insurance.

Comments

- The datasets in these exercises were chosen because they allow social assistance to be separated from other kinds of government transfers. In many datasets, unfortunately, this separation is not possible due to the limitations of the original data. In such cases, the total amount of transfer income will be contained in a higher-level variable such as **hit**, and lower level variables such as **hitsi** and **hitsu** will be unfilled. You can consult the ≪ Variable Availability Matrix ≫ document to determine which variables are available in each dataset.

- In datasets containing gross incomes, the **grossnet** variable can take values of 100, 110, or 120. In datasets containing only net income, **grossnet** will always take a value of 200. Note, however, that a few datasets have **grossnet** codes of 300, 310, or 320, because they contain a mixture of gross and net incomes. See the "Variable Definition List" document for more information.

8. Producing compact and concise output

Goal

The program we have developed produces results for two indicators (poverty and inequality), three definitions of income, and six countries. This results in a total of 36 values of interest in the resulting log file. We could copy these values into a spreadsheet by hand, but this would be very time-consuming and would increase the likelihood of introducing errors by accidentally copying the wrong number. In this exercise, we will modify the program to create compact output that can be transferred into a spreadsheet with only one cut-and-paste.

The final output will be a set of comma-separated values, in which each country is on a separate row and each indicator is on a separate column.

Activity

To produce easy-to-use output, three additional modifications need to be made to the program:

- Add code to store the values we want in macros, so that they can be output later.
- Suppress all unnecessary output so that it will not appear in the log
- Add code to print out the values we have stored as macros in the form of a comma-separated list.

Transfer your results into tables like the ones shown below. The easiest way to do this is to copy and paste the comma-separated block of results into a spreadsheet.

Guidelines

Remember that you can put **quietly** before any command to suppress its output.

We are using the functions **ineqdec0** and **sum** to produce the Gini coefficient and the poverty rate. Both of these functions store their results in temporary macros. The **ineqdec0** command stores the gini in **r(gini)**, and **sum** stores the mean (which is equivalent to the proportion in poverty) in **r(mean)**. We can suppress the normal output of these functions and store the needed results in macros instead. For example, within the loop in our existing program we can add:

```
quietly ineqdec0 e`var'_tb [w=hpopwgt*nhhmem]
local gini`var'  = r(gini)
```

The **`var'** refers to the current income variable being used by the loop. If the program is currently computing values for **dhi**, it will save the Gini of disposable household income in the macro *ginidhi*.

The above code will produce results with many unnecessary decimal places. Using the **round**

command will produce rounded-off values that are easier to read:

```
local gini`var' = round(r(gini), .001)
```

This example rounds to three decimal places; replacing ".001" with ".0001" would round to four decimal places.

The **display** command (or **di** for short) can be used to output the values we have stored in macros. If the name of the macro is enclosed by the characters ` and ', Stata will replace the name of the macro with its value when producing the output.

```
di "`ccyy',`ginimi',`ginisiti',`ginidhi',`povratemi',`povratesiti',`povratedhi'"
```

By placing this line inside your main loop, a new line of comma-separated values will appear in the log for each dataset.

You can tell the program to create a set of column headers before the first line of data with a line like this:

```
if "`ccyy'" == "gt06" di "dataset,gini_mi,gini_siti,gini_di,povrate_mi,povrate_siti,povrate_di"
```

Program

```
program define make_variables


  gen miss_comp = 0
 replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. | hxit==. & inrange(grossnet,100,199)
  replace miss_comp=1 if dhi==. | factor==. | hitsi==. | hitsu==. | hitp==. & grossnet==200


  drop if miss_comp==1


  sum dhi [w=hpopwgt], de
  global topline = 10*r(p50)


  gen mi = factor + hitp
  gen siti = factor + hitp + hitsi + hitsu - hxit
  *For net income datasets, omit tax variable from second income measure
  replace siti = factor + hitp + hitsi + hitsu if grossnet==200


  foreach var in mi siti dhi {


    gen e`var'_tb = `var'


    replace e`var'_tb = 0 if `var'<0
    replace e`var'_tb = $topline if `var'>$topline


    replace e`var'_tb = (e`var'_tb/(nhhmem^0.5))


  }


  quietly sum edhi_tb [w=hpopwgt*nhhmem], de
  global povline = r(p50)*0.5
```

```
end

foreach ccyy in gt06 us04 dk04 pl04 si04 il05 {

  quietly use dhi factor hitsi hitsu hitp hxit hpopwgt nhhmem grossnet using $`ccyy'h, clear

  quietly make_variables

  foreach var in mi siti dhi {

    quietly gen byte poor`var'=(e`var'_tb<$povline)

    *Calculate and store gini, relative poverty rate
    quietly ineqdec0 e`var'_tb [w=hpopwgt*nhhmem]
    local gini`var' = round(r(gini), .001)
    quietly sum poor`var' [w=hpopwgt*nhhmem]
    local povrate`var' = round(100*r(mean),.1)

    *For net income datasets, blank out market income measures, since these cannot be
calculated correctly
    if "`var'" == "mi" & grossnet==200 {
      local gini`var' =.
      local povrate`var' =.
    }

  }

  *Output gini and poverty rate measures as a comma separated list. If this is the first
country being
  *computed, output a line of column headers first.
  if "`ccyy'" == "gt06" di "dataset,gini_mi,gini_siti,gini_dhi,povrate_mi,povrate_siti,povrate_dhi"
```

```
  di "`ccyy',`ginimi',`ginisiti',`ginidhi',`povratemi',`povratesiti',`povratedhi'"

  macro drop topline povline

}
```

Results

If you have modified the program correctly, you should see a block of text like this at the end of your log file:

dataset,gini_mi,gini_siti,gini_di,povrate_mi,povrate_siti,povrate_di

gt06,.514,.502,.489,27.7,27,24.9

pl04,.531,.327,.315,42,13.2,10.8

si04,.,.235,.231,.,7.7,7.1

us04,.492,.397,.374,28,21.3,17.4

dk04,.447,.264,.228,29.2,13.4,5.6

il05,.518,.394,.369,32.8,22.5,19.4

If you copy and paste this into a spreadsheet, most spreadsheet programs should recognize this as a set of comma-separated values and parse it automatically. You can also copy the text into a text file, save it with the **.csv** extension, and open it with your spreadsheet program.

If the lines are interrupted with Stata messages, make sure you have put **quietly** before every command that produces unwanted output.