Self-Teaching Package

Version 2016

R 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

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

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 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:

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

- 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 valid observations (non-missing) are in this dataset for **dhi**?

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, Stata, or R).
    - 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:

```
df ← read.LIS('gt06p')
```

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:

http://www.lisdatacenter.org/data-access/lissy/job-submission/using-r-on-the-lissy-system/

✓ R reminders

- To access a variable within a data frame, you can use either the syntax `df$<variable>` or `df[['<variable>']]`
- You can get basic descriptive statistics for a variable (such as the minimum, maximum, mean and median) using `summary(<variable>)`. The option `digits` is needed to ensure that the display precision is great enough, `summary(<variable>, digits=<#>)`.

- ✓ To get the total number of non-missing observations of a variable, you can use `sum(!is.na(<variable>))`. This creates a vector which is TRUE whenever the variable is not missing, and then gives you the sum of that vector, i.e., the number of non-missing observations.
- ✓ 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

```
df <- read.LIS('gt06h')
print(summary(df$dhi, digits=10))
print(sum(!is.na(df$dhi)))
```

Results

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

Question: How many valid observations (non-missing) are in this dataset for **dhi**?

- There are 13,664 valid observations.

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**).

Produce two different sets of descriptive statistics for the variables you have selected:

- For continuous variables, the statistics - unweighted, and weighted by person - 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 **dhi** or any of its income and expenditure sub-components is missing And see how many cases have missing data

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 is 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

- ✓ Note to R users: R is a very flexible language, and there are typically many ways to code the same operation. The tips in these lessons suggest one method of achieving some of the exercises goals, but if you are familiar with other techniques you should feel free to use them.

  Note that for these early lessons, which concentrate on simple descriptive statistics, R may be somewhat more cumbersome to use than other statistical packages which are designed to make such simple estimates very easy to retrieve. This is in contrast to the later lessons, where R's power and flexibility make it possible to produce more complex estimates very quickly and compactly.

- ✓ Use of weights in basic descriptive is not include in R base but R packages, such as **Hmisc**, include many functions useful for working with weighted data. Anyway, to avoid using packages you may not be familiar with, we prefer to create simple functions from scratch. For instance here is a simple function to generate a weighted mean.

```
wmean <- function(x, weight) {
  y     <- x[which(!is.na(x))]
  wgt   <- weight[which(!is.na(x))]
  wmean <- sum(y*wgt/sum(wgt))
  return(wmean)
}
```

  Note that this function is not robust in the sense that it does not test for missing arguments when it is called and … so forth. But you may want to re-use and improve it for later your own coding purpose

- ✓ Use the option "labels=FALSE" with `read.LIS` to return only numeric codes rather than value labels, which will make recoding easier later on.

- ✓ You can use the “vars” argument to the **read.LIS** function to keep only certain variables (columns):
  - `df <- read.LIS(‘gt06’, labels=FALSE, vars=<varlist>)`
- ✓ This avoids unnecessary burden on the machine so that submitted jobs will run faster. You can also use the “subset” to keep only certain observations. For example, to select only cases with non-missing household income:
  - `df <- read.LIS(‘gt06’, labels=FALSE, subset=”complete.cases(dhi)”)`

Program

```
wmean <- function(x, weight) {
  y     <- x[which(!is.na(x))]
  wgt   <- weight[which(!is.na(x))]
  wmean <- sum(y*wgt/sum(wgt))
  return(wmean)
}
wNtile <- function(var, wgt, split) {
  x  <- var[order(var)]
  y  <- wgt[order(var)]
  z  <- cumsum(y) / sum(y)
  cop  <- rep(NA,length(split))
  for (i in 1:length(cop)) {
    cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
  }
  return(cop)
}

vars <- c('dhi','factor','hitsi','hitsu','hitp','hxit','hpopwgt','nhhmem','grossnet')
df   <- read.LIS('gt06h', labels=FALSE, vars=vars)
print(row_total <- nrow(df))
print(row_drop  <- with(df, length(which((complete.cases(dhi,factor,hitsi,hitsu,hitp,hxit) ==
TRUE)))))
round(((row_total - row_drop) / row_total) * 100, digits = 2)

for (x in c('nhhmem','grossnet')) {
 cat(toupper(x))
 print(table(df[[x]], useNA = 'ifany'))
 print(paste(round(prop.table(table(df[[x]], useNA = 'ifany')) * 100, digits = 2), "%", sep =
""))
 cat(paste(" "), sep = '\n')
}
for (x in c('hpopwgt','dhi','factor','hitsi','hitsu','hitp','hxit')) {
 df1 <- df[!is.na(df[[x]]), ]
 print(c(toupper(x)))
 print(c(nb_obs = sum(!is.na(df1[[x]]))))
 print(summary(df1[,x], digits = 5))
 print(c(weighted_mean = round(wmean(df1[[x]], df1$hpopwgt*df1$nhhmem), digits = 0),
weighted_median = round(wNtile(df1[[x]], df1$hpopwgt * df1$nhhmem, split = 0.5), digits = 0)))
  cat(" ", sep = '\n')
}
```

Results

	number of observations	percent
Valid	13,664	99.84
Missing	22	0.162
Total	13,686	100

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	

Unweighted

	number of observations	mean	median	minimum	maximum
hpopwgt	13,686	193.85	119	2	2,657
dhi	13,664	39,468	25,388	-184,160	2,727,800
factor	13,664	35,864	20,722	0	3,865,200
hitsi	13,664	1,099.3	0	0	160,000
hitsu	13,664	48.34	0	0	46,800
hitp	13,664	3,653	0	0	468,400
hxit	13,664	2,191.5	0	0	1,137,300

Weighted

	mean	median
hpopwgt	444	295
dhi	48,004	31,160
factor	44,593	26,400
hitsi	1,212	0
hitsu	66	0
hitp	3,895	0
hxit	2,953	0

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, suggesting that low-income households are over-represented in the sample.

Question: What percentage of cases is 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 in-kind transfers of 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 **dhiT**, a top- and bottom-coded version of the original **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**, 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, **cdhi,** 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**, **dhiT**, **cdhi** , and **edhi**.

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:

```
df$edhi <- df$dhiT /(df$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).

Program

```
wmean <- function(x, weight= NULL) {
  if (is.null(weight))
    weight <- rep(1, length(x))
  y     <- x[which(!is.na(x))]
  wgt   <- weight[which(!is.na(x))]
  wmean <- sum(y*wgt/sum(wgt))
  return(wmean)
}
wNtile <- function(var, wgt = NULL, split) {
  if (is.null(wgt))
      wgt <- rep(1, length(var))
  x  <- var[order(var)]
  y  <- wgt[order(var)]
  z  <- cumsum(y) / sum(y)
  cop  <- rep(NA,length(split))
  for (i in 1:length(cop)) {
    cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
  }
  return(cop)
}
topBottom <- function(var, botline, topline) {
  tb                <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
setups <- function(data_file) {
  vars    <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitsa', 'hitp', 'hxit', 'hpopwgt', 'nhhmem',
'grossnet')
  subset  <- 'complete.cases(dhi, factor, hitsi, hitsu, hitsa, hitp, hxit)'
  df      <- read.LIS(data_file, labels=FALSE, vars=vars, subset=subset)
  botline <- 0
  topline <- 10 * wNtile(df$dhi, df$hpopwgt, 0.5)
  df$dhiT  <- topBottom(df$dhi, botline, topline)
  df$edhi <- df$dhi / df$nhhmem^0.5
  df$cdhi <- df$dhi / df$nhhmem
  return(df)
}

df <- setups('gt06h')
for (x in c('dhi', 'dhiT')) {
  cat(paste("VARIABLE: ", toupper(x), sep=""), sep = '\n')
  cat(paste("Average: " , format(round(wmean(df[[x]]    , df$hpopwgt)     , digits = 0), big.mark
= ",")), sep = '\n')
  cat(paste("Median : " , format(round(wNtile(df[[x]], df$hpopwgt, 0.5), digits = 0), big.mark =
",")), sep = '\n')
  cat(paste("Minimum: " , format(round(min(df[[x]]), digits = 0), big.mark = ",")), sep = '\n')
  cat(paste("Maximum: " , format(round(max(df[[x]]), digits = 0), big.mark = ",")), sep = '\n')
  cat(" ", sep = '\n')
}
for (x in c('cdhi', 'edhi')) {
  cat(paste("VARIABLE: ", toupper(x), sep=""), sep = '\n')
  cat(paste("Average: " , format(round(wmean(df[[x]]    , df$hpopwgt * df$nhhmem)     , digits =
0), big.mark = ",")), sep = '\n')
  cat(paste("Median : " , format(round(wNtile(df[[x]], df$hpopwgt * df$nhhmem, 0.5), digits = 0),
big.mark = ",")), sep = '\n')
  cat(paste("Minimum: " , format(round(min(df[[x]]), digits = 0), big.mark = ",")), sep = '\n')
  cat(paste("Maximum: " , format(round(max(df[[x]]), digits = 0), big.mark = ",")), sep = '\n')
  cat(" ", sep = '\n')
  }
```

Results:

	Household income (no top or bottom codes)	Household income	Per capita income	Equivalised income
Mean	46,576	44,745	9,157	20,010
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

- To make the most of R, it is advisable to break your code into *functions* as much as possible. In future exercises, we will adapt our program in order to make use of functions. For this exercise, we will provide a function definition that you can include in your code to produce weighted Gini coefficients:

```
gini <- function(x,weight) {
  ox     <- order(x)
  x      <- x[ox]
  weight <- weight[ox]/sum(weight)
  p      <- cumsum(weight)
  nu     <- cumsum(weight*x)
  n      <- length(nu)
  nu     <- nu/nu[n]
  res    <- sum(nu[-1]*p[-n])-sum(nu[-n]*p[-1])
return(res)
}
```

After it has been defined, you can call this function just as you would call any built-in R function, for example:

```
      gini(df$dhi, df$hpopwgt))
```

This function is derived from the **reldist** package, http://cran.r-project.org/web/packages/reldist/.

Program

```
gini <- function(x,weight) {
  ox     <- order(x)
  x      <- x[ox]
  weight <- weight[ox]/sum(weight)
  p      <- cumsum(weight)
  nu     <- cumsum(weight*x)
  n      <- length(nu)
  nu     <- nu/nu[n]
  res    <- sum(nu[-1]*p[-n])-sum(nu[-n]*p[-1])
return(res)
}
wNtile <- function(var, wgt, split) {
  x  <- var[order(var)]
  y  <- wgt[order(var)]
  z  <- cumsum(y) / sum(y)
  cop  <- rep(NA,length(split))
  for (i in 1:length(cop)) {
    cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
  }
  return(cop)
}
topBottom <- function(var, botline, topline) {
  tb                <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
setups <- function(data_file) {
  vars    <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitp', 'hxit', 'hpopwgt', 'nhhmem', 'grossnet')
  subset  <- 'complete.cases(dhi, factor, hitsi, hitsu, hitp, hxit)'
  df      <- read.LIS(data_file, labels=FALSE, vars=vars, subset=subset)
  botline <- 0
  topline <- 10 * wNtile(df$dhi, df$hpopwgt, 0.5)
  df$dhi  <- topBottom(df$dhi, botline, topline)
  df$edhi <- df$dhi / df$nhhmem^0.5
  df$cdhi <- df$dhi / df$nhhmem
  return(df)
}

  df <- setups('gt06h')
  round(gini(df$dhi , df$hpopwgt)          , digits = 3)
  round(gini(df$cdhi, df$hpopwgt*df$nhhmem), digits = 3)
  round(gini(df$edhi, df$hpopwgt*df$nhhmem), digits = 3)
```

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.

- In R, you can take the sum of a logical vector, and the result will the proportion of elements in that vector which have the value TRUE. This means that you can place a logical comparison inside a sum statement to produce a poverty rate, as in the following example:

```
sum((df$edhi < 0.5 * wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt)
/ sum(df$hpopwgt))
```

This code computes the weighted median income, sum the equivalised disposable income that is for those cases that are below half of that median, and then divides the result by the sum of the weights, which ends up to the percentage in relative poverty.

- Whether your poverty rate 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.

Program

```
wNtile <- function(var, wgt, split) {
  x  <- var[order(var)]
  y  <- wgt[order(var)]
  z  <- cumsum(y) / sum(y)
  cop  <- rep(NA,length(split))
  for (i in 1:length(cop)) {
    cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
  }
  return(cop)
}
topBottom <- function(var, botline, topline) {
  tb               <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
setups <- function(data_file) {
  vars    <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitp', 'hxit', 'hpopwgt', 'nhhmem', 'grossnet')
  subset  <- 'complete.cases(dhi, factor, hitsi, hitsu, hitp, hxit)'
  df      <- read.LIS(data_file, labels=FALSE, vars=vars, subset=subset)
  botline <- 0
  topline <- 10 * wNtile(df$dhi, df$hpopwgt, 0.5)
  df$dhi  <- topBottom(df$dhi, botline, topline)
  df$edhi <- df$dhi / df$nhhmem^0.5
  df$cdhi <- df$dhi / df$nhhmem
  return(df)
}
df <- setups('gt06h')
maxline <- 0.5
round(100 * (sum((df$edhi < maxline * wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt)
     / sum(df$hpopwgt)), digits = 2)
round(100 * (sum((df$edhi < maxline * wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt *
     df$nhhmem) / sum(df$hpopwgt * df$nhhmem)), digits = 2)
```

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, esiti,** and **edhi.** 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

-

Program

```
gini <- function(x, weight) {
  ox     <- order(x)
  x      <- x[ox]
  weight <- weight[ox]/sum(weight)
  p      <- cumsum(weight)
  nu     <- cumsum(weight*x)
  n      <- length(nu)
  nu     <- nu/nu[n]
  res    <- sum(nu[-1]*p[-n])-sum(nu[-n]*p[-1])
return(res)
}
wNtile <- function(var, wgt, split) {
    x  <- var[order(var)]
    y  <- wgt[order(var)]
    z  <- cumsum(y) / sum(y)
    cop  <- rep(NA,length(split))
    for (i in 1:length(cop)) {
        cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
    }
    return(cop)
}
topBottom <- function(var, botline, topline) {
  tb                <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
```

```
setups <- function(data_file) {

  vars     <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitp', 'hxit', 'hpopwgt', 'nhhmem',
'grossnet')

  subset   <- 'complete.cases(dhi, factor, hitsi, hitsu, hitp, hxit)'

  df       <- read.LIS(data_file, labels = FALSE, vars = vars, subset = subset)

  botline  <- 0

  topline  <- 10   * wNtile(df$dhi, df$hpopwgt, 0.5)

  df$dhi   <- topBottom(df$dhi, botline, topline)

  df$edhi  <- df$dhi / (df$nhhmem^0.5)

  df$mi    <- df$factor + df$hitp

  df$emi   <- topBottom(df$mi  , botline, topline) / (df$nhhmem^0.5)

  df$siti  <- df$factor + df$hitp + df$hitsi + df$hitsu - df$hxit

  df$esiti <- topBottom(df$siti, botline, topline) / (df$nhhmem^0.5)

  return(df)

}

df <- setups('gt06h')

maxline <- 0.5

for(var in c('emi', 'esiti', 'edhi')) {

  cat(paste("VARIABLE: ", var), sep = '\n')

  cat(paste("Gini Coefficient -" , round(gini(df[[var]], df$hpopwgt*df$nhhmem), digits
= 3)), sep = '\n')

  cat(paste("Poverty Rate     -", round(100 * (sum((df[[var]] < maxline *
wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt * df$nhhmem) /
sum(df$hpopwgt * df$nhhmem)), digits = 2)), sep = '\n')

 cat(" ", sep = '\n')

}
```

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**), Hungary 2005 (**hu05**) 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			
HU05			
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			
HU05			
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 plus universal benefits and taxes?

Guidelines

- Functions

  The R language lends itself to *functional programming*, in which programs are constructed around the inputs and outputs to functions. When a chunk of code needs to be executed repeatedly for different data, it can be useful to put that code in a function, and then have the main program call the function. We have already used functions in the calculation of the Gini coefficient. Since this exercise requires us to perform identical recodings on multiple data sets, we can create a function to do the recoding. The function will look like this:

  ```
  setups <- function(data_file) {

       <recoding commands from the previous program go here>

    return(df)

  }
  ```

  This function takes a data frame as it argument, performs some recoding on that data frame, and then returns the data frame. So, if you have loaded a dataset into memory, you can then recode it.

- 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 use **grossnet** in your loop in order to tell R to subset the data differently for net income datasets. For instance, you could pass the following string to read.LIS in the “subset” argument:

```
"complete.cases(dhi,factor,hitsi,hitsu,hitp) & (complete.cases(hxit) | grossnet == 200"
```

This will drop cases that are missing tax data (**hxit**) only in gross income data sets, and not in net income datasets (where this variable is always missing).

- You can use the **ifelse** command with **grossnet** to help in coding your income variables. The ifelse command is a vectorized form of **if**. It takes three vectors as arguments. It checks whether each element of the first vector is TRUE, and then returns either the element from the second vector (if it is TRUE) or the element from the second vector (if it is FALSE). For instance:

  ```
  ifelse(c(TRUE,FALSE,TRUE), c(1,1,1), c(2,2,2)))
  ```

would return the vector c(1,2,1). In your code, try the following to produce your income variable:

```
df$siti  <- ifelse(df$grossnet %in% 100:199,
                   df$factor + df$hitp + df$hitsi + df$hitsu - df$hxit,
                   df$factor + df$hitp + df$hitsi + df$hitsu)
```

- Looping through datasets

You can use a loop to iterate over datasets, just as you have used loops elsewhere in your code:

```
for(ccyy in datasets) {

       df <-setups(ccyy)

       <other code to produce output>

}
```

Program

```
gini <- function(x,weight) {
  ox     <- order(x)
  x      <- x[ox]
  weight <- weight[ox]/sum(weight)
  p      <- cumsum(weight)
  nu     <- cumsum(weight*x)
  n      <- length(nu)
  nu     <- nu/nu[n]
  res    <- sum(nu[-1]*p[-n])-sum(nu[-n]*p[-1])
return(res)
}
wNtile <- function(var, wgt, split) {
    x  <- var[order(var)]
    y  <- wgt[order(var)]
    z  <- cumsum(y) / sum(y)
    cop  <- rep(NA,length(split))
    for (i in 1:length(cop)) {
        cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
    }
    return(cop)
}
topBottom <- function(var, botline, topline) {
  tb               <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
setups <- function(data_file) {
  vars     <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitsa', 'hitp', 'hxit', 'hpopwgt',
'nhhmem', 'grossnet')
  subset <- 'complete.cases(dhi,factor,hitsi,hitsu,hitp) & (complete.cases(hxit) |
grossnet == 200)'
  df       <- read.LIS(data_file, labels = FALSE, vars = vars, subset = subset)
  botline  <- 0
  topline  <- 10   * wNtile(df$dhi, df$hpopwgt, 0.5)
  df$dhi   <- topBottom(df$dhi, botline, topline)
  df$edhi  <- df$dhi / (df$nhhmem^0.5)
  df$mi    <- df$factor + df$hitp
  df$emi   <- topBottom(df$mi   , botline, topline) / (df$nhhmem^0.5)
  df$siti  <- ifelse(df$grossnet %in% 100:199,  df$factor + df$hitp + df$hitsi +
df$hitsu - df$hxit, df$factor + df$hitp + df$hitsi + df$hitsu)

  df$esiti <- topBottom(df$siti, botline, topline) / (df$nhhmem^0.5)

  return(df)

}


  datasets <- c('gt06', 'us04', 'dk04', 'pl04', 'hu05', 'il05')

  maxline <- 0.5

  for (ccyy in datasets) {

    df <- setups(paste(ccyy,'h',sep=''))

    for(var in c('emi', 'esiti', 'edhi')) {

      cat(paste("VARIABLE: ", var), sep = '\n')

      cat(paste("Gini Coefficient -" , round(gini(df[[var]], df$hpopwgt*df$nhhmem),
digits = 3)), sep = '\n')

      cat(paste("Poverty Rate     -", round(100 * (sum((df[[var]] < maxline *
wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt * df$nhhmem) /
sum(df$hpopwgt * df$nhhmem)), digits = 2)), sep = '\n')

      cat(" ", sep = '\n')

    }

  }
```

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	0.487	0.389	0.364
DK04	0.447	0.264	0.228
PL04	0.531	0.336	0.315
HU05	*0.535**	0.316	0.289
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	14.7	10.8
HU05	*44.1**	12.2	7.4
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 Hungary 2005 is a net income dataset, the Gini before taxes and transfers cannot be included. While the example program below does produce a result for Hungary, 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 plus universal benefits and taxes?

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

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 **hits**, 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, we will modify the program as follows:

- Create a matrix to store only the results we need from the program, and label its dimensions appropriately.
- Replace the code that prints out results with code that stores those results in the matrix.
- Print out the matrix at the very end of the program, in Comma Separated Values format.

The result will be 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

One advantage of R is that it does not limit the number of separate data objects that can be in memory at one time. This means that we can have a separate matrix to hold our results, while also keeping a LIS data set open for processing. To set up the matrix, put this code before the program's main loop:

```
result           <- matrix(NA,length(datasets),6)
rownames(result) <- datasets
colnames(result) <- c('gini_mi','gini_siti','gini_dhi','pov_mi','pov_siti','pov_dhi')
```

This creates a 6x6 matrix filled with the NA (missing) value, which we will then fill. Inside your main loop, insert the Gini and poverty rate to the appropriate cells.

```
result[match(ccyy,datasets), match(var, c('emi','esiti','edhi'))] <- “your-gini”
```

```
result[match(ccyy,datasets), 3 + match(var, c('emi','esit','edhi'))] <- “your-pov”
```

By using the **match** command, we can ensure that each number is inserted into the correct column and row. Then at the very end of the program, *after* the main loop, we insert:

```
write.csv(result)
```

This will create output like that shown below in the section **Result**.

Program

```
gini <- function(x,weight) {
  ox     <- order(x)
  x      <- x[ox]
  weight <- weight[ox]/sum(weight)
  p      <- cumsum(weight)
  nu     <- cumsum(weight*x)
  n      <- length(nu)
  nu     <- nu/nu[n]
  res    <- sum(nu[-1]*p[-n])-sum(nu[-n]*p[-1])
return(res)
}
wNtile <- function(var, wgt, split) {
    x  <- var[order(var)]
    y  <- wgt[order(var)]
    z  <- cumsum(y) / sum(y)
    cop  <- rep(NA,length(split))
    for (i in 1:length(cop)) {
        cop[i] <- x[Find(function(h) z[h] > split[i], seq_along(z))]
    }
    return(cop)
}
topBottom <- function(var, botline, topline) {
  tb               <- ifelse(var < botline, botline, var)
  tb[tb > topline] <- topline
  return(tb)
}
setups <- function(data_file) {
  vars     <- c('dhi', 'factor', 'hitsi', 'hitsu', 'hitsa', 'hitp', 'hxit', 'hpopwgt', 'nhhmem',
'grossnet')
  subset <- 'complete.cases(dhi,factor,hitsi,hitsu,hitp) & (complete.cases(hxit) | grossnet ==
200)'
  df       <- read.LIS(data_file, labels = FALSE, vars = vars, subset = subset)
  botline  <- 0
  topline  <- 10   * wNtile(df$dhi, df$hpopwgt, 0.5)
  df$dhi   <- topBottom(df$dhi, botline, topline)
  df$edhi  <- df$dhi / (df$nhhmem^0.5)
  df$mi    <- df$factor + df$hitp
  df$emi   <- topBottom(df$mi  , botline, topline) / (df$nhhmem^0.5)
  df$siti  <- ifelse(df$grossnet %in% 100:199,  df$factor + df$hitp + df$hitsi + df$hitsu -
df$hxit, df$factor + df$hitp + df$hitsi + df$hitsu)
  df$esiti <- topBottom(df$siti, botline, topline) / (df$nhhmem^0.5)
  return(df)
}
```

```
datasets <- c('gt06', 'us04', 'dk04', 'pl04', 'hu05', 'il05')

maxline <- 0.5

result           <-  matrix(NA,length(datasets),6)

rownames(result) <- datasets

colnames(result) <- colnames(result) <-
c('gini_mi','gini_siti','gini_dhi','pov_mi','pov_siti','pov_dhi')


for (ccyy in datasets) {

    df <- setups(paste(ccyy,'h',sep=''))

    for(var in c('emi', 'esiti', 'edhi')) {

      result[match(ccyy, datasets),     match(var, c('emi', 'esiti', 'edhi'))] <-
round(gini(df[[var]], df$hpopwgt*df$nhhmem), digits = 3)

      result[match(ccyy, datasets), 3 + match(var, c('emi', 'esiti', 'edhi'))] <- round(100 *
(sum((df[[var]] < maxline * wNtile(df$edhi, df$hpopwgt * df$nhhmem, 0.5)) * df$hpopwgt *
df$nhhmem) / sum(df$hpopwgt * df$nhhmem)), digits = 2)

   }

}

print(write.csv(result))
```

Results

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

```
"","gini_mi","gini_siti","gini_dhi","pov_mi","pov_siti","pov_dhi"
"gt06",0.514,0.502,0.489,27.7,27,24.9
"us04",0.487,0.389,0.364,28,21.3,17.4
"dk04",0.447,0.264,0.228,29.2,13.4,5.6
"pl04",0.531,0.336,0.315,42,14.7,10.8
"hu05",0.535,0.316,0.289,44.1,12.2,7.4
"il05",0.518,0.394,0.369,32.8,22.5,19.4
```

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.