

Using PovMap2

A USER's GUIDE

Qinghua Zhao
Peter Lanjouw

The World Bank

Table of Contents

Preface

PART 1 INTRODUCTION

Introduction

Three stage of Poverty Mapping

Statistical model

Poverty/Inequality measurements

Brief history of Poverty Mapping Software

What is new in PovMap2

Workflow in Poverty Mapping

PART 2 CONCEPTS AND COMPONENTS

Data Array

Hierarchical ID

Relational among Variables without using Relational Database

Continues Variable vs. Categorical Variable

Operation to categorical Variable

Data Grid

File and their Types

PART 3 USING POVMAP2.0

Workflow and Navigator

Importing External Data

Create PovMap Project

Screen Checker: Finding Candidate Variable Pairs

- Matching and undo a matching

- Comparing Variables in Survey and in Census

- Summary Statistics of Selected Variable

- Creating Expression and Variable

- Using Categorical Variable

- Generating Compounded Variable

DRAFT for comment; not for citation

Aggregate census data and make it available in survey

Using Functions

Rules of Interlocking

Batch Processing with Script

Screen Consumption Model

Define per-capita household consumption

From Variable to Repressor

Analysis Provided

OLS

Forward Regression

Backward Regression

Stepwise Regression

Correlation

Means

Viewing Repressors

Testing for Over-fitting

Interactive model building

Importing from Existing Model

Testing for over-fitting problem

Testing for model structure change (F-test)

Screen Cluster Effect

Cluster Effect or No Cluster Effect

Distribution of Cluster Effect

Setting Outliers

Examine cluster effect table

Screen Household Effect Model

Definition of the dependent variable in alpha model

What is `_YHAT_` and Why

Special treatment if error term is too small

Does household effect exist?

Screen Idiosyncratic Effect

Analyzing Idiosyncratic Outputs

DRAFT for comment; not for citation

Distribution of Idiosyncratic Effect

Examine household effect table

Simulation setting

Poverty Line

Weighting with Household Size

Trimming

Yhat

Beta vector and Alpha vector

Cluster

Simulation setting

Number of replications

Initial random seed

Aggregation level

Request YDump and/or PDump

Partial Perturbation Method (Classical Method)

Result of Simulation

Output

Log File

YDump and PDump

Tools and Utilities

Viewing Data Array

Exporting Data Array

Subsetting Data Array

Appending Data Array

Changing ID

Project Setting

Checking ID matching

PART 4 SPECIAL TOPICS

Guideline for Modeling

Trimming

DRAFT for comment; not for citation

Random Seed

Cluster Effect at Higher Level

PART 5 APPENDIX

Function Details

Arithmetic functions

Logical expressions

Aggregation functions

Special functions

Recode

If

Truncation

Example of Using Functions

System Requirement

Updating Online

Getting help

FAQ

PART 1 INTRODUCTION

What is Poverty Mapping

"Poverty mapping" is a newly developed method to estimate the welfare level and the degree of inequality at lower aggregation levels such as township or ward. It uses a model of household expenditure from a survey dataset to estimate household welfare and apply it to a census dataset which does not include household expenditure or income information. Poverty indicators at the community level are then formed as aggregates.

Three stages of Poverty Mapping

Poverty Mapping consists of three stages. In the first stage, the census and survey data are examined for compatibility. Only the variables with same definition and distribution are allowed to be used in the second stage or the modeling stage. In the modeling stage, a series of regressions are run to model the expenditure and decompose the random unexplained components. Once a believable welfare estimation model is obtained, the poverty mapper will then apply it to the third stage known as the simulation stage. The simulation stage uses the model parameters and performs repeated drawings on different random components to bootstrap the household expenditure. The estimated household welfare is then aggregated on different levels.

Statistical model

Users of this manual should always refer to the paper by Elbers, Lanjouw and Lanjouw (2001) for theoretical background and statistical inference.

The computing of poverty mapping begins during the estimation of the expenditure function. For simplicity, we assume per capita expenditure of a household is the basic left hand side variable and the word 'cluster' is an aggregation level in the survey and census datasets.

$$(1) \quad \ln y_{ch} = E[\ln y_{ch} | \mathbf{x}_{ch}] + u_{ch}$$

where

c is the subscript for the cluster

h is the subscript for the household within cluster c .

y_{ch} is the per capita expenditure of household h in cluster c .

\mathbf{x}_{ch} is the household characteristics for household h in cluster c .

a linear approximation of model (1) is then written as:

$$(2) \quad \ln y_{ch} = \mathbf{x}_{ch}' \boldsymbol{\beta} + u_{ch} \quad (\text{also referred to as } \mathbf{Beta} \text{ model})$$

since survey data is just a sub sample of the whole population, the location information is not available for all regions in the census data. Thus, we cannot include the location variable in the survey model. Therefore, the residual of (2) must contains the location variance.

$$(3) \quad u_{ch} = \eta_c + \varepsilon_{ch}$$

Here η_c is the cluster component and ε_{ch} is the household component. As mentioned above, the estimate of η_c for each cluster in the census dataset is not applicable, therefore we must estimate the deviation of η_c . Taking the arithmetic expectation of (3) over cluster c

$$(4) \quad u_c = \eta_c + \varepsilon_c.$$

Hence

$$E[u_c^2] = \sigma_\eta^2 + \text{var}(\varepsilon_c) = \sigma_\eta^2 + \tau_c^2.$$

Assuming η_c and ε_{ch} are normally distributed and independent each other, Elbers et al gave a estimate of variance of the distribution of the locational effect η_c :

$$(5) \quad \text{var}(\hat{\sigma}_\eta^2) \approx \sum_c [a_c^2 \text{var}(u_c^2) + b_c^2 \text{var}(\hat{\tau}_c^2)] \approx \sum_c 2[a_c^2 \{(\hat{\sigma}_\eta^2)^2 + (\hat{\tau}_c^2)^2 + 2\hat{\sigma}_\eta^2 \hat{\tau}_c^2\} + b_c^2 \frac{(\hat{\tau}_c^2)^2}{n_c - 1}].$$

When the location effect η_c does not exist, equation (3) is reduced to $u_{ch} = \varepsilon_{ch}$.

According to Elbers et al, the remaining residual ε_{ch} can be fitted with a logistic model and will regress a transformed ε_{ch} on household characteristics:

$$(6) \quad \ln \left[\frac{e_{ch}^2}{A - e_{ch}^2} \right] = \mathbf{z}_{ch}^T \hat{\boldsymbol{\alpha}} + r_{ch}. \quad (\text{also referred to as } \mathbf{Alpha} \text{ model})$$

where A set to equal $1.05 * \max\{\varepsilon_{ch}^2\}$. The variance estimator for ε_{ch} can be solved as

$$(7) \quad \hat{\sigma}_{\varepsilon, ch}^2 = \left[\frac{AB}{1+B} \right] + \frac{1}{2} \widehat{\text{Var}}(r) \left[\frac{AB(1-B)}{(1+B)^3} \right].$$

The result from above indicates a violation of assumptions for using the OLS in model (2), so a GLS regression is needed. In GLS the variance-covariance matrix is a diagonal block matrix with structure:

$$(8) \quad \begin{bmatrix} \sigma_{\eta_c} + \sigma_\varepsilon & \sigma_\varepsilon & \sigma_\varepsilon & \sigma_\varepsilon \\ \sigma_\varepsilon & \sigma_{\eta_c} + \sigma_\varepsilon & \sigma_\varepsilon & \sigma_\varepsilon \\ \sigma_\varepsilon & \sigma_\varepsilon & \sigma_{\eta_c} + \sigma_\varepsilon & \sigma_\varepsilon \\ \sigma_\varepsilon & \sigma_\varepsilon & \sigma_\varepsilon & \sigma_{\eta_c} + \sigma_\varepsilon \end{bmatrix}$$

Overall, the procedure for stage 1 of the poverty mapping computation can be listed as:

- s1. estimate "Beta" model (2)
- s2. calculate the location effect η_c (3)
- s3. calculate the variance estimator $\text{var}(\sigma_\eta^2)$ (4)

- s4. prepare the residual term ε_{ch} for estimating "Alpha" model (6)
- s5. estimate GLS model with (8)
- s6. use a singular value decomposition to break down the variance-covariance matrix from previous step. This will be used for generating a vector of a normally distributed random variable such that the joint variance-covariance matrix will be in the form of (8)
- s7. read in census data, eliminate records containing missing values, generate all census variables needed for both *Beta* and *Alpha* models.
- s8. save all datasets needed for the simulation (the "PDA" file).

Bootstrapping

The fully specified simulation model is defined as follows:

$$(9) \quad \ln \tilde{y}_{ch} = \mathbf{x}_{ch}' \tilde{\beta} + \tilde{\eta}_c + \tilde{\varepsilon}_{ch}$$

where $\tilde{\beta} \sim N(\hat{\beta}, \hat{\Sigma}_\beta)$

$\tilde{\eta}_c$ is a random variable (could be normally distributed or T-distributed) with a variance defined in (5)

$\tilde{\varepsilon}_{ch}$ is a random variable (either normally distributed or T-distributed) with a variance defined in (7), $B = \exp(\tilde{Z}_{ch}' \tilde{\alpha})$ and $\tilde{\alpha} \sim N(\hat{\alpha}, \hat{\Sigma}_\alpha)$

Trimming could be applied to the random variable $\tilde{\eta}_c$ and $\tilde{\varepsilon}_{ch}$ as well as to random vector $\tilde{\beta}$ and $\tilde{\alpha}$. In the case of a normal distributed random variable, a range (-1.96, 1.96) will make 10% of random $--N(0,1)--$ drawing to be redrawn. For random vector of size m , the vector will be redrawn if the mode of the vector (a χ^2 distributed random variable) is outside the specified range.

Poverty/Inequality measurements

After estimating $\ln \tilde{y}_{ch}$, several poverty and inequality measures will be computed. They include Generalized Entropy class :

$$GE(\lambda) = \frac{1}{\lambda^2 - \lambda} \left[\frac{1}{W} \sum w_i \left(\frac{y_i}{\bar{y}} \right)^\lambda - 1 \right] \quad (\lambda \neq 0, \lambda \neq 1)$$

$$GE(0) = \frac{1}{W} \sum w_i \log \frac{y_i}{\bar{y}} \quad \text{and} \quad GE(1) = \frac{1}{W} \sum w_i \frac{y_i}{\bar{y}} \log \frac{y_i}{\bar{y}}$$

Atkinson class of measures :

$$A(c) = 1 - \left[\frac{1}{W} \sum w_i \left[\frac{y_i}{\bar{y}} \right]^{1-c} \right]^{\frac{1}{1-c}} \quad (c \neq 0, c \neq 1)$$

and Gini index:

$$Gini = \frac{W+1}{W-1} - \frac{2}{W(W-1)\bar{y}} \sum w_i y_i [\rho_i + 0.5(w_i - 1)] \quad \text{where} \quad \rho_{i+1} = \rho_i + w_i$$

In the above definitions, w_i is the weight of household i and W is the total population.

Brief history of Poverty Mapping Software

Creation of the software tool for Poverty mapping began in early 2000 when Peter Lanjouw completed a pilot program written in SAS. He used a partial perturbation method which was later called 'classical method'. In 2001 Gabriel Demombynes, along with Peter Lanjouw, Jenny Lanjouw and Chris Elber, developed another SAS version which used a simultaneous drawing method in bootstrapping. Due to the limitation of the SAS language, these nicely written SAS codes suffered greatly in their performance. It was soon clear that the SAS language was not adequate enough to bootstrap complicated larger datasets in practice. At this point Qinghua Zhao was delegated to devise an alternative. The focus then was purely on the bootstrapping module. By mid-2003, a usable package, version 1.1, was published and used in many countries. This version and its predecessor 1.2.4, have played an important role in poverty mapping activities all around the world. Version 1 can complete the simulation of 100 replications on 1 million observations in 4 to 5 minutes. However, all processes of stage 1 and 2 have to be done with other statistical packages and are very time consuming. At the same time, demand for poverty mapping rose in many countries and thus a package to complete all stages of poverty mapping seamlessly was needed. In response to this demand, PovMap2 was designed and the beta version was delivered in early 2006. Today, corrections and enhancement in PovMap2 are still taking place.

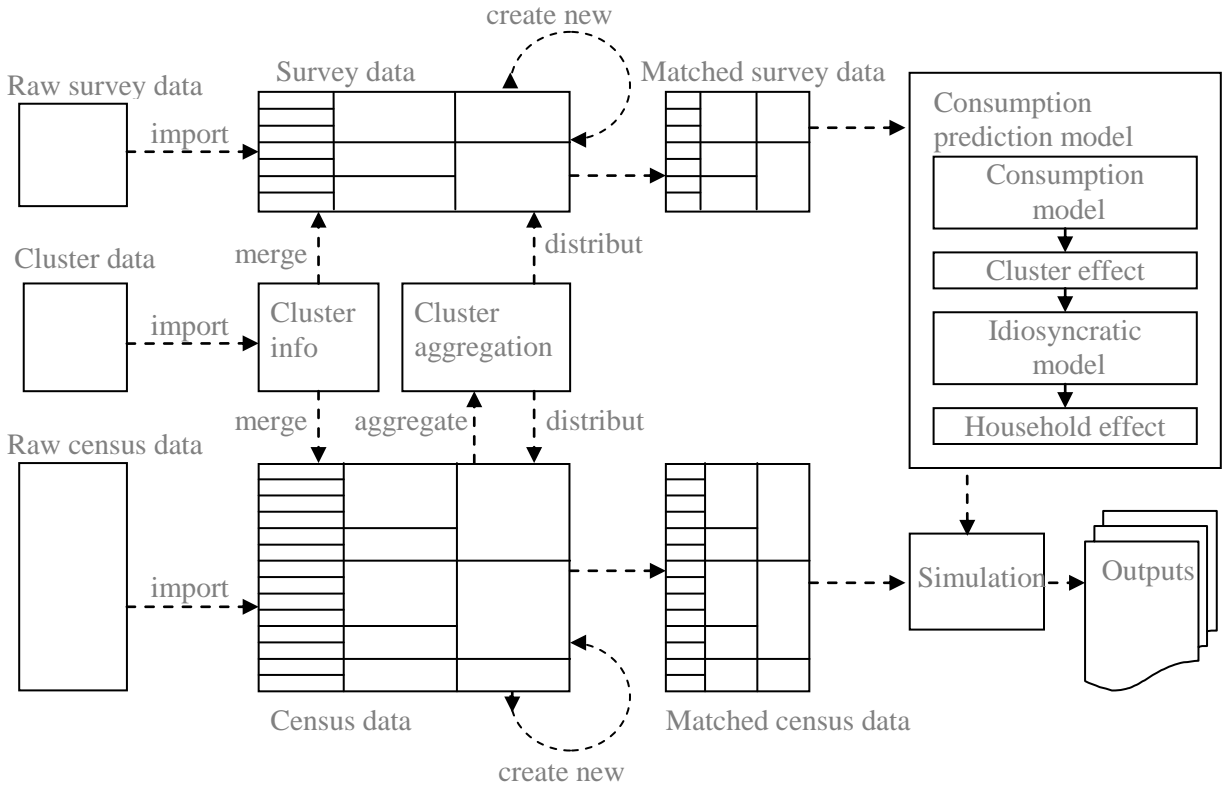
What's new in PovMap2

- A single platform for processing all computational needs in poverty mapping. Eliminating possible errors when using commercial statistical packages.

- Ability to read and process variables or formulas quickly. In this database the information of a record is not stored sequentially, which is very different from a traditional relational database. However, the concept of a record can still be used here. Our new database engine can read/write variables much more quickly.
- A new mixed mode data access is provided. It will allow users to mix household variables with the district aggregation in the same formula without first producing the district aggregation database.
- A correctly implemented mixed model variable access will also make it possible to store the variables of higher aggregation levels much more efficiently. This is crucial since large amounts of district level variables are needed in modeling.
- Introduction of categorical variables. A categorical variable is equivalent to a set of dummies. The provision of categorical variables eases the task of comparing them and/or its cross product. Limited operation allowed for the categorical variable insures the correctness of categorical variable manipulation. Adding, subtracting, or dividing two categorical variables is prohibited. While multiplication of two factors is interpreted and implemented as making new categorical variables of paired values. In contrast, a traditional database treats all numeric variables equally, whether or not they are categorical or continuous variables, thus mistakenly adding a categorical variable with a continuous variable is possible in most statistical packages or databases.
- Variable comparability is strictly enforced; only variables compatible in survey and census are allowed to enter the regression stage. This will greatly reduce potential errors.
- Since the regression uses only the record with no missing value, it is very important for researchers to know how many records will become unavailable as the result of data processing. It is quite difficult in traditional statistical software to produce such a report but users will be able to do it in PovMap2.
- While users are exploring the data within PovMap2, all the actions will be recorded in the form of a script log. The script log can be used or modified at a later date., or to be used in another dataset.
- PovMap2 has a content sensitive help system.
- Advanced data processing and tabulation function. User can use PovMap2 to finish all computation needs without switching to other software tools.

Dataflow in Poverty Mapping

The following chart illustrates the dataflow of poverty mapping. The majority of it is processed in the 'checker' screen of PovMap2. The box labeled "Consumption prediction model" consists of four screens which need to be handled sequentially.



PART 2 CONCEPTS AND COMPONENTS

Aggregation level and Hierarchical ID

A typical expenditure prediction model contains not only household information, but characteristics of villages and counties where the households reside. In order to link information of different levels into the household level, proper keys must be used throughout the data preparation stage. PovMap2 has adopted a compounded structure for ID, each section of the ID represents a different level of aggregation. For example, a hierarchical ID may look like

SSCCDDDDHHHH

Where SS is a two digits code for stratum, CC is two digits code for county within stratum SS, DDD is a three digits code for enumeration district within stratum SS and county CC, and the HHHH is the household ID within that district. We can construct the identifier for other levels by truncating this ID to different lengths. For example, SSCCDDD would be the district identifier, SSCC, the county identifier and SS is the stratum identifier. Please note that truncation of SSCCDDDDHHHH into SSCCD or SSCCDDDDHH may not provide correct level.

In PovMap2 the hierarchical ID is stored as a double precision number. Suppose the dataset (survey and census) has multiple identifiers such as STRATUM (ranged from 1 to 9), COUNTY (ranged from 1 to 99), DISTRICT (ranged from 1 to 125), VILLAGE (ranged from 1 to 38) and HOUSEHOLD (range from 1 to 23539). A compounded ID at district level can be created with

$$\text{DISTID} = (\text{STRATUM} * 100 + \text{COUNTY}) * 1000 + \text{DISTRICT},$$

or

$$\text{DISTID} = \text{STRATUM} * 100000 + \text{COUNTY} * 1000 + \text{DISTRICT}.$$

a compounded village ID could be defined as

$$\text{VID} = ((\text{STRATUM} * 100 + \text{COUNTY}) * 1000 + \text{DISTRICT}) * 100 + \text{VILLAGE}$$

Similarly, the household ID could be defined as

$$\text{HID} = (((\text{STRATUM} * 100 + \text{COUNTY}) * 1000 + \text{DISTRICT}) * 100 + \text{VILLAGE}) * 100000 + \text{HOUSEHOLD},$$

or

$$\text{HID} = \text{STRATUM} * 1000000000000 + \text{COUNTY} * 10000000000 + \text{DISTRICT} * 10000000 + \text{VILLAGE} * 100000 + \text{HOUSEHOLD},$$

It looks like

DRAFT for comment; not for citation

SCCDDDVVHHHHH

Users of PovMap2 can also use an automatic formation

STRATUM \ COUNTY \ DISTRICT \ VILLAGE \ HOUSEHOLD

to define the compounded ID. PovMap2 will first figure out the range of each identifier and then determine the correct multipliers to form an expression for the compounded identifier.

It is strongly recommended that users of PovMap2 use the same definition to construct the compounded ID for census and survey data. Because identifiers in survey data may have smaller ranges, using the automatic ID formation may result in a different definition, e.g. SCCDDDVVHHHHH for census but SCCDDVHHHHH for survey. To avoid that from happening, an explicit formula may be specified as following:

STRATUM:1\COUNTY:2\DISTRICT:2\VILLAGE :2\HOUSEHOLD:4

The exact definition of the range of hierarchical ID is 'any integer number in the range of -9007199254740991 to 9007199254740992 or $-2^{53}+1$ to 2^{53} '. This limitation is due to the internal use of double precision variables as the carrier of hierarchical IDs. Please note that use of a decimal point in a hierarchical ID is not preferred because the operation of 'shift' works only on an integer. As for using the negative part of this spectrum, be advised that it is hard to read, inconvenient and thus not recommended.

Data array

A Data array is similar to 'table' in a relational database. It contains multiple columns, each one storing one variable. The header of a data array defines the attributes of each column. Variables defined in the header part of a data array could be vector, expression and/or alias. Vector corresponds to a sequence of data on the disk but expression and alias do not occupy any storage space. When expression or alias is evaluated, it is stored in a vector in the memory, not on the disk.

All external datasets have to be converted into a data array in PovMap2. Each PovMap contains a hierarchical ID, and each variable in a data array may be either continuous or categorical in type. The data array is sorted by its hierarchical ID at the time of conversion and cannot be altered afterwards.

Regardless of the difference in data structure, the concept of record or observation is still valid in a data array.

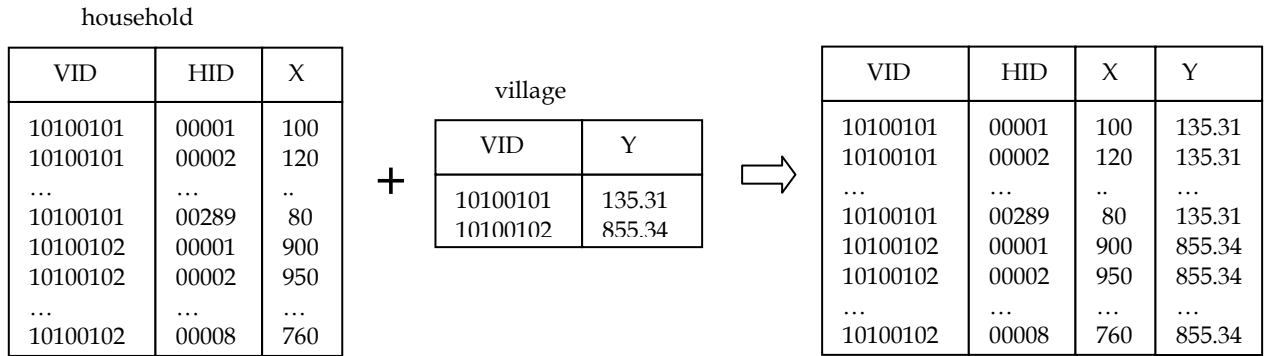
Relation without relational database

Even though the data engine of PovMap2 is not a relational database, it does perform the typical relational databases' function as long as the relation between the two data arrays is

DRAFT for comment; not for citation

defined with a common compounded ID. Imagine a dataset at the household level is linked with a dataset at the village level with village identifier VID. A SQL statement may look like:

```
Select household.vid, household.hid, household.x, village.y
from household join village where household.vid=village.vid
```



In PovMap2, this is done through the use of aggregation and distribution. Aggregation is a relation from multiple to one, and distribution is from one to multiple. When two data arrays are connected with a common key (in this case VID), both data arrays are surveyed to form a multiplication factor. A multiplication factor determines the number of cells to be aggregated or the number of cells to be repeated in distribution.

Users of SQL system should be familiar with the concept of 'left join', 'right join' or 'inner join'. If we consider household level data as in the 'left', then PovMap2 only provides left join since a record with missing household variables is useless.

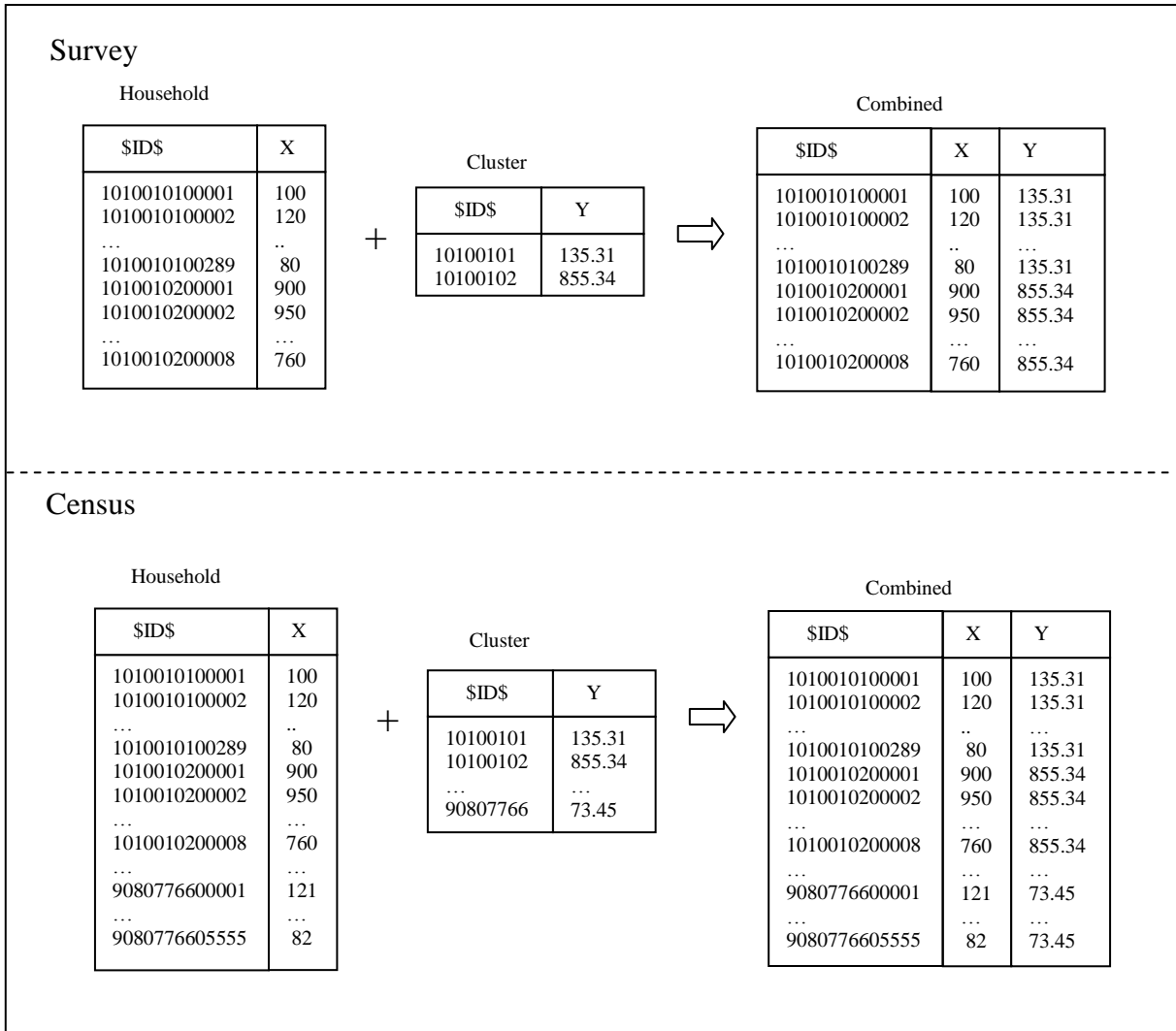
PovMap2 Project

A PovMap2 project always uses four data arrays. The reference to these data arrays and the linkage between them is stored in a special file with file extension PMP. Please do not alter the content of the PMP file.

The four data arrays are organized to store:

- Survey household level data --each record is for one household in survey.
- Survey cluster level data – each record is for one cluster in survey.
- Census household level data – each record is for one household in census.
- Census cluster level data – each record is for one cluster in census.

Combining the information on the section concerning hierarchical ID and the relation between data arrays, we can then construct a hypothetical example:



From this example, we can see a PovMap2 project file needs to store additional information on how the household and cluster files are linked. That is the *number of digits to shift* in survey and census.

In its simplistic form, a PovMap2 project need only have a survey household data array, census household data array, digits to shift for survey data to link to cluster level, and the digits to shift for census data. The cluster level data array will be constructed automatically. By default, the name of the cluster level file is a concatenation of the household file and “_cluster”.

Continue variable vs. Categorical variable

Variables in a PovMap2 data array have two different types – continuous variable and categorical variable. Continuous variables are those that take value from a continuous domain. Weight, height, income and spending are typical continuous variables. On the other hand, categorical variables take value from a specific set, typically a few integer values. *Gender* (1=male, 2=female) or *Education* (0=illiterate, 1=elementary, 2=middle school, 3=high school, 4=college, 5=post graduate) are typical categorical variables. In most cases, categorical variables take a handful integer values, but exceptions do exist. Commonly used Standard Industry Classification (SIC) code (<http://www.census.gov/epcd/www/sic.html>) is also a categorical variable but its value may range from two digits to four digits. There are some variables that can be treated as continuous as well as categorical depending on the interpretation. For example, ‘years of education’ can be treated in either way. Users of PovMap2 can identify the variable type to be explicitly continuous or categorical. PovMap2 also tries to guess the variable type during dataset import. Variables with limited integer values (limited in range of 0 to 15 by default) are consider to be a categorical while anything else is continuous. If SIC3 or SIC4 is used in the dataset, users should change the type to categorical manually.

Operator to Categorical variable

Categorical variables are also different from continuous variables in what kind of operation can be applied to them. Valid operations to continuous variables include +, -, *, / and functions like Log or Exp. Valid operations to categorical variables are limited. Addition, subtraction, or division of two categorical variables have no meaning (image what would be the meaning of *Gender/Education* ?). Invalid operation to categorical variable will cause an exception and the execution will not be conducted.

The valid operation of categorical variables include: a) interaction, b) comparison operation and c) arithmetic operation with a constant. Categorical variables that interact with a continuous variable will be explained separately. In following example, categorical variable *Gender* has value

- 1 Male
- 2 Female

categorical variable *HeadSector* is the two digits SIC code describing what industry the head of household is working, its values are:

- 7 Agriculture
- 10 Mining
- 15 Construction
- 20 Transportation/Public Utilities
- 40 Manufacturing
- 50 Wholesale Trade
- 52 Retail Trade
- 60 Finance, Insurance and Real Estate
- 70 Services
- 99 Unclassified

a) The interaction of two categorical variables is shown below. The internal value of interaction result is constructed with appropriate concatenation. Thus the order of interaction matters, *HeadSector*Gender* has different internal value than *Gender*HeadSector*.

<i>Gender</i>	<i>HeadSector</i>	Meaning	<i>Gender*HeadSector</i>	<i>eadSector*Gender</i>
1	15	A male working in construction	115	151
2	52	A female working in retail	252	522
1	10	a male in mining industry	110	101
1	15	a male working in construction	115	151
2	70	a female in service sector	270	702

b) The comparison expression for categorical variable is obvious: *HeadSector=10* will identify all heads of household working in the mining industry. *(HeadSector=70)&(Gender=2)* will identify all females who work in the service sector. It can also be expressed as *Gender*HeadSector=270* or *HeadSector*Gender=702*. Since **Recode** function is a compound comparison function, it can be applied to categorical variables but the user has to provide the label, i.e. the labels does not become involved in the computation.

c) There may be additional situations in using categorical variables, such as converting a 4 digit SIC code to two digits. It would be convenient if we could do *SIC2=int(SIC4/100)*. This brings up the third usage on categorical variables: arithmetic operation of categorical variables and a constant is allowed and the outcome is a numeric value. Users have to reset the type to categorical and provide a label to it manually.

Interact numeric value with categorical variable

In preparing data for regression, a special operation between a continuous and categorical variable is allowed. Because a categorical variable is equivalent to a group of dummy variables, the interaction of a continuous and a categorical variable is defined as the product of the continuous variable and set of dummies.

For example, *WorkingYears* is a continuous variable measuring the number of years worked. Putting it in an income model

$$y = a + b * WorkingYears + c * Male \quad (\text{Male is defined as Gender}=1)$$

This implies there is an increment of b dollar regardless of gender for each additional working year, and being male, there is always c dollar difference compare to the female regardless the experience. By introducing the interaction part $Male * WorkingYears$ and $Female * WorkingYears$, the regression become

$$y = a + d * Male * WorkingYears + e * Female * WorkingYears$$

This model allows for more flexibility. In *PovMap2*, the continuous variable x interacts with the categorical variable c and is implemented as n vectors: $x*(c=code1)$, $x*(c=code2)$, $x*(c=code3)$... $x*(c=coden)$ where n is the number of categories of c .

Labeling categorical variable

The label for categorical variable is crucial for understand its meaning. Users of *PovMap2* can specify labels in the provided text box in the following format:

1=Male

2=Female

The integer value on the left is the internal value of that categorical variable, and any text (including the space between two words not including the leading and padding space) on the right side of equal sign is the label. Wrapping label to next line is not allowed. When two already labeled categorical variables are interacted, the label will look like $Male * Construction$, or $Female * Service$.

Please note that labeling in *PovMap* is different than applying format to variables (as in SAS or Stata). Users may have to repeatedly define 1=Yes 2=No for each occurrence of a yes/no question. Use of clipboard will make it much easier.

Missing value

Missing value widely exists in statistical datasets. PovMap2 data arrays also allow for missing values. Arithmetic operations involving missing values always have a missing value. Similar to SAS, missing values are greater than any 'non-missing' value.

PART 3 OPERATING POVMAP V2.0

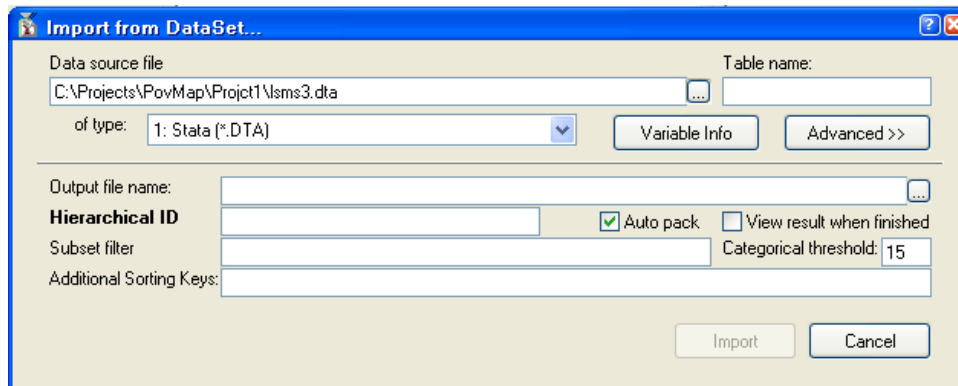
Import datasets

External datasets must be converted to PovMap2 data arrays before further use. The supported data format includes Stata, dBase, fixed column ASCII and tab (or comma) delimited formats, and, any datasets that can be read with an ODBC driver. To be able to read a non-ODBC dataset, PovMap 2 will use the file extension to identify the dataset type. Here are the file extensions associated with the supported non-ODBC dataset:

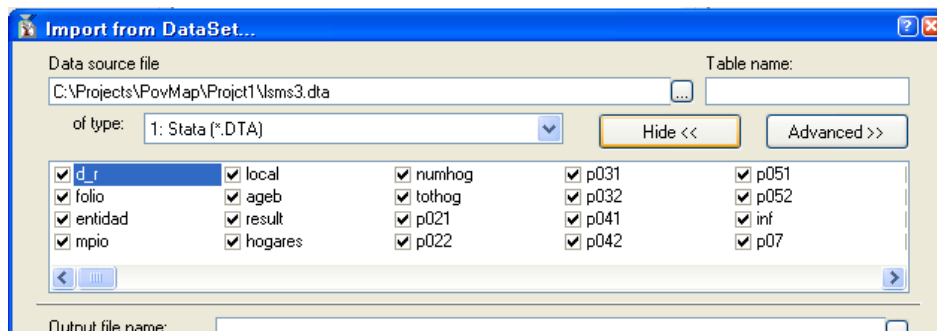
.dta	Stata dataset (version 2 to 8)
.dbf	dBase III, dBase IV, FoxPro dBase file
.csv	comma separated value ASCII file with field name in first row
.dat	fixed format ASCII file along with .dct file to describe the field attributes

Please note that you must determine the hierarchical ID before you start importing any dataset.

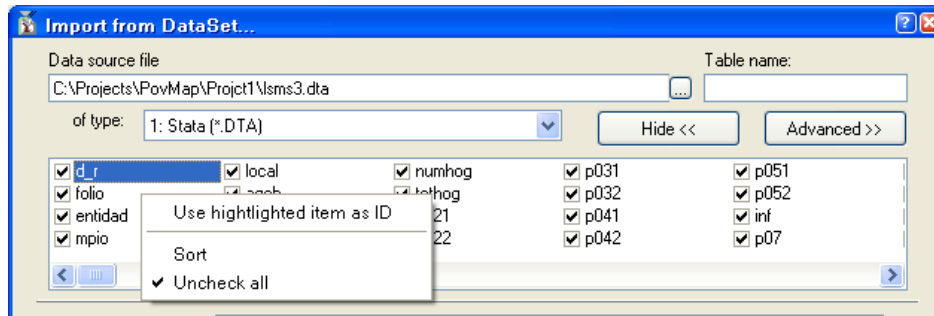
To start importing a dataset, click menu item Tools->Import..., you will see the following screen:



Click on the “Variable Info” button. Here you will see the names of all variables in the middle section. By default, all variables are selected, and the order of variables shown is the same order as in the dataset. Users can change the order of variable list by right-clicking within the empty space in the variable list. A pop up box will appear to let you sort by name or check/uncheck all.



Right-click to change order and select/unselect variables



The box in the lower-right corner labeled 'Categorical threshold' provides a 'cut-off' point for identifying categorical variables during importing. Any variable representing a whole number and being less than the threshold will automatically be marked as a categorical variable. This is designed to save the user from having to repeatedly set the variable type. Users can set the variable type to zero to disable the 'auto-categorizing' (assuming there is no negative integer). This setting can easily be changed in 'checker' screen.

- Notes on importing DTA, DBF files: DTA and DBF files contain header information. PovMap2 can read variable definitions from these headers and thus do not require additional auxiliary files.
- Notes on importing comma delimited files (i.e. CSV files): Variable names must be provided on the first line, separated by a comma. CSV files typically do not have a fixed length, thus each line is treated as one record. Data cells are delimited by a comma or tab.

```

nhid1,nhid2,nhid3,agric,lownfrm
10901512,1030111,5,1,0
10901512,1030211,8,1,0
10901512,1030311,,4,
```

Empty data cells will be read as missing. The number of data cells in each line should not exceed the number of variables defined on the first line. If the number of data cells on a line is less than the variables defined, the missing value will be assigned to all missing cells.

- In order to read a fixed format text file (i.e. TXT, DAT, or ASC), an auxiliary file should be provided to define all variables. This file should have the extension .DCT (dictionary file):

```
1      11 1 nhid1
13     11 1 nhid2
25      4 b nhid3
30      4 b agric
34      8 R earning
```

The format of the above file is a reduced form of a dictionary file used by a popular database conversion tool, DBMSCOPY. Users can utilize the DBMSCOPY dictionary file directly within PovMap2. The format should consist of each line beginning with a space. This will make it compatible with DBMSCopy. The four columns represent the starting column; width of a variable; type of variable; and variable name, respectively.

- For reading a SAS dataset, users must have the SAS system and SAS ODBC driver installed on their computers. For further details please refer to the following document: <http://support.sas.com/techsup/technote/ts626.html>

The SAS ODBC driver is very different from MS-Access driver. First it is necessary to identify the library location. Now assume you have data file

```
C:\Projects\PovMap\Census\CenData.sas7bdat
```

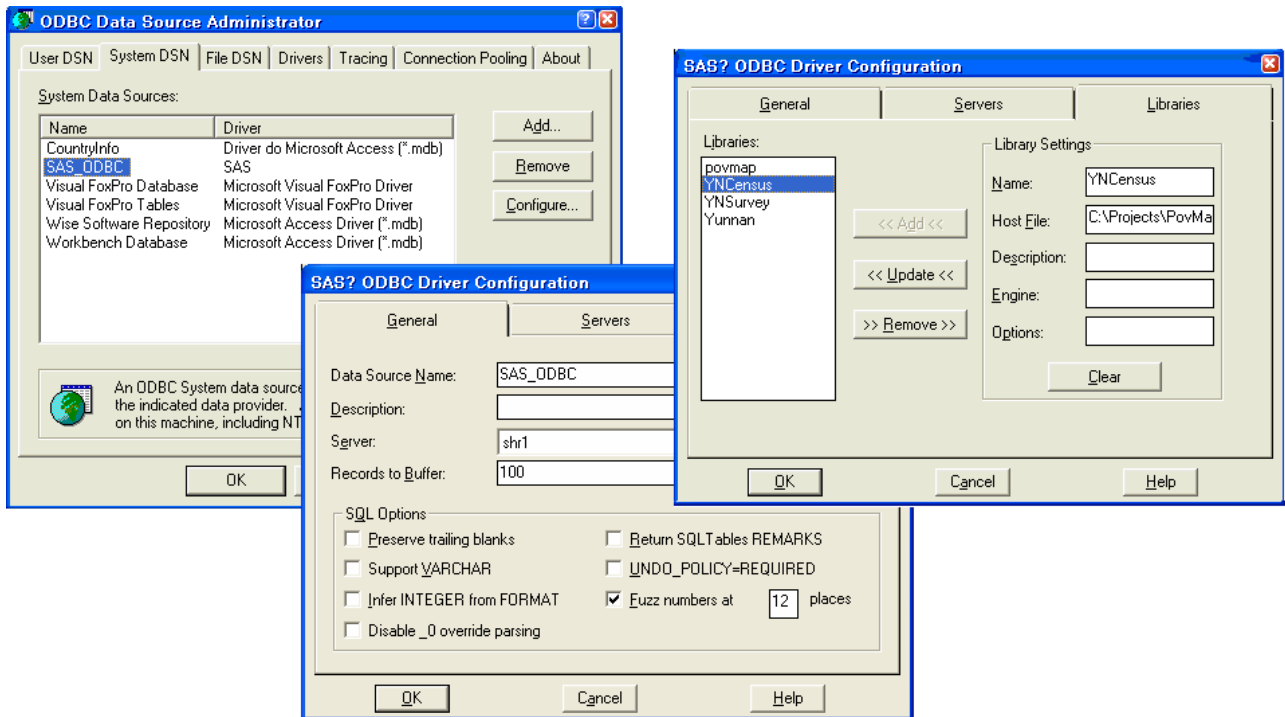
and

```
C:\Projects\PovMap\Survey\SurveyData.sas7bdat.
```

First thing to do is to create a system data source name (DSN) with Windows' ODBC definition utility. This DSN name will be used as library reference. A library reference in SAS is defined by a LibName statement which maps a directory to a library name. SAS users are accustomed to the following code; which is very similar to what we need to specify in PovMap2:

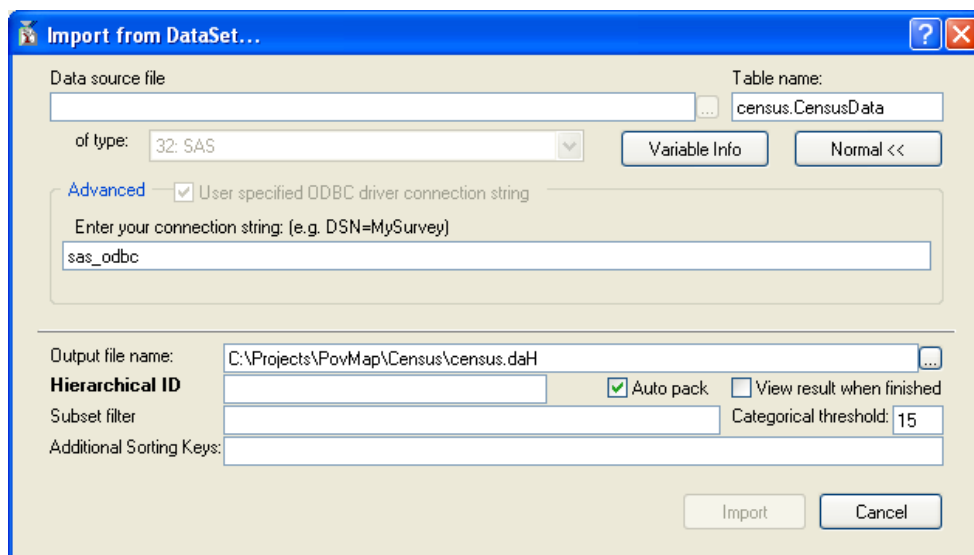
```
Libname myLib "Projects\PovMap\Census";
Data one;
  Set mylib. CenData;
  . . . .
```

To define DSN, open the Windows ODBC driver set up utility. The following illustration consists of three screen shots of the SAS ODBC driver configuration.



The name in the libraries box is a library name that refers to a location (directory) as in the 'Host File' box.

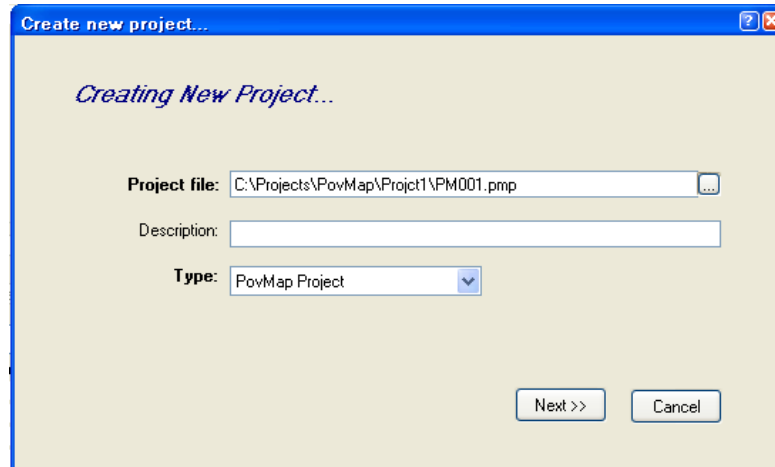
To use SAS datasets in PovMap, click Tools->Import->Advanced, then insert the dataset reference. The following screen shot resembles a Libname statement.



- For all other datasets, PovMap2 will be able to read them as long as they have a corresponding ODBC driver.

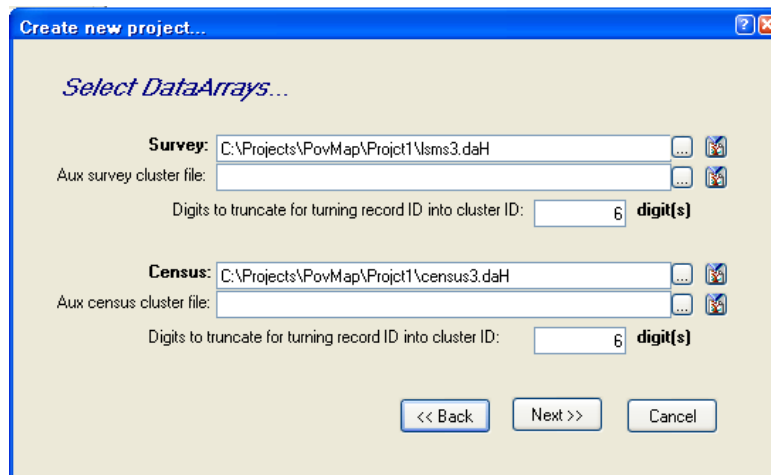
Creating PovMap Project

To create a poverty mapping project, click File->New Project and you will be prompted to fill in the *Creating New Project* form:



Where project file (type PMP) will store all information about this project.

Click *Next* to proceed where you will then need to specify how data arrays are linked inside this project. Please refer to section “PovMap2 Project” to gain a better understanding of the data linkage. Besides the survey and census household file names, you must also provide a number to shift the ID of the household file to match the ID of the cluster file. This needs to be done for both survey and census respectively.



With the provided information, PovMap2 will examine, for survey and census separately, the ‘mergeability’ of data arrays. The summary of ID matching is shown next screen-shot. We can read from there the ID at household level in survey ranges from 22561021 to 488327719, each observation in survey household file has its own ID (as in the min=1 and mix=1 on ‘cluster size’). After shifting household ID 6 digits to the right, the ID will group to size of 10 and range from 22 to 488. Similar reading is for census data.

If you find the summary ID matching is not what you expected, go back to review the information in the previous screen.

It should be noted that the ID in household level file need not to be household ID, one can use hierarchical ID at cluster or village instead. The only setback of doing so is that when user view the data array, they can only tell 'this is the third record in that cluster' but they can't tell which household that is. In this case, the number of shift to form cluster ID can simply be 0.

	Range of ID		Cluster size	
	min	max	min	max
Record ID in survey:	22561021	488327719	1	1
Survey cluster ID from truncation:	22	488	10	10
Cluster ID in aux survey cluster file:				
Record ID in census:	1497822	496634168	1	1
Census cluster ID from truncation:	1	496	10	213
Cluster ID in aux census cluster file:				

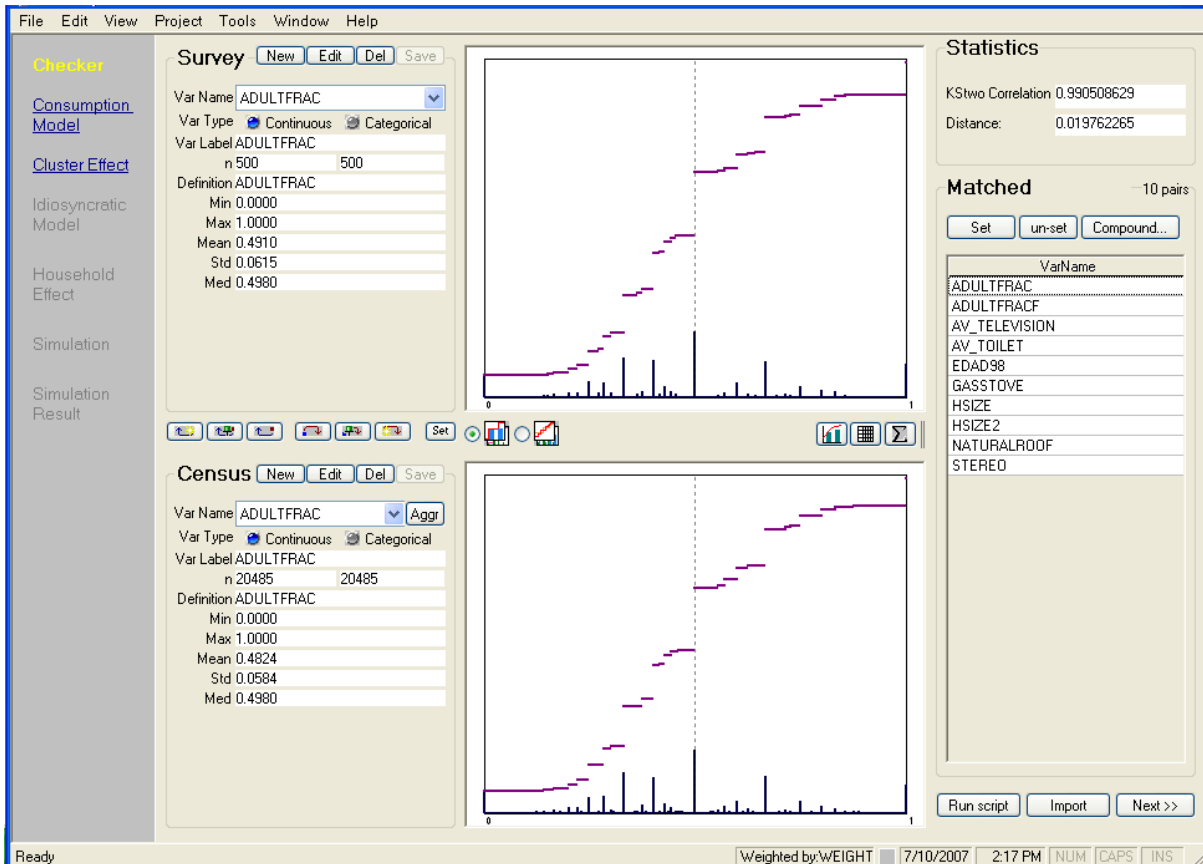
Next screen to emerge is to assign the weight. Survey data are typically weighted but census data are typically not weighted.

	NumObs deduction
Survey: WEIGHT	0
Census: \$DEFAULTWEIGHT\$	0

By now all the conditions for setting up a project are ready. The information can't be altered (except the weighting). If the mistake were made such as the shift of ID, user has to start over.

Screen Checker -- Finding Candidate Variable Pairs

Paring up survey variable and census variables is done by 'Checker' screen. Following screen shut shows the components of checker.

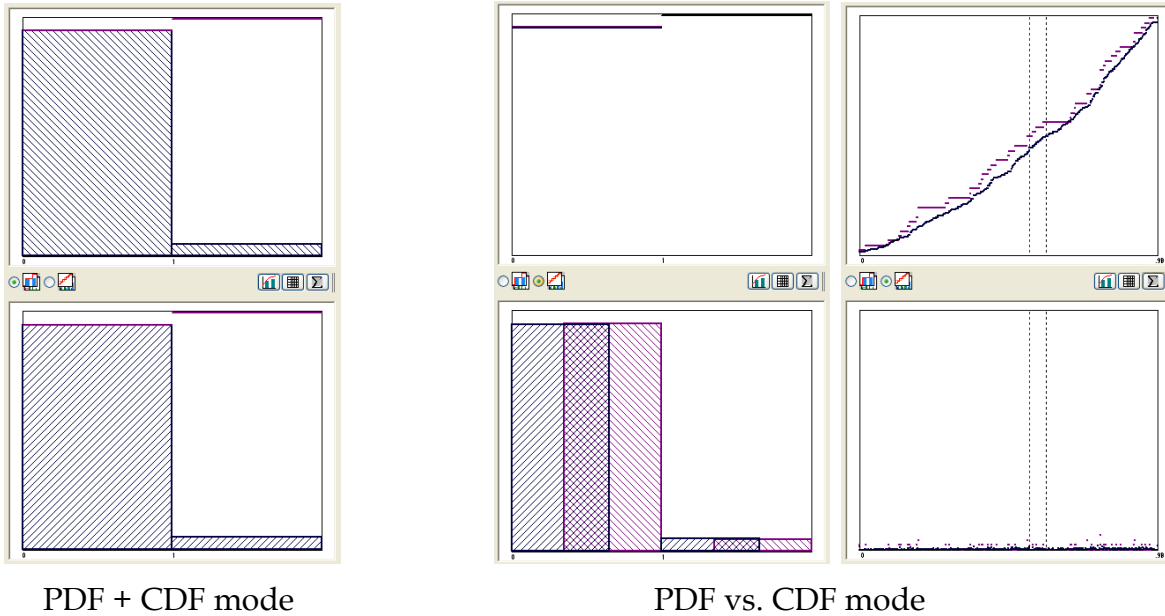


Checker allows users to recode/create variables, compare descriptive statistics of the survey and census data, edit variable properties, and set the variables that will be used in the regression. Matching independent variables in the survey and census can be done manually using , or by inserting the independent variables into the window (under certain condition). In order to be set (included) into the model, variables in the survey and the census must have the same type. If the variable in survey and census has different name, PovMap will prompt for the common name. Once set into the model, the variables are referred to as "matched". The matched variables to be included in the model are shown on the right side of the check screen.

Identifying a variable's type (continuous or categorical) is a unique feature of PovMap, a variable's type can be changed using the window when necessary. Using the "draw" feature, a visual comparison of survey and census variables can be conducted, displaying frequencies and cumulative distributions respectively for categorical and continuous variables. Multiple visual representations enable users to examine potential "matched" variables, button is for 'unlocking' the matching.

The graphic button on the center of screen provide different charting combinations, Two radio buttons let you set the charting mode to PDF+CDF mode or to CDF vs. PDF

mode. In the following charts, the far-left chart has survey's CDF and PDF plotted on the upper part and CDF and PDF of census plotted on the lower part.



In the contrast, the charts on the right superimpose CDF of survey and census on the upper part and PDF of survey and census on the lower chart.

Detail Explanation of Checker Screen

1. Comparing Variables in Survey and in Census

For helping diagnostic the likelihood of selected variable pair, PovMap2 provides a statistics on the upper right corner of the screen.

When comparing two categorical variables in the survey and census, PovMap displays the chi-square statistic, which compares the survey frequencies to census frequencies. The significance of the chi-square statistic indicates whether the survey and census have similar frequency distributions, and is one method for determining if the variable should be included in the analysis.


When two continuous variables is compared, PovMap calculates the Kolmogorov-Smirnov (KStwo) statistic, which is a measure of the correlation between the cumulative probability distribution functions in the survey and census. The distance measure provided with the KStwo is the maximum distance between the survey and census distribution; small values suggest that the survey variable is representative of census variable. The KStwo value represents the significance of the distance measure, which when significant, theoretically suggests that the survey is not representative of the population for the chosen variable.¹


¹ For a detailed discussion of how to interpret the chi-square and KStwo statistics, see chapter 14.3 in the *Numeric Recipes in C* website: <http://www.nrbook.com/a/bookcpdf.php>

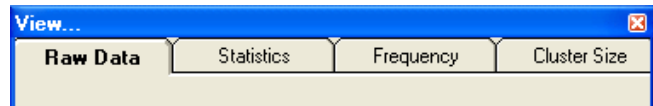
Important reminder on the comparing categorical variable with chi-square test: the values of two categorical variables may not have the same meaning even through they take same value, e.g. value '1' of variable *GENDER* may mean 'Male' while value '1' in *OWNTV* may mean 'owning a TV'. Comparing of any two variables should not be done blindly.

2. Summary Statistics of Selected Variable

For any selected pair of variable, the buttons on the center of the screen can be used to check the value, frequency table (for categorical variable), univariate table for

continues variable. Button  and


 both open a same popup screen with four tabs, the difference between them is the default tab.

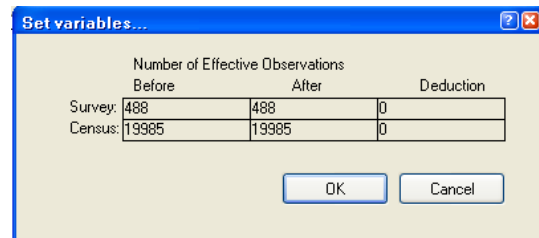


In order to accelerate the computation of distributional analysis for continuous variables, a 'bin count' method is used. It is accurate enough for diagnostic purpose. This method uses 500 bins to cover whole range of the continuous variable with equal interval. The interval is optimally rounded to a proper value to avoid value with long decimal (e.g. 1.24976531832).

3. Set and un-set a pair

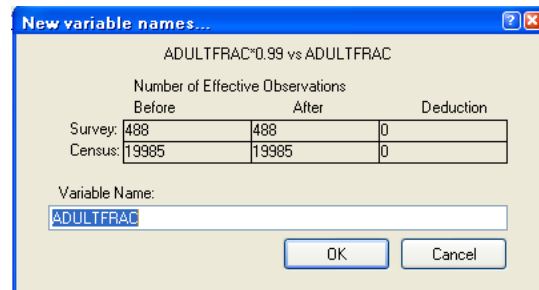
When the variable pair being compared is satisfactory, you can use button

 to add them into the matched variable list. This will popup a dialog to show you the possible drop of effective records in following regression analysis due to the missing value in the variable you are adding.




box

If the survey variable and census variable have different name, the dialog box will have an additional text box about the common name you are going to use. In case, a new variable will be created and original variable will not be changed (original variable is read only. Always!).



this the

Un-set (un-pair, un-match) a variable is straight forward, click button  to select what variable pair to be removed from the paired list. This action did not delete the variable from the data array.

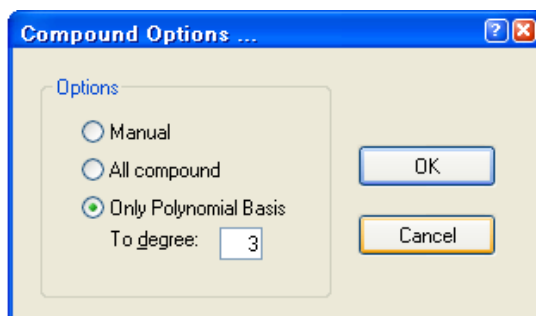
4. Rules of Interlocking

In order to avoid conflict, complex rules are implemented. You may see some buttons or text boxes in the checker screen locked, grayed-out or blocked, that is because the internal locking rules. The rules may be too complicate to list them all but following principal can help you understand them:

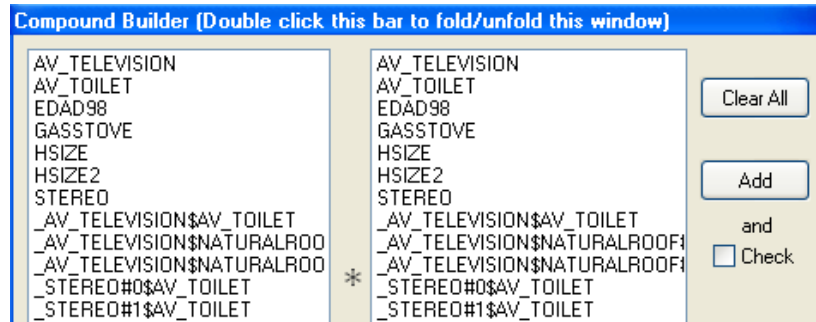
- Expression does not take disk space but variable always occupy disk space.
- When an expression is saved with button **Save**, they will be evaluated and a variable will be create to hold all the value. In the 'Definition' box, the formula of this expression will be kept and further editing is disabled.
- Anything in the matched list is variable (occupy disk space).
- Original variable should not be altered (change, delete) but non-original variable can be deleted.
- Categorical variable has to be integer type.
- Variable or expression can not be paired if they are different type.
- Each variable can be paired only once. If A is paired with B, then B (or A) can't be paired to C.
- Variable with single value can't be used for charting
- The property of a paired variable (type, value...) can't be changed, un-set them before making changes.
- Variable in the cluster section will be evaluated in the household level on demand.

5. Generating Compounded Variable

The paired variable can be used to generate higher order compound mechanically. Three compound methods are provided: *Manual*, *All compound* and *Only polynomial basis*.



Denote the paired variables as $\{x_i\}$ where $i=1,2,\dots,m$. selection of *All compound* will produce $\{X_i * X_j\}$ where $i,j=1,2,\dots,m$ and $j \geq i$. Selection of *Only polynomial basis* will produce $\{X_i^2, X_i^3, \dots X_i^k\}$, where k is the highest order. Selection of *Manual* will show an additional screen to let you select a subset of $\{X_i\}$



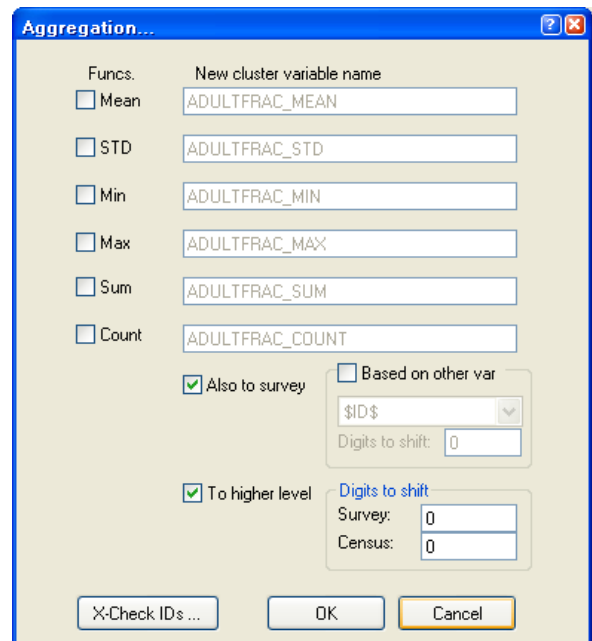
Since the compound is only applied to the paired variable, the definition of each pair should be already carefully confirmed, thus, the compound dialog has an button, click it to let PovMap2 select the variable for you when variable pair in the compound list satisfy a pre-specified level.

6. Aggregate census data and make it available in survey

It is often needed to make aggregation of census variable such as average ownership of TV or percentage of people with higher education at cluster (or county, district even province/state) level. Typical procedure of making such operation in common statistical package like SAS or Stata involves making aggregation into another dataset and then merge it back. In PovMap2, thanks to the required sorting order, this can be done instantly with button .

The dialog screen will let you specify whether the aggregation should be distribute to survey data arrays, user can also change the level of aggregation higher. The later is very useful to produce the province/state average.

If your aggregation would be based on a order incompatible to the hierarchical ID such as urban/rural or the variable of terrain type, the items inside the 'Based on other variable' can be activated and filled, whatever variable selected can be used (directly or with shift) to determine the grouping of this aggregation.

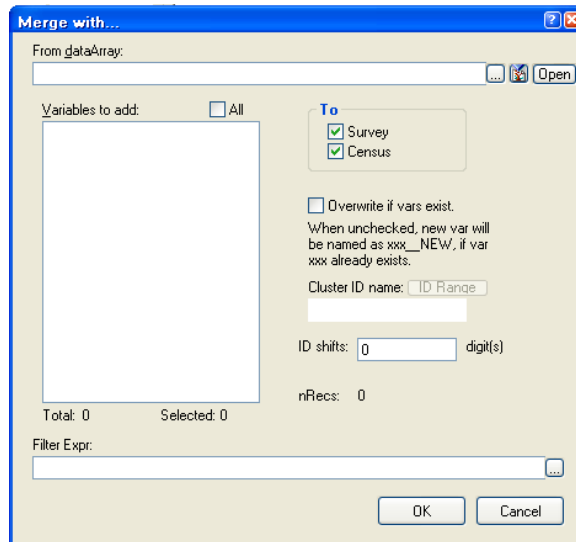


Obvious, aggregating census variable into survey's cluster level requires the hierarchical ID in census and survey to be compatible. Otherwise, the survey data array will receive all missings.

7. Merge in external dataset

To merge in an external dataset, user should convert the dataset into PovMap2's data array and then click menu item *Project->Merge Cluster Vars*. Merge in arrays to household level is not allowed because it is unlikely to happen.

The operation of this dialog screen is similar to the *aggregation* function. only different in the data flow is instead of aggregating household to cluster level the cluster level data read from external data array.

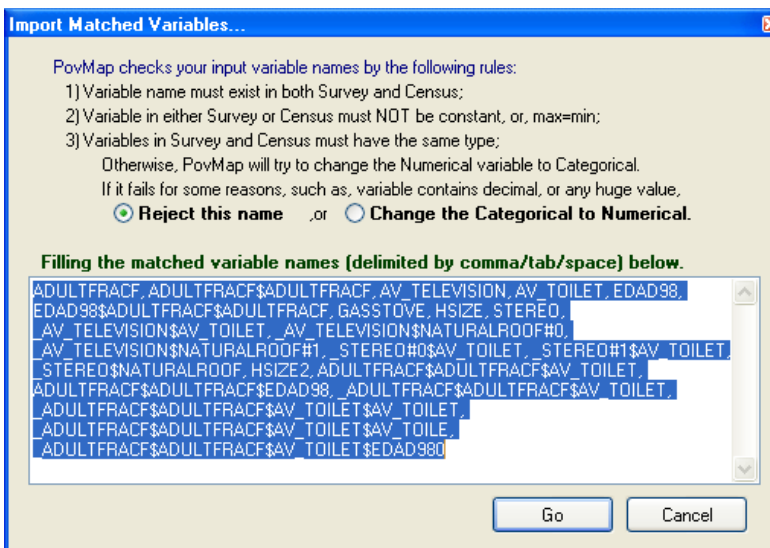


data
very
The
data
are

8. Set multiple pairs by import

In response to user's request about setting a group of variables in pair, PovMap2 can accept a list of variable names in text mode then parse it to individual name and set it in pair. Click button and a dialog box as in the next picture. The conditions for using this function are

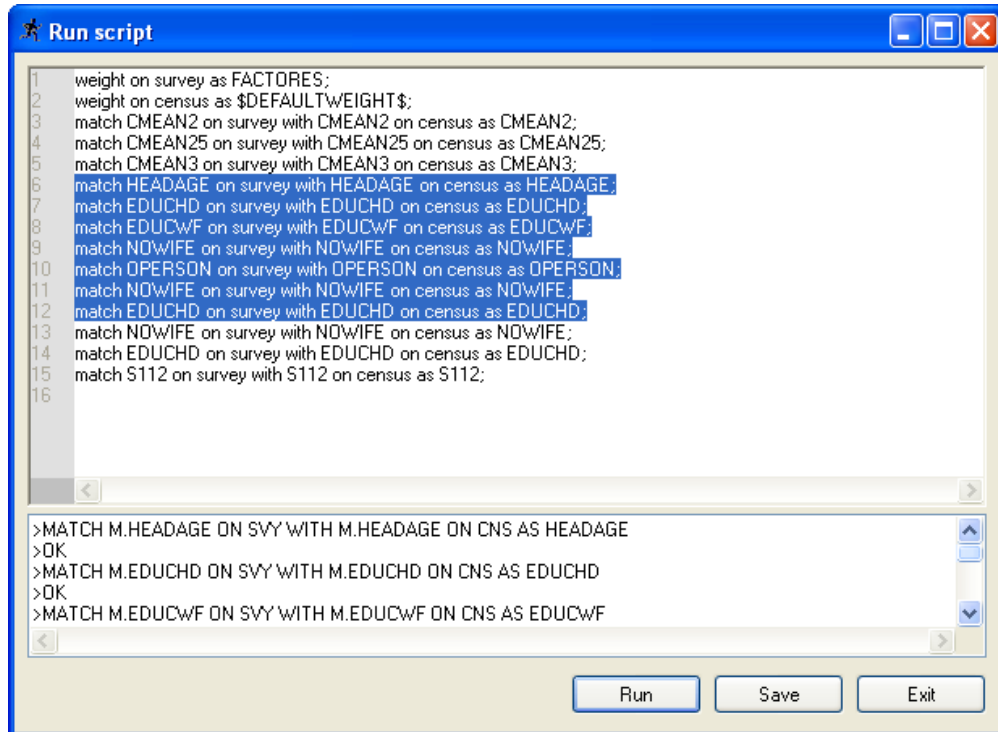
- 1) Variable name must exist in both Survey and Census;
- 2) Variable in either Survey or Census must NOT be constant, or, max=min;
- 3) Variables in Survey and Census must have the same type.



Otherwise, PovMap2 will try to change the numerical variable to categorical. If it fails for some reasons, such as, variable contains decimal, or any huge value, user can choose to either reject that name of change to continue variable.

9. Batch job with script

All the tasks of creating new expression, saving as variable and matching them up can also be done in batch mode. By default, your Poverty Mapping project will open a log file. All the actions that change the data array will be recorded as script. The script can be reused.



```
1 weight on survey as FACTORES;  
2 weight on census as $DEFAULTWEIGHT$;  
3 match CMEAN2 on survey with CMEAN2 on census as CMEAN2;  
4 match CMEAN25 on survey with CMEAN25 on census as CMEAN25;  
5 match CMEAN3 on survey with CMEAN3 on census as CMEAN3;  
6 match HEADAGE on survey with HEADAGE on census as HEADAGE;  
7 match EDUCHD on survey with EDUCHD on census as EDUCHD;  
8 match EDUCWF on survey with EDUCWF on census as EDUCWF;  
9 match NOWIFE on survey with NOWIFE on census as NOWIFE;  
10 match OPERSON on survey with OPERSON on census as OPERSON;  
11 match NOWIFE on survey with NOWIFE on census as NOWIFE;  
12 match EDUCHD on survey with EDUCHD on census as EDUCHD;  
13 match NOWIFE on survey with NOWIFE on census as NOWIFE;  
14 match EDUCHD on survey with EDUCHD on census as EDUCHD;  
15 match S112 on survey with S112 on census as S112;  
16
```

```
>MATCH M.HEADAGE ON SVY WITH M.HEADAGE ON CNS AS HEADAGE  
>OK  
>MATCH M.EDUCHD ON SVY WITH M.EDUCHD ON CNS AS EDUCHD  
>OK  
>MATCH M.EDUCWF ON SVY WITH M.EDUCWF ON CNS AS EDUCWF
```

Run Save Exit

Screen 2 -- Consumption Model

Overview

PovMap2's second page, consumption model, is also referred above as the "beta" model. Functions built into this screen are for finding a best consumption prediction model with survey data. The basic elements handled here are regressors. Regressor could be a continue variable, or a group of dummies generated from a categorical variable. User can view the regressor, make mean table and correlation table or running regression analysis. Multiple model selection methods are available including OLS, forward selection, backward selection and setepwise selection. An experimental 'overfitting' diagnostic method is also available. The result of latest regression can be stored into a handy 'Model Pad' for further comparison or for model testing.

Dependent Variable: LNCONPC

Statistical procedure: OLS

Model selection significant level: Entry: 0.2 Stay: 0.15

Regressors:

Estimate linear model with OLS. At least one regressor should be selected.

Number of Observations used in the Model=488 Number of Records in the dataset=500
 Number of Regressor=26 Number of Model=7 LHS variable=LNCONPC
 total Weight=488.0000 Num of Cluster=50

SST=269.3436 SSR=119.8507 MSE=0.3108 RMSE=0.5575
 F=64.2708 R2=0.4450 adjR2=0.4380

Result: Drop Outliers

	Coefficient	Std. Err.	t	Prob >t	Label
intercept	6.6250	0.1066	62.1710	0.0000	Intercept
GASSTOVE_1	0.3234	0.0578	5.5956	0.0000	Dummy for (GASSTOVE)=1
HSIZE	-0.3395	0.0359	-9.4560	0.0000	HSIZE
HSIZE2	0.0149	0.0029	5.1093	0.0000	HSIZE2
_AV_TELEVISION\$NATURALROOF#1	0.4250	0.1870	2.2726	0.0235	AV_TELEVISION * (NATURALROOF=1)
_STEREO#0\$AV_TOILET	-0.4819	0.0832	-5.7936	0.0000	(STEREO=0) * AV_T
_STEREO\$NATURALROOF_01	-0.3278	0.0915	-3.5817	0.0004	Dummy for _STEREO\$NATURALROOF=1

Ready Weighted by: WEIGHT 7/31/2007 4:46 PM NUM CAPS INS

Detail

1. LHS, RHS and Regressor

User must specify the LHS variable, or dependent variable, as well as the RHS regressors.

Regressor could be a continuous variable, or a group of dummies generated from a categorical variable. The name of dummy regressor is a concatenation of the name of categorical variable, an underscore sign and the value. In principal, a categorical variable with k different values will spend-off k dummy regressors (i.e. no omitted dummy). However, categorical variable with value 0 and 1 will be treated slightly

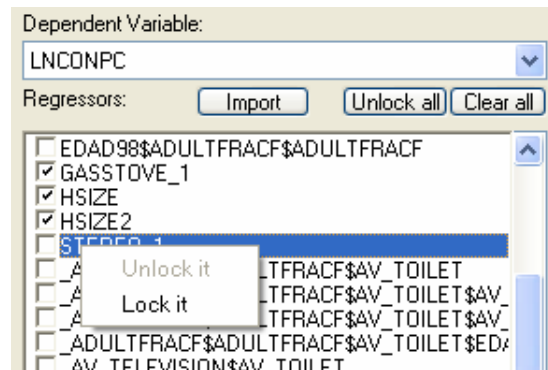
different – only dummy for value 1 will be used as a regressor. This approach makes the regressor list much more intuitive and readable when the data array has a lot of dummy variables.

2. Four states check box

The check box in the regressor list is a special type, it has four status: unchecked (white background), unchecked and locked (gray background), checked (checked with white background) and checked and locked (checked with gray background). Basically, it is combination of *checked/unchecked* and *locked/unlocked*. The regressor in a *locked* mode will remain its status during model selection, while *unlocked* regressor may be added to or removed from the model.



To lock a regressor, right click over the regressor and then select lock or unlock. Button **Unlock all** will release all locks in the regressor list. Button **Clear all** will remove all checked regressor from the model if they are not locked.



2. Analyzing mean table and correlation table

Selecting *Mean of Selected* from the task dropdown and click **Run** button, a mean table will be shown as follows, which include all selected regressors and the

	Count	Mean	Std.Err.	Min	Median	Max	WeightSum	0.0500	0.100
LNCONPC	488	5.1912	0.5531	3.2746	5.1754	7.6284	488	4.0218	4.282
AV_TELEVISION	488	0.4310	0.0841	0.0000	0.5200	0.8837	488	0.0000	0.000
AV_TOILET	488	0.5667	0.0824	0.0000	0.5890	0.9861	488	0.1000	0.142
GASSTOVE_1	488	0.2869	0.2050	0.0000	0.0000	1.0000	488	0.0000	0.000
HSIZE	488	5.0246	6.4429	1.0000	5.0000	14.0000	488	1.0000	2.000

LHS variable. The columns with heading 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95 are the percentage deciles. This table is a 'sortable' table, user can click on the column header to sort the result.

Similarly, user can make correlation table. It is also a 'sortable' table. It is useful to checking the correlation among regressors.

	AV_TELEVISION	AV_TOILET	GASSTOVE_1	HSIZE	LNCONPC
AV_TELEVISION	1.0000	0.1852	0.5182	0.0259	0.1292
AV_TOILET	0.1852	1.0000	0.0736	0.0040	-0.1818
GASSTOVE_1	0.5182	0.0736	1.0000	0.0457	0.1975
HSIZE	0.0259	0.0040	0.0457	1.0000	-0.5513
LNCONPC	0.1292	-0.1818	0.1975	-0.5513	1.0000

3. Regression analysis and model selection

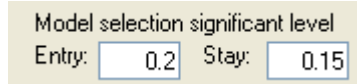
To build a good consumption prediction, user can choose from OLS regression, Forward selection, backward selection or stepwise selection.

OLS

This method is the default and provides no model selection capability. All the regressors selected in RHS list will be used for estimation. It is possible to see an “Matrix is un-inversable” error message when the regressors selected is co-related.

Forward Selection

The forward-selection technique begins with regressors already selected. For each of the regressors, this method calculates F statistics that reflect the regressor ‘s contribution to the model if it is included. The p-values for these F statistics are compared to the value in *Entry* box. If no F statistic has a significance level greater than the *Entry* value, the forward selection stops. Otherwise, the forward method adds the regressor that has the largest F statistic to the model. The forward method then calculates F statistics again for the regressors still remaining outside the model, and the evaluation process is repeated. Thus, regressors are added one by one to the model until no remaining regressor produces a significant F statistic. Once a regressor is in the model, it stays.



Model selection significant level	
Entry:	<input type="text" value="0.2"/>
Stay:	<input type="text" value="0.15"/>

Backward elimination

The backward elimination technique begins by calculating F statistics for a model, including all of the regressors. Then the regressors are deleted from the model one by one until all the regressors remaining in the model produce F statistics significant at the value specified in the *Stay* box. At each step, the regressor showing the smallest contribution to the model is deleted.

Stepwise selection

The stepwise method is a modification of the forward-selection technique and differs in that regressors already in the model do not necessarily stay there. The stepwise method looks at all the regressors already included in the model and deletes any regressor that does not produce an F statistic significant at the *Stay* box. Only after this check is made and the necessary deletions accomplished can another regressor be added to the model. The stepwise process ends when none of the regressors outside the model has an F statistic significant at the value in *Entry* box and every regressor in the model is significant at the *Stay* level, or when the regressor to be added to the model is the one just deleted from it.

Single step model selection

All the model selection methods described above have correspondent *Single Step* variations. This, associated with the regressor locking function, provide most flexibility in model building.

4. Testing for over-fitting problem

This function is an experimental component. It is designed to determine if the explanatory power of the beta model is dependent on the idiosyncrasies of the particular sample. The test generates a set of sub-samples and runs regressions analysis of same model on each of them, the sub-sample is formed by excluding observations related to certain categorical variable. The statistical summary of the regressions are collected in one table. When over-fitting is not present, regression statistics and parameters should be roughly the same. Large differences between models, however, suggest that the model is over-fit to a specific characteristic of the whole sample. For example, if the cluster ID is the examined categorical variable, large differences in R^2 's and parameter coefficients suggest that the model fit of the entire sample is dependent on idiosyncrasies within a specific cluster or clusters.

5. Viewing data and Excluding outliers

User can use *view data* to browse through the survey data array. All the effective observations are shown in the data grid. If some of the rows need to be excluded, user can right-click on the row header and select *Set outlier* to mark it as an outlier, a blue dot will show up on the right of scroll bar and the outlier row will also be mark with same color. This mark is useful to re-allocating the outlier record: all you need is to drag the slider on the scrollbar to near the dot and the outlier record will show up on the data grid.

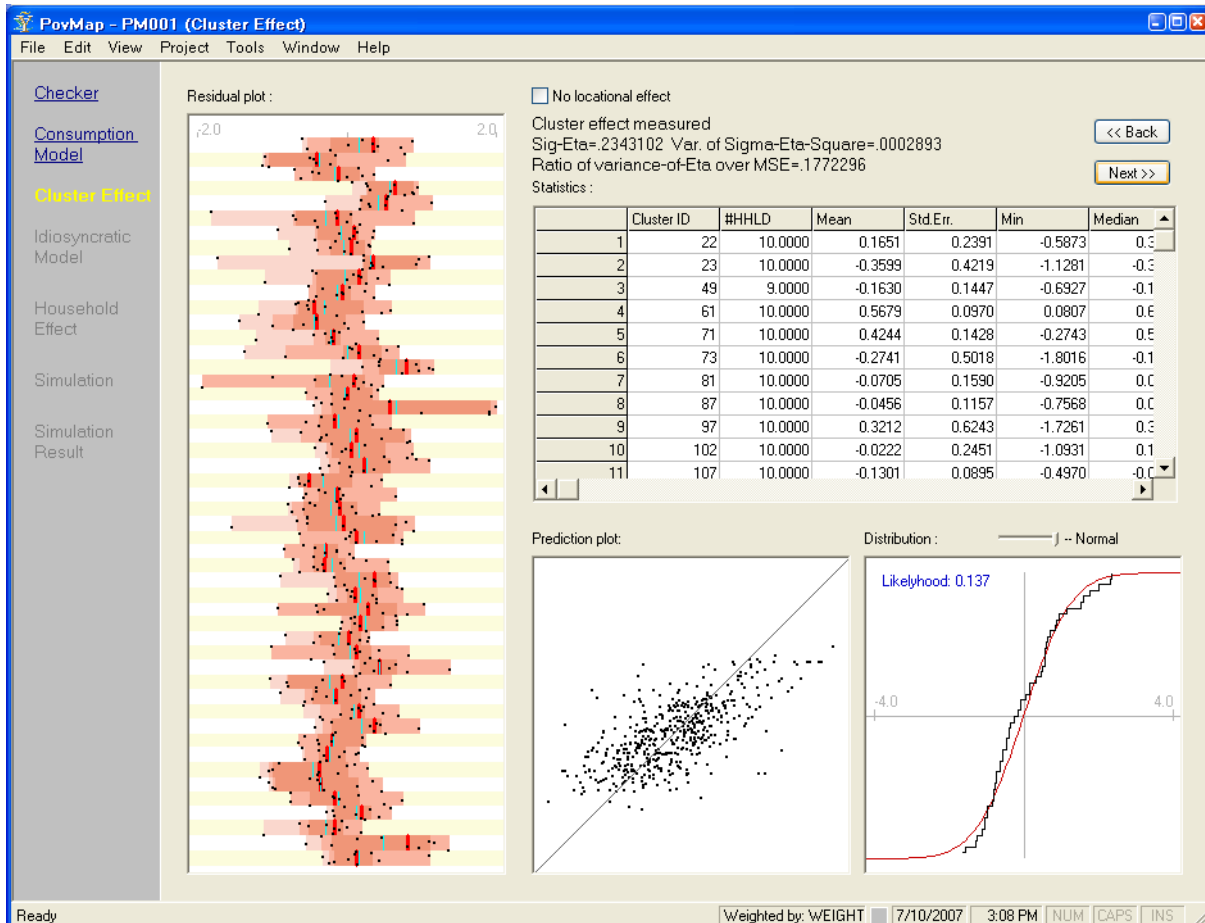
Result: Drop Outliers

	LNCONPC	EDAD98	HSIZE
1	5.2791	32	4
2	5.5130	60	5
3	5.2374	46	11
4	5.4433	36	6
5	4.8006	45	9
6	6.1213	64	3
7	6.1980	62	4
8	5.1172	48	5
9	5.2331	67	6
10	4.9869	50	7
11	6.0171	28	5
12	5.5447	39	3
13	4.5898	55	3
14	4.7524	62	5

Screen 3 -- Cluster Effect


Overview

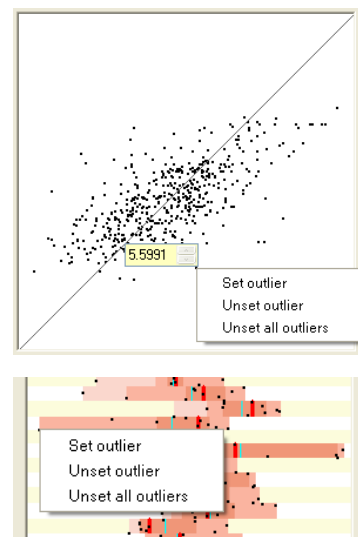
This page displays components of the cluster-level effect and enables users to disable the “locational effect” and determine the distribution of locational component, which will be applied to the census data during the simulation. Specifically, this screen displays 1) the residuals plotted of all household color coded by cluster, 2) a detail of locational effect per cluster, 3) a scatter plot of actual (Y) and estimated (\hat{Y}) values for each household, and 4) a and cumulative distribution of the cluster effect.



Detail Explanation of Cluster Effect Screen

- 1) The Residual Plot organizes the model residuals by cluster. The mean and median residual for each cluster are displayed by green and red vertical lines, respectively. The distribution of the residuals is displayed on a three-tone salmon-colored gradient by percentiles. From lightest to darkest, the distributions shown represent the 0-100th, 10th – 90th, and 25th – 75th percentiles.

- 2) The Prediction Plot is a scatter plot of the actual (Y) and estimated (\hat{Y}) values for each household, where the y-axis represents the actual and the x-axis represents the estimated dependent variable.
- 3) The illustration of the cumulative distribution of the model's predicted values (\hat{Y}) enables users to identify and visually choose a normal or t distribution using  scroll bar. Users should choose the distribution that most closely matches the shape of the cumulative distribution of their model. Representing the sum of the difference between the predicted and pre-set (normal or t) cumulative distributions, the likelihood statistic is a tool for choosing the most appropriate distribution. Generally speaking, the smaller the likelihood statistic, the more similar the distributions. The selected distribution will be stored by PovMap and used in the simulation process. As anthropometric indicators for a population are typically normally distributed,
- 4) When location effect is not desired, a check box on the upper-center can be used to ask PovMap2 to turn off the locational effect. Intuitively, if the residual plot is homogeneous across all clusters, they may not be much cluster effect to model. This can also be seen from the ratio of variance of eta to the mean squared error (MSE). The ratio tells how much of total variation (measured by MSE) can be interpreted by the cluster effect. If this ratio become negative (which is mathematically impossible but computational feasible due to the accumulation of computing error), user **must** disable the locational effect in order to continue to next screen.
- 5) On the X-Y plot, user can right click over a point then select from three choices *Set Outlier*, *Unset outlier* and *Unset all outliers* to declare the select data point to be outlier (see detail on consumption model screen). The outliers marked this way have the same behave as the outliers marked in the data grid of consumption model screen, they will be used on next regression if the *Drop outliers* box is checked. User can also find a text box with light yellow



round showing up to the upper-left of the data point, this is the Y values in the neighborhood of selected point. Similarly, in the residual plot, user can also declare outliers.

However, dropping cases should not be done by visual examination only, this is not a statistically defensible procedure. Outliers should be removed from the data, if at all, based on robustness of the regression.

b
a
c
k
g

Screen 4 -- Idiosyncratic Model

Overview

Also referred to as the *alpha* model, the idiosyncratic model estimates household effect. As the household variance is *not* constant, and is allowed to vary with some explanatory variables, it is considered a model of heteroskedasticity. The dependent variable, annotated as `_ALPHALHS_` (i.e. alpha-left-hand-side variable), is affected only by variables whose value affects the variance of the error term, and we have no basis for deciding *a priori* which variables will have variances that vary systematically with the value of the variable. Thus it is logical to estimate the parameters using stepwise regression, in contrast to the beta model. Following models of heteroskedasticity, potential regressors are generated from matched variables and additional interaction terms involving \hat{Y} and \hat{Y}^2 (\hat{Y} is the predicted consumption level from *beta* model).

Dependent Variable: `_ALPHALHS_`

Statistical procedure : `OLS`

Model selection significant level: Entry: `0.2` Stay: `0.15`

Estimate linear model with OLS. At least one regressor should be selected.

Number of Observations used in the Model=485 Number of Records in the dataset=485
 Number of Regressor=141 Number of Model=20 LHS variable=_ALPHALHS_
 total Weight=377098.0000 Num of Cluster=39

SST=2673.9929 SSR=711.9553 MSE=4.2194 RMSE=2.0541
 F=8.8806 R2=0.2663 adjR2=0.2363

Result:

	Coefficient	Std. Err.	t	Prob >t	Label
<code>_intercept_</code>	-4.1308	0.3136	-13.1737	0.0000	Intercept
<code>S1019</code>	0.0000	0.0000	6.1370	0.0000	S1019
<code>S1121</code>	-0.0001	0.0000	-3.9045	0.0001	S1121
<code>S12</code>	-0.0027	0.0008	-3.3990	0.0007	S12
<code>S1219</code>	0.0000	0.0000	-5.9199	0.0000	S1219
<code>S1221</code>	0.0000	0.0000	3.9734	0.0001	S1221
<code>S23</code>	0.0184	0.0073	2.5223	0.0120	S23
<code>S314</code>	0.0392	0.0135	2.8963	0.0040	S314
<code>S417</code>	0.0582	0.0176	3.3006	0.0010	S417
<code>S420</code>	0.1002	0.0295	3.3945	0.0007	S420

Details

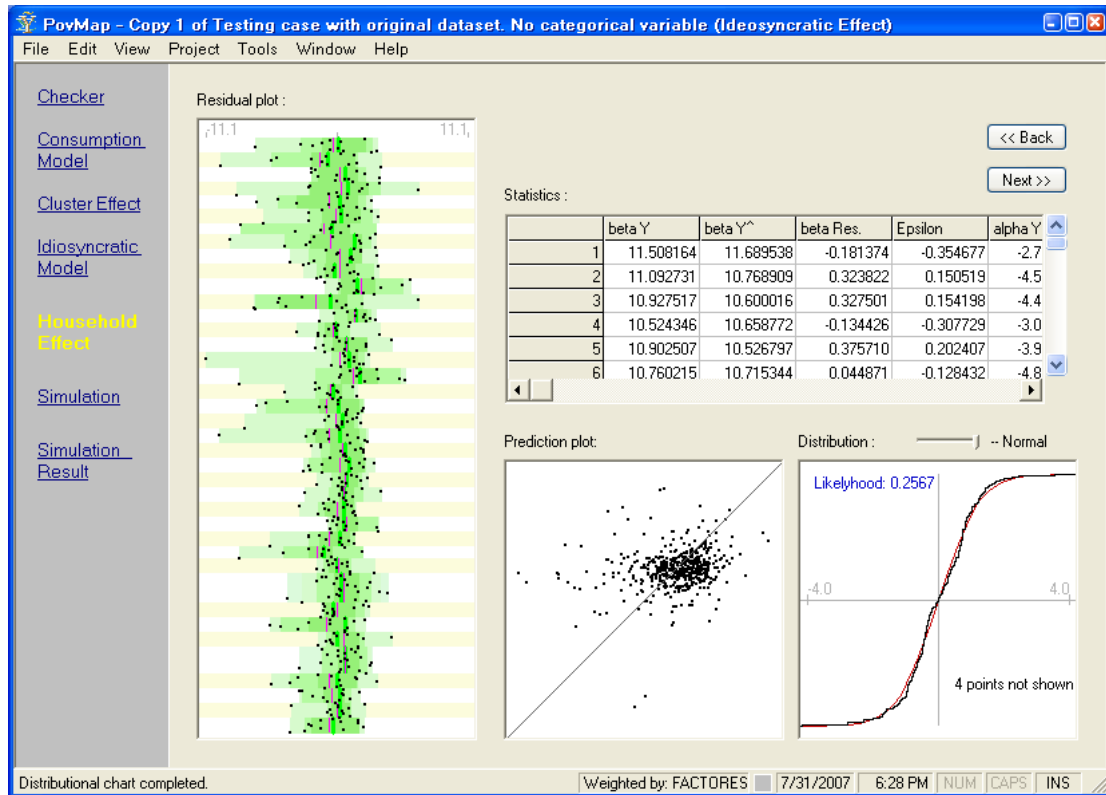
1. Unlike the *Beta* model screen, the button *Add to Model Pad* does not exist. That button is designed for making statistical inference on whether to partition the sample, it is not the task to be done in this screen.
2. For not modeling the idiosyncratic effect, use *Clear All* to empty the model.

Screen 5 -- Household Effect

Overview

The household effect screen examines how much heteroskedastic variation can be explained by idiosyncratic model. It includes 1) the alpha residuals plot, 2) a table with *Beta* and *Alpha* estimates by household, 3) a scatter plot of the actual (αY) and estimated ($\alpha \hat{Y}$) for each household, and 4) cumulative distribution of the estimated error term.

Despite all the information provided, only minor adjustments can be made to the model in the household effects screen. Specifically, only the distribution of the error term can be determined using the slider bar.



Details

1. As in the cluster effect, noting the likelihood statistic can help determine which distribution is appropriate. The likelihood statistic has the same meaning as in the cluster effect, showing sum of the difference between the predicted and pre-set (normal or t) cumulative distributions.
2. There is no outlier setting function in this screen.

Screen 6 -- Simulation

Overview

The simulation process in PovMap refers to the point in poverty mapping when the parameter and error estimates from the survey are applied to the census data. The simulation screen allows users to modify simulation settings by 1) changing the distribution of cluster effect and idiosyncratic effect, 2) setting trimming parameters for simulation, 3) specify the simulation parameters, 4) specifying additional information for output, and 5) identifying the type and level of simulation. In addition, all the settings could be reformatted into a text mode configuration file similar to PCF file in PovMap version 1.

The screenshot shows the PovMap Simulator Configuration window. The window title is "PovMap - Copy 1 of Testing case with original dataset. No categorical variable (Simulator Configuration)". The interface is divided into several sections:


- Distribution:** Cluster effect: Normal, Household effect: Normal.
- Trimming:** min Y imputed: AUTO (with a note "****(9.36)"), max Y imputed: 12, Beta: 0.99, Alpha: NONE, Eta: NONE, Epsilon: NONE, Estimated Y: NONE.
- Simulation:** Number of replications: 10, Initial random seed: last, Additional shift for cluster effect: 0, Aggregation levels: 0 3, Simulation method: Classical (partial) Draw, Output style: Total Only.
- Indices:** FGT: FGT0, FGT2 (checked), Poverty line: 45476, General Entropy: GE0 (checked), Atkinson: ATK1, ATK2 (unchecked).
- Household Size:** OPERSON (selected), Gini, 90% to 10% ratio, 95% to 5% ratio (unchecked).
- Misc.:** Y in logarithmic form (checked), Saving all poverty/inequality indices, Saving all estimated Y (unchecked).

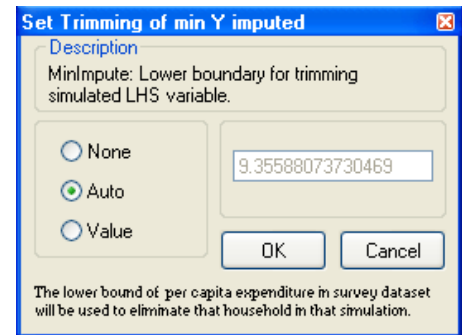
The "Output file" is C:\Projects\PovMap\w2\TestData\small485\Test485.pou. The status bar shows "Ready", "Weighted by: FACTORES", "8/1/2007", "11:46 AM", and keyboard shortcuts NUM, CAPS, INS.

Details


1. Household size must be identified before simulation started. Household size acted as a weight to the household data in census, this ensures the simulation result to representing the total population.
2. The distributions of the cluster and household effect are loaded from earlier screens (slider control in the cluster effect screen and house effect screen). Also available on the dropdown list is the *Semi parametric* distribution, it draws from the pool of residuals in survey. The household effect has one more type of random component

-*hierarchical semi parametric*. It must be coupled with the use of *Semi Parametric* random component in the cluster effect. In this mode, when cluster effect picked up the *i*th cluster, the household effect will be drawn from all household residuals in cluster *i*.

3. Number of replication -- Number of simulation. A required field. Preset to 100 times.
4. Poverty line. A required field. Its value could be a number or a variable name which stores a 'flexible' poverty line, i.e. a poverty line may vary over region, cluster or even household.
5. Min Y imputed – The lower boundary for trimming simulated LHS variable. Preset to *Auto* (*n.nnn*) where *n.nnn* is lower bound of Y value in survey. Click button  will bring up a dialog box like the one on the right. The choice *None* will remove any limitation on the minimum value of imputed Y. When *Auto* is used, the lower bound of Y value in survey dataset will be used. This value is shown on the right but no editing is allowed). If *Value* is selected, then the value box will be fully editable and user can specify a desired value in the box.
6. Maximum Y imputed – similar to the minimum Y imputed, but the maximum value.
7. Beta – the acceptant probability of *Beta* vector. Default to *None* or 1. If we denote this number as p_0 , and solve for x_0 such that $\chi^2(x_0, df) = p_0$, then the redrawing of Beta vector occur when $|beta| > x_0$. This is because *Beta* vector is drawn from normal distribution such that its mean equals to the estimated *Beta* and its covariance matrix equals to the covariance matrix of *Beta*, there is small chance, during the drawing, the drawn *Beta* vector become too strange and cause the imputed Y completely out of range. Since the mold of *Beta* is a χ^2 distributed random variable, redrawn when $|beta| > x_0$ could eliminate extreme value.
8. Alpha – the acceptant probability of *Alpha* vector. Treated in the way similar to *Beta* vector.
9. Eta – trimming for cluster effect. Default to *None*. Specifies the range for cluster effect (location effect) trimming. Similar to item 6 (Min Y imputed). The selection *Auto* corresponds to the range $(-v, v)$ where v is the largest absolute value of cluster effect in survey.
10. Epsilon – trimming of household effect. Default to *None*. Similar to item 10 (Eta).
11. Estimated Y – trimming for simulated y including all random components. Default to *None*. Its value determines a range (y_{min}, y_{max}) . Any simulated y will be excluded (set to missing) outside of this range. Please note that this number must be in the real term.



12. Indices – Indices of poverty and inequality measurements. At least one indices show be selected. For General Entropy measurements, user specific value is provided. The value 1.5 will interpreted to GE1.5. The Alkinson measurements have similar arrangement. On the *Distribution* box, user can select from (10 20 30 40 50 60 70 80 90), (20 40 60 80 100), (1 5 10 25 50 75 90 95 99), :10 or :20. Notation ‘:10’ means the distribution will be shown in 10 equal interval groups. User can also type in different percentile values in similar fashion.
13. Number of replication – Default to 100.
14. Initial random seed -- This determines what is the first random number used in the bootstrap. Preset to 1234567. All random components will be affected. When omitted or set to 0, internally produced fully random number will be used. This seed is derived by the system clock in 1/1000 second resolution. Thus no two simulations will be equal if seed is set to 0. A small button nearby could be used to retrieve the random see used in the last simulation, which could be very useful in determine a proper trimming.
15. Aggregation Level – Specify at what level the simulations is run and aggregated. Grouping is form when the shifted hierarchical ID changes. When n=0, the hierarchical ID will be used and aggregation is for each districted ID value. When n>0, the ID in census dataset will be shifted n digits to the right to produce a shorter ID that represents an aggregation on higher level, new aggregation will be outputted when this value changes. For example, if ID is the form SCCDDD (multiple household share same district ID. Cluster is at district level), then SIMULATION=3 will produce a estimates at the county level (SCC). Multi-level simulation can be requested by values like 0 3 5, which estimates at district level (SCCDDD), county level (SCC) and stratum level (S).
16. Additional shift for cluster effect – for simulating cluster effect on higher level than cluster. Default to 0. It is often needed to simulate cluster effect on ‘above cluster’ level. This option let user specify where the cluster effect should be drawn. Using the hierarchical ID SCCDDD as an example, if county is a more appropriate clustering level, use 3 in this box.
17. Simulation method – specifying the method. Currently have *Simultaneous drawing* and *Classical (Partial) Drawing*. Other two options is still under development. See Peter Lanjouw’s notes on this topic on Special topic section.
18. Output file name – the name of output file from simulation, This name will also affect other auxiliary files (see below).
19. Y in Logarithmic form – indicating the LHS variable is in log form thus, the real term should be computed with exponential transformation. Default to *Yes*.
20. Saving all poverty/inequality indices – for requesting all poverty/inequality measurements to be saved in a file. The output file is in the same directory and has the same file name as the output file except the file extension is ‘pdump’.

21. Saving all estimated Y – for requesting all estimated y to be saved in a file. The output is a PovMap2 data array in the same directory as the output file and its file name is made of output file name and ‘_ydump’. This data array can be easily converted to Stata or other file format. User could also adding other variables into the *ydump* file by selecting from the variable list in ‘*Along with the following variables*’. The selected variable will be listed in the text box but editing in the box is prohibited.
22. Using Script tab. Script tab is design to provide user with a script based simulation configuration similar to the PCF file in the version 1 of PovMap. A button  can be used to convert all interactive specifications in Config tab as a text based configuration file. User can modify the configuration to achieve additional functions, mainly in submitting multiple simulation runs without human intervention. When user want to run simulation 10 times (say 10 simulations each consists of 100 replications), they can repeat the Simulation= clause 10 times as show in next box:

```
Outputfile=C:\Projects\PovMap\Projct1\PM001x.pou
nSim=100
CDist=T(11) *** distribution of location
(clustering) effect ***
HDist=N *** distribution of household effect ***
PovLine=45676
Indices=FGT0
seed=1234567
IsLog=YES
ydump=YES

Simulation=6
Simulation=6
Simulation=6
Simulation=6
Simulation=6
Simulation=6
Simulation=6
Simulation=6
Simulation=6

End
```

Screen 7: Result from Simulation

Overview

The simulation result screen has maximum four tabs, 1) summary, 2) results, 3) yDump output and 4) pDump output.

1. The summary page provides a overview on the model, simulation setting and random number generation log. It is intended to cover all aspects not included in the result tab.
2. The results page is a big spreadsheet. Each row summaries the simulation result of an aggregation group which is determined by *Aggregation Levels*. The columns include the number of household or individual in each, the min/ max imputed Y's, and the average and standard error of the estimated LHS variable, each requested indices takes two column: one for the average and the other for the standard error (across all simulations).

	Unit	nHHLDs	nDroppedHHI	nIndividuals	nSim	Min_Y	Max_Y	Mean	StdErr	avg_FGTO	se_FGTO
1	2090150	154962	201	154761	100	-8.4472	27.4011	10.0308	0.1065	0.0751	0.0138
2	2090152	569	4	565	100	-9.8005	30.3382	9.2050	2.3656	0.3526	0.2919
3	2090153	382	0	382	100	-8.4111	27.2669	8.8234	2.2781	0.4040	0.3020
4	2090156	1710	0	1710	100	-8.0656	30.1166	9.2919	2.5948	0.4137	0.2313
5	2090157	621	18	603	100	-2.1284	31.1294	9.6468	1.1832	0.3605	0.1420
6	2090158	860	119	741	100	-8.9957	29.8239	8.4314	2.4771	0.5044	0.2219
7	20901	159104	342	158762	100	-9.8005	31.1294	10.0081	0.1078	0.0836	0.0143
8	209	159104	342	158762	100	-9.8005	31.1294	10.0081	0.1078	0.0836	0.0143
9	2	159104	342	158762	100	-9.8005	31.1294	10.0081	0.1078	0.0836	0.0143

Details

1. Unit – the ID of one aggregation group. Representing all households that have the same 'shifted' hierarchical ID.
2. nHHLDs – number of households in that aggregation group.
3. nDroppedHH – number of household excluded from the simulation. It is determined once each run before all simulations to eliminate household with 'bad' predictor. (i.e. a out of bound $X*\beta$ where 'out-of-bound' is related to the trimming of imputed y)
4. nIndividuals – total number of individual in that aggregation group.
5. nSim – the number of simulation
6. Min_Y – the minimum of estimated y of all replications in that aggregation group.
7. Max_Y -- the maximum of estimated y of all replications in that aggregation group
8. Mean – average of mean value of estimated y of all replications
9. StdErr – average of standard error of estimated y

2.8 Tools and facilities

2.8.1 Viewing data array

2.8.2 Exporting data array

2.8.3 General provision of grid operation

2.8.4 Concatenation of data arrays

2.8.5 Additional sorting order

2.8.6 Project property

2.8.7 Save to another project t

2.8.8 Compact and repair data array

PART 4 SPECIAL TOPICS

Guideline for Data Preparation

This note is provided by Peter Lanjouw as a basic outline of the steps that are involved in implementing the poverty mapping methodology. We try to give an intuitive flavor of what is involved in each of the steps that need to be carried out, rather than being perfectly precise. The note should be read in parallel with the more rigorous methodological discussion provided in Elbers, Lanjouw and Lanjouw (2002) 'Micro Level Estimation of Welfare' Policy Research Working Paper 2911, the World Bank. We assume familiarity with the broad outlines of the poverty mapping methodology.

The text below illustrates a number of points with reference to certain country examples (usually Morocco). The specific country context invariably influences how the poverty map methodology is implemented. The aim of the examples is to provide some feel of the kind of in-country background against which various decisions and assumptions have to be made.

The data preparation stage is concerned with identifying the common variables that exist between the household survey and the population census. These variables will form the "bridge" that allow us to predict consumption levels into the population census. For the poverty mapping exercise to be valid, it is crucial that these linking variables are identically defined in the two datasets. This cannot be simply assessed by looking at the respective questionnaires.

Steps to be followed in preparing the data:

1. Comparing census and survey variables
 - a) Calculate means of "candidate" variables in the survey
 - "Candidate" variables are those for which both the survey and census instruments ask identical, or very similar, questions on. These variables are "candidates" for inclusion in the prediction models estimated in the next stage, subject to them being truly the same in the two questionnaires. To establish whether the questions are indeed soliciting information on the same point, it is important to closely scrutinize the wording of questions in the questionnaires. But this is not enough; it is also important to calculate some basis summary statistics on these variables in both data sources, to check whether they are indeed trying to get at the same thing.
 - In most settings there are generally four or five classes of variables from which the set of "candidates" are constructed. First, both survey and census questionnaires generally include questions on the demographic characteristics of households - size of family, age of members, gender, relationship to head, and so on. Second, household survey and population census questionnaires generally ask about the education levels of all family members. Such information can be very useful for the poverty mapping project in that these provide a window on the human capital of

the household – an important correlate with economic welfare. Third, survey and census questionnaires typically ask about the occupational status of family members, sometimes even soliciting detailed information about specific sectors of employment and precise activity of each working-age family member. Fourth, it is common to find cases where the census and survey questionnaires provide detailed information on a variety of housing characteristics, ranging from materials with which the house has been built, to access to a variety of utilities. Finally, it is not unheard of to find cases where census and survey questionnaires provide some details on the household ownership of some consumer durables. It is important, where possible, to produce a set of candidate variables which draws from all five classes of variables. Experience shows that this greatly improves prospects for obtaining models with high explanatory power.

- Construction of candidate variables should attempt to capture as well as possible the economic welfare of households. This implies that one needs to look well beyond variables defined at the level of the household head. For example, it is possible that the household head is a pensioner but that he or she lives together with one or more grown-up family members who have their own specific occupations and who contribute to household income (and thus consumption). To capture this feature of the household, it may be necessary to construct candidate variables such as dummy variables on occupations of other family members, with an eye towards capturing in particular those cases where there are other family members with well paying jobs. This would help to distinguish such households from those where the household head is a pensioner, and no other family members contribute to household income either. The latter type of household is clearly very likely to be poorer than the former.
- “Candidate” variables should be defined at the level of possible responses to specific questions. For example, in the case of a question on the type of water supply used by a household, one candidate variable might be defined as: “drinking water supply=private well”; another might be defined as “drinking water=cistern truck”; and so on.
- Where candidate variables are not categorical (e.g. household size, dependency ratios, etc.) also construct percentile distributions. These percentile distributions permit close comparison between the census and surveys of the tails of the respective distributions.
- Candidate variables may be constructed by combining information from various variables. For instance, household welfare and (delayed) school enrollment are likely to be correlated. Delayed enrollment can be captured by calculating an ‘education deficit’ as age-6-number of years in school.
- Use sampling weights in these calculations. When comparing against population statistics from the census, it is important that the survey means be calculated so as to reflect population, not sample, moments.

b) Calculate means at the level of “domains”, not only the national level.

- Means and percentile distributions should be calculated at the level of geographical disaggregation at which the regression models in stage 2 will be estimated. This level of geographical disaggregation can be designated a “domain”. How these domains are defined depends on the sampling design of the household survey. A domain should generally represent a stratum (or an aggregation of strata) in the household survey and should be large enough to include a sufficient number of households for the next stage regression analysis (preferably 300-600 households).

In some settings, such as Morocco, the household survey sample has been stratified down to the regional level, and distinguishes as well between urban and rural areas within each region. There are about 16 regions in Morocco, and given the limited sample size of the household survey (about 5000 households) this implies that there are some regions in which only few households were sampled (less than 100). One proposal might be to combine the 16 regions in Morocco into 6-8 “domains” which are built on the basis of 2-3 geographically contiguous regions. The issue to consider when contemplating this option concerns what is being assumed when contiguous regions are combined into a single domain. When a single model is being estimated for the domain, it will essentially be assumed that parameter estimates on regressors in the regions that make up the domain are the same across the regions. Whether this is reasonable or not can be tested explicitly on the basis of Chow tests of structural differences across sub-samples. Note, a degree of flexibility can also be maintained by including regional dummies in the model estimated for the domain, and by interacting with these dummies with at least some of the household characteristics.

Note, it is important that separate rural and urban domains should be defined.

c) Means and percentile distributions to be calculated with the census data for each domain.

- Census data to be divided into the same domains as defined above.
- For continuous variables like household size, calculating percentile distributions – even when the survey and census means are comparable, remains important. In Uganda for instance, mean household size was identical between the survey and census. Yet the survey distribution had much thinner tails than that for the census. In one stratum for instance, the fraction of one person households was 18.4% according to the census and 16.3% according to the survey. Further investigation pointed towards a problem with the replacement of non-responding households. As non-responders are more likely to be small, these households are under-represented in the survey, unless the replacement scheme takes household size into account. As this was not the case, it was decided to adjust the survey weights. The reweighing procedure followed is known as poststratification adjustment. It ensures that the weighed relative frequency distribution among mutually exclusive and exhaustive categories in the survey corresponds precisely to the relative

distribution among those same categories in the census. A danger of reweighing along one dimension –household size in this case, is that other survey variables that are representative using the ‘old’ weights become unrepresentative once the weights have been adjusted. In the Ugandan case however, reweighing increased the number of variables that passed the census-survey comparison test.

d) Census and survey means to be scrutinized carefully for comparability.

- Where the survey and census year do not correspond to exactly the same period, it is not reasonable to expect the two means to coincide exactly. This implies that statistical tests of equality of the two means may not be that meaningful – even where equality is rejected there may still be a case for using the variable in question simply because during the intervening time period there has been some change in values of the indicator in question. For the purpose of the poverty mapping methodology the fact that there has been some change in the value of the indicator over time (levels of education have improved somewhat, or access to public utilities has gradually expanded, etc.) does not necessarily invalidate it from use in the procedure. The key assumption that does then need to be imposed, however, is that the basic conditional correlation between welfare and this indicator remains unchanged over the time period. This assumption cannot generally be directly tested. Whether it is a reasonable one to make depends on knowledge one has of the underlying processes which have taken place over time (relative price shifts, etc.)
- Rather than comparing census and survey means for strict equality the key issue here is to get a sense of whether the variables are capturing the same thing. Does the census apply a different definition of what constitutes household membership? Do the census and survey employ the same definition of household head? Are occupation codes the same? Sometimes the way questions are posed results in different types of responses. For example, suppose that the census asks about source of potable water and allows for only three possible responses: private well; cistern truck; other. The household survey may ask the same question but allow for a much larger variety of possible responses. This could result in the situation that even when one looks at the “private well” response, the two data sources suggest that different percentages of the population have this as main source of drinking water. Simply because more options were available in the survey than were in the census, responses to even supposedly comparable options are no longer comparable. It is important to check for this possibility by careful scrutiny of all possible responses. Only those that really seem to be capturing the same basic features of the data can be designated as “candidate” variables for the next stage regression models.
- In Morocco, the population census data has not been entered for the entire population (even though each household in the population was covered). Rather, for reasons of cost and time, information from a sample of the entire population census questionnaires was entered and analyzed. In total, data for about 1,000,000

households were entered, representing around 20% of the population. The sampling structure of the census sub sample is not a simple random sample. Rather, a scheme was applied whereby for communes with less than 500 households all data were entered; for communes comprising 500-1200 households 50% of the household questionnaires were entered; for communes of 1200-3000 households 25% of household questionnaires were entered; and for communes larger than 3000 households 10% of questionnaires were entered. The computerized census data file includes expansion factors with each household that reflect this sampling structure. When calculating census means and percentiles, it is necessary to use these weights so as to produce meaningful summary statistics at the level of the domains described above.

2. Constructing commune (or cluster) -level variables from the census and ancillary sources

- Alongside the household level variables, the regression models in the second stage will also include some variables that are not at the household-level, but rather at the level of the cluster (or primary sampling unit) that underpins the household survey. Most household surveys are based on a complex sample design that involves both stratification and clustering. For example, prior to drawing the sample of households the country is first divided into a number of mutually exclusive strata (rural and urban areas, regions, etc.). Then within each stratum, a series of clusters (groupings of households) are drawn randomly. Finally, within each drawn cluster, a sample of households is drawn.
- One of the important concerns in the second stage regression modeling exercise is the question of whether the econometric model is able to capture intra-cluster correlation across households in welfare. It is possible, for example, that within a specific cluster, households are all typically less well off, or better off, than similar looking households in other clusters. This could be due to cluster-level factors such as whether or not land is irrigated in that cluster, whether household in a particular cluster have access to certain public goods and infrastructure, and so on. While many of the cluster-level characteristics of interest may not be readily observed in the census and survey data that are available for analysis, it may be possible to proxy these factors by including a number of such cluster-level variables in the regression model. The way that this is approached in this methodology, is to use a two pronged strategy:
 - I. Means and proportions are constructed in the population census at the level of the cluster that underpins the household survey. In many countries the cluster in the survey is equivalent to the census tract in the population census and so these means and proportions are constructed at the census tract level. Once a census-tract database of means and proportions has been constructed in the census, the means for the relevant clusters can be merged with the household survey and these census means can thereby be added to the list of “candidate variables” for the second stage analysis. Parameter estimates obtained from

the second stage analysis can then be applied to all of the cluster means in the census.

- II. Ancillary datasets may be available that provide summary statistics at the local level for the country as a whole. The BADOc dataset in Morocco provides a wide variety of statistics at the commune level for all communes in Morocco. This database can be added to both the census and the household survey and in this way all BADOc variables can also be included as “candidate variables” for the the second stage analysis.
- In the Morocco application an important issue arises with regards to the fact that the household survey concept of cluster does not coincide exactly with the census tract in the population census. Rather, the survey cluster, called the “primary unit”, is comprised of a 2-3 census tracts. It is important therefore to construct means in the population census not at the census tract level, but at the “primary unit” level and to use these for analysis in the subsequent stages. For this to be possible it will be necessary to insert a code for the primary unit into the population census based on exactly the same scheme used to construct the primary units in the sampling frame from which the survey’s sampling units were drawn.
 - Note, although it was important to check for common definitions between the census and survey for variables at the household level, the census mean variables can be constructed for any and all variables in the census, whether or not they appear in the survey at all. The point here is that these census means will be inserted into the household survey dataset prior to estimating the consumption models (see below) and thus there will be no issue associated with comparability between the census and survey. Indeed, the more that these types of variables are used in the estimations the more comparability gets imposed, and the less we will need to appeal to our assumptions of comparability and stability.

Guideline for Consumption model estimation

This note is provided by Peter Lanjouw as basic outline of estimating the econometric model of consumption (or income) on those variables determined to be common between the census and survey is estimated. It is important to stress that this estimation should not be approached in the way that economists would generally approach estimating a consumption model. It is clear that even if there is a sizeable set of “candidate” variables determined to be commonly defined between the census and the survey, there are many important determinants of welfare that are unlikely to be included amongst the candidate variables. Thus, the model that will be estimated is likely to suffer from omitted variable bias. In addition, there are a number of variables that are included in the set of candidate variables that would be better viewed not as determinants of economic wellbeing but quite possibly as the reverse: having been caused by consumption or income levels. Hence the estimated model may also suffer from problems of reverse causality. Both of these sources of what is conventionally termed as endogeneity are likely to be present in the model to be estimated. In conventional economic analysis this would be viewed as problematic as it

would hamper the interpretation of the parameter estimates from the model. In our setting, however, these issues are not of concern. The key point to recognize is that our objective in this modeling exercise is not to obtain parameter estimates on regressors that can be readily interpreted and given economic meaning. Rather, our concern is to specify a model that will allow us to forecast consumption as well as possible. That our parameter estimates suffer from omitted variable bias, for example, is entirely desirable because this means that the parameter estimate is capturing not only the correlation between consumption and the specific regressor in question, but is also reflecting the influence of variables that we have not been able to include in the specification. The better we are able to capture the influence of these omitted variables, the better the fit we will get from our model. An important analog of this discussion is that we should not seriously judge the quality of the model we are estimating by scrutinizing the parameter estimates on various regressions and invoking some general notion of whether or not these are “reasonable” based on experience with conventional economic models in other settings. It is not unheard of that our consumption model will get significant parameter estimates on regressors with even the sign being opposite to what one might conventionally expect.

Steps to follow in modeling consumption.

1. Insert census means and ancillary variables (such as the BADOc variables mentioned above) into the household survey dataset for those clusters and communes that occur in the household survey dataset.
2. Construct a series of interaction terms and higher order terms with the household level variables in the survey dataset. For example, household size can be squared cubed, logged etc, education dummies can be interacted with occupation dummies, interactions with province dummies can be created (provided that for each province there are observations in the survey) and so on. It is recommended that such obvious interactions are created at the first stage and that their census-survey distributions are compared.
 - Experience shows that it is often useful to interact as well some household level variables with the census means and BADOc variables.
 - Even when all interacted terms individually pass the census-survey comparison test, it does not follow automatically that in interaction they pass the test as well. Outlier welfare predictions may result from interactions giving extreme results. This occurs especially with interactions with census means and the maximum census mean for the survey is small relative to the maximum census mean in the census. Ideally before including any interaction, a means test should be carried out. In practice many interactions are created during the second stage and most of them are fine. But if suspect results occur this is one place to check. The output from the prediction stage program provides census and survey means as well as their maxima and is a good place to look for outlying interaction terms.

3. The consumption model use OLS to estimate a model of consumption, y , on a selection of household characteristics (not census means or BADOc variables) denoted x .

- A separate model will be estimated for each domain that has been defined in the data preparation stage.
- After having constructed a variety of interaction and higher order household characteristics, and having added census means and BADOc variables at the primary unit and commune level, the list of “candidate variables” is likely to have become very large. Given the interest in estimating separate models for each “domain” the degrees of freedom in the household survey dataset are usually quite limited, and so a subset of variables will need to be selected from among the eligible “candidate variables”.
- We apply a variety of criteria in our effort to settle on a reasonable specification:
 - i. A key indicator to look at when selecting variables for inclusion in the household regression model is their contribution to the overall R^2 of the regression model. It is generally hoped that in the end it will be possible to produce a regression specification for each domain that results in an R^2 that is as high as 0.5 or more. This is often the case, but not always. We have found that the procedure generally becomes quite unsuccessful if the R^2 remains below 0.35 (although the issue really has to do with the degree to which the poverty map will ultimately be disaggregated – the more disaggregated the intended poverty map, the more important that the R^2 be high).
 - ii. Of course, increasing the number of regressors in the model cannot reduce the R^2 (although the adjusted R^2 may well fall). But a second, equally important, criterion to satisfy is that parameter estimates on variables that are accepted for inclusion in the regression should be quite precise (with probability values of, say, 0.15 or less - 0.05 is a good probability value to start with). This criterion tends to reduce sharply the variables that are accepted in the model specification.
 - iii. In general, it is very rare for the final model specification to include more than 40 household level variables. Often successful specifications include less than 20. Parsimony in the specification is desirable as experience suggests that this helps to keep one source of error in the final welfare estimates, the model error, low.
 - iv. It is good practice to check the variance inflation factors of each of the variables. High values (more than 5-10) are an indication of a strong correlation between two variables included in the model. Dropping one of the variables will make the model more robust without affecting the R^2 much.
- Sing model selection procedure. As in many of the statistical software programs that offer a variety of selection procedures for choosing regressors from among a large pool of potential variables on the basis of user-specified criteria. PovMap2 also allows the analyst to select regressors based on their contribution to the R^2 of the model, or

alternatively based on the degree of statistical significance on their coefficients (with the analyst also able to specify whether selection should be forward or backward, and what should be the total number of regressors to be chosen). These procedures are quite helpful in searching for a model specification but on their own they are rarely sufficient to determine the final specification. As mentioned above, the goal is to simultaneously obtain a good fit and precisely estimated coefficients. There is also a general sense that the specification should include variables from the five broad classes described earlier. Experience thus suggests that these packaged selection procedures should be used in association with judgment from the analyst.

- Given that it can be difficult to assess whether a particular model specification is satisfactory or not, due to the various criteria that have to be balanced, there are two additional checks that can be carried out to help with model assessment:
 - i. If the domain for which the model is being estimated is reasonably large, it can be useful to draw a sub-sample from the domain and estimate the model for that sub-sample. Parameter estimates from the estimated model can subsequently be applied to the remaining sub-sample and predicted mean consumption can then be compared to actual mean consumption. If the model is appropriate the “out of sample” predicted mean consumption should be very close to actual mean consumption for this sub-sample.
 - ii. In a similar vein, we can check for problems of “over fitting” by re-estimating the model repeatedly after dropping clusters from the analysis, one at a time, and checking whether parameter estimates on regressors of the model are stable in the face of these slight changes in the sample. If the parameter estimates bounce around as a result of dropping one or other cluster from the sample, this indicates that the model specification has become too closely aligned with the specific structure of the sample. While the usual criteria of R² and precision of estimates may look fine, the model may not be appropriate for applying to census data and predicting consumption for households in the census. Clearly, problems of over fitting are far more likely when degrees of freedom in the regression are close to being exhausted. But experience shows that it is important to check for this also when there still appear to be ample degrees of freedom.

4. Weighting. The question arises, given the complex sampling design that underpins the household survey, whether the consumption model described above should be estimated weighted or not. As a general rule, our experience suggests that weighting can be very valuable in settings where there is wide variation in household expansion factors across primary sampling units. Even where weighting is not strictly necessary, there do not seem to be any major costs associated with weighting (apart from some costs associated with additional programming complexity). So, as a general rule of thumb, it is probably sensible to do all modeling with weighting.

- There exists a simple test for the need for weighting that has been described by Deaton (1997). It takes the following form: interact each of the k household variables in the

proposed specification with the household weight. Re-run the model with this 2*k specification and test on the basis of an F-test whether the interacted household variables are jointly significant. If the test rejects that the parameters on the interacted variables are all jointly zero, then the model should be estimated with weights. As mentioned above, our experience suggests that we rarely fail to reject interacted parameter estimates that are jointly zero, and so weighting is the most common recommendation.

- There are cases where a weighted model is rejected. After a search for a good unweighted model, this gets rejected as well. In such a situation preference is given to the weighted model.

5. Choosing cluster-level variables. So far, our discussion has been focused on the model specification of household level variables. Yet, as already mentioned earlier, it will be important to add to this basic specification also some variables that capture community-level characteristics. The basic point is as follows. There is every likelihood that within a particular cluster (or primary sampling unit) in the household survey there will be significant within-cluster correlation across households in the error term from the basic model of y on household-level x variables. The presence of such an important location effect in the residuals will have the effect of increasing significantly the size of the standard errors that will accompany our poverty estimates. There is thus a real interest to capture this location effect in our model specification. If every cluster in the census occurred in the household survey then the solution would be a straightforward one of estimating the model with cluster-level fixed effects. However, this option is not available to us (far more clusters occur in the census than are covered by the household survey) and so it is important to utilize proxies for these location effects. Our strategy in this regard is to select a number of cluster-level variables calculated from the census and ancillary data sources (means, proportions, variances, etc. – see above), include these in the consumption model, and in this way attempt to capture the location effect. If we are successful in this regard, the remaining share of the overall residual that can be attributed to a location effect will have been driven down towards zero, and our subsequent standard errors on the poverty estimates will be free from this influence.

- There are a very large number of community-level variables that can be calculated from the census and ancillary data sources and inserted into the household survey dataset. These can help to dramatically expand the number of candidate variables to be included in the consumption model. As mentioned above, introducing such variables has the additional attraction of helping to impose comparability between the survey and the census, as these community-level variables are, by construction, identical between the two data sets. However, there is also a potential problem. Over fitting is much more likely to become an issue when one is dealing with community-level variables calculated at the cluster level. Recall that the household survey has a complex sample design. Thus, 10-20 households are typically sampled from a given cluster. Recall, as well, that a given domain over which the consumption model is estimated, may cover somewhere between 100-500 households. This implies that the

consumption model may be estimated across only 10-50 clusters. Clearly, if a large number of census means are added to the specification (these will vary only across clusters) then we may rapidly be in a situation of over fitting - where the model perfectly explains the cluster-level variation in the sample, but does not necessarily provide a reliable basis for extrapolation to a different dataset.

- A useful rule of thumb to avoid over fitting is to include no more than the square root of n regressors in the model. When thinking about household level variables n refers to the number of households in the sample (so that if the domain covers 500 households we wouldn't want to go far beyond 22-23 household level regressors). When we are thinking about census-mean variables n refers to the number of clusters in the sample, not the number of households. In a survey domain of 500 households, where clustering takes the form of 10 households selected per cluster, we have 50 clusters in our domain, and would therefore want to limit our census-mean variables to no more than 7.
- The question thus arises how to select the best census-mean variables for our consumption model, bearing in mind that we may not want to include more than, say, 6 or 7 in our specification and we are faced with a very large set of candidate variables. In approaching this question the key thing to bear in mind is that our aim with the census mean variables is not so much to add to the overall explanatory power of the regression model, but rather to remove as much as possible of the intra-cluster correlation in the residuals on the model estimated on only the household level variables.
- An effective method to select census-mean variables is to employ the following approach:
 - i. First run the basic (weighted) regression of y (log per capita consumption, say) on household and individual level x variables only.
 - ii. Take the residuals from the regression in (i) and regress these (also weighted) on a series of cluster-level dummies. Thus if the domain in question includes, say, 50 clusters, we construct 50 dummy variables, one for each cluster. (Note, suppress the constant term in this regression, or put in only 49 of the 50 cluster dummies in the model).
 - iii. Transpose the row-vector of parameter estimates on the cluster-level dummies into a column vector. This vector will have the dimension of $n \times 1$, where n represents the number of cluster dummies in the regression estimated in (ii) (50 in our example).
 - iv. Regress the column vector of parameter estimates on cluster dummies produced in (iii) on the full pool of candidate census means and other cluster-level aggregates from the census and ancillary datasets. This regression should also be estimated weighted by the sum of household level expansion factors within each cluster. Use the max R^2 selection criterion that is included in the software package to select the 5 or 6 best cluster-level aggregates calculated from the census or obtained from ancillary data sources included among the pool of candidate

variables. (If the max R2 selection criterion is not available in the software package that is being used, trial and error will be required). Once again, although the R2 is our focus of attention, we will not want to use cluster-level aggregates that are not precisely estimated in this regression. So, again, there is a need to balance R2 with precision.

6. Re-estimating the full consumption model. Once cluster-level variables have been selected, the full consumption model, including both household-level and cluster-level variables, can be re-estimated. It may be the case that as a result of adding the cluster-level variables, one or more of the household-level variables comes to lose their statistical significance (perhaps because they were picking up cluster-level variation). So some fine-tuning of the final model may be necessary. Again, the goal here will be to have a specification that includes only significant parameter estimates, and that has an overall R2 that is satisfactorily high.

7. Specifying a model of heteroskedasticity. The inclusion of cluster level variables in the model specification is intended to minimize the degree of intra-cluster correlation in the residuals. It is unlikely that the intra-cluster correlation will have been removed entirely. That which remains will be reflected in the standard errors produced for the final poverty estimates. If the cluster-level variables in the model have been reasonably successful, however, the bulk of the overall residual from the model will now comprise a household specific disturbance term. In the poverty mapping methodology we do not impose the assumption that these household specific disturbances are independent and identically distributed, and allow for heteroskedasticity in these disturbances. Implementation of the poverty mapping module for the final simulation stage of the poverty mapping procedure thus requires not only that a consumption model specification be provided by the user, but also the specification of a model of heteroskedasticity for the household level component of the disturbance term. Settling on the specification of the heteroskedasticity model involves the following steps:

- We first purge the residual on the final consumption model of any remaining cluster-level component. This can be done by regressing this residual on the vector of cluster dummies. We treat the residual of this second model as the household-specific component of the overall residual. To avoid confusion let us denote this second residual as “household error”.
- For each household we square the household error term.
- In order to avoid modeling heteroskedasticity in such a way that could end up predicting a negative squared household error term, we employ a logistic model of heteroskedasticity which involves transforming the squared household error term prior to regressing it on a set of household characteristics. The transformation consists of the following steps.

- i. We identify the highest value of the squared household error term. Multiply this highest value of the squared household error term by 1.05. Denote this number as A.
 - ii. For each household calculate the difference A-squared household error term. By construction this difference will always be positive.
 - iii. For each household calculate the ratio of the squared household error term to the difference calculated in (ii). This term is non-negative and larger; the larger is the squared household error.
 - iv. Taking the natural log of the term produced in (iii) produces the dependent variable for the logistic model of heteroskedasticity.
- There is no theory to guide us in the search for a specification of the heteroskedasticity model. Once again, we are guided mainly by the dual objectives of explanatory power and statistical significance of the parameter estimates. We also want to keep things manageable (and avoid over fitting) by not allowing the specification to be come too large. The rule of thumb governing the number of variables to include in the model (no more than the square root of n) can be applied here too.
 - In terms of settling on explanatory variables, it seems clear that all the explanatory variables included in the consumption model should be candidates. In addition it seems reasonable to include as well predicted (log) per capita consumption from the consumption model. Finally, there is no reason why these explanatory variables should only enter in directly, so we also allow them to interact with all other variables (for example, household size interacted with predicted consumption, with age of household head, and so on). All of these variables can then be considered as valid candidates for inclusion in the heteroskedasticity model.
 - We use OLS to regress our dependent variable (define above) on the full set of candidate variables and select $\hat{\beta}$ that best explain the variation in the dependent variable and that are estimated with a reasonable degree of precision. Once again this selection can be obtained using packaged selection modules that come with the statistical software.

Reference Sources

Elbers, C.; J.O.Lanjouw; P.Lanjouw, 2002; *Micro Level Estimation of Welfare*. Washington DC World Bank Policy Research Paper WPS2911, October.

Elbers, C., J.O.Lanjouw, P. Lanjouw, 2004. *Imputed Welfare Estimates in Regression Analysis*. Washington DC: World Bank Policy Research Paper WPS 3294, April.

Elbers, C., Fujii, T., Lanjouw, P., Ozler, B., Yin, W. 2007. Poverty alleviation through geographic targeting: How much does disaggregation help? *Journal of Development Economics* 83, 198 – 213.

Foster, J., J.Greer, E. Thorbecke, 1984, A class of Decomposable Poverty Measures. *Econometrica* 52:761-66

Fujii, T. 2005, Microlevel Estimation of Child Malnutrition Indicators and its Application in Cambodia. Washington DC: World Bank Policy Research Working Paper 3662, July 2005

Fujii, T. 2003. *Commune level poverty estimates and ground truthing*. Report for UN World Food Programme (mimeo).

Fujii, T., Lanjouw, P., Alayon, S., Montana, L., 2004. *Micro-level Estimation of Prevalence of Child Malnutrition in Cambodia*.

Zhao, Q., 2005. *User Manual for PovMap 1.1a*. Development Research Group. From the World Bank website, August 12, 2006.
<http://iresearch.worldbank.org/PovMap/index.htm>

Zhao, Q., 2006a. *General guidelines for consumption model estimation*. The World Bank (mimeo).

APPENDIX: Technical Requirements

The following paragraphs discuss the technical requirements for implementing hunger mapping using PovMap.

Software and hardware needs

Those wishing to map hunger must understand the technical capacity needed to manage and manipulate data from various sources. Statistical packages, software conversion software, GIS software, sizeable computational space (i.e. Random Access Memory) and storage capabilities are required for efficient construction, management and analysis of the survey and census data sets.

Common Software Packages

Though PovMap 2.0 has the ability to recode variables, it is probably easier to use a common statistical analysis package for the construction and “matching” of the survey and census datasets and for the development of the predictive model. Frequently used standard statistical packages include SPSS, SAS, and Stata. Both SPSS and Stata were used by the Tufts Team to recode and organize the datasets and develop and test the model used to map hunger for the DR, Ecuador, and Panama.

Working with the data

Data sets come in a variety of file types. Each file type (e.g. .dbf, .sav, .dta, .ascii) has its own specifications and limitations. Though many statistical packages are able to read files from different sources, file conversion software can be an invaluable tool. The Tufts team made use of Stat/Transfer² to quickly transfer .dbf and .sav files to Stata’s .dta format.

The large size of census data sets can also cause problems. Census data typically contain millions of cases. Performing numerous recodes and calculations on census data can be time consuming and cumbersome. To facilitate the construction of Ecuador’s data set (which contains approximately 12 million observations), the Tufts Team utilized Stata on a 64-bit mainframe computing cluster. The increased computing power of a mainframe computer greatly improved the efficiency of the recoding process. Users of SAS, SPSS and Stata will be able to perform the same operations on their personal 32-bit personal computers, but performance and speed will be limited by each computer’s processing and memory capabilities.

² <http://www.stattransfer.com/>

System Requirements

Due to the computational power needed to manage large data sets, manipulate geographic data, and run PovMap, the Tufts Team recommends the following system requirements for computers. Though most processes can be completed using less robust systems, the requirements given below will greatly reduce the time needed to run calculations on large data files.

- 2GHz or greater processor
- 1GB or greater Random Access Memory (RAM)
- 60 GB or greater hard drive

Appendix A: Expression , Operator and Function

1. Mathematic Functions

ABS(*expr*)

Returns the absolute value of *expr*.

ACOS(*expr*)

Returns the arc cosine of a specified *expr*. The values of *expr* can range from -1 through +1. The values returned by ACOS() range from 0 through pi (3.141592) in radians.

ASIN(*expr*)

Returns in radians the arc sines of *expr*. The values of *expr* can range from +1 through -1, and the values ASIN() returns can range from -pi/2 through +pi/2 (-1.57079 to 1.57079).

COS(*expr*)

Returns the cosines of *expr*, which are specified in radians.

Count()

Returns the number of observations. Value repeats to full-length of vector.

EXP(*expr*)

Returns the exponential values of *expr*. On overflow, the function returns missing and on underflow, EXP() returns 0.

LOG(*expr*)

Return the natural logarithms of *expr*. For non-positive value, LOG() returns a missing.

MAX(*expr*)

Returns the highest value in *expr*. Ignores missing.

MEAN(*expr*)

Returns the arithmetic average of all non-missing value.

MIN(*expr*)

Returns the smallest value of *expr*. Ignores missing.

NA()

Returns a vector contains all missing values.

PCTL(*expr*, *p*) where *p* is a constant number, $0 \leq p \leq 1$.

This function returns a constant vector (all elements have the same value) indicates the *p* percentile of *expr*.

Example: $\text{IIF}(X > \text{PCTL}(X, 0.99), \text{NA}(), X)$ will set value of *X* to missing for cells larger than the 99 percentile of *X*.

SIN(*expr*)

Returns the sines of *expr*, which are specified in radians.

SQRT(*expr*)

Returns the square root of the specified expression *expr* (cannot be negative)

SUM(*expr*)

Returns the total of values of all elements in specified expression *expr*. The result value repeats in full length as *expr*

WMean(*expr*, *weight*)

Mean of *expr* with explicitly specified weight.

WSum(*expr*, *expr_w*)

Sum of *expr* with explicitly specified weight.

2. Aggregation Functions**CCount()**

Returns the number of observations (or, the size) of each cluster.

CCountBy(*numDigits*)

DRAFT for comment; not for citation

Returns the number of observations (or, the size) of each new group formed by truncating cluster-ID (truncated once when you created new project) numDigits more digits off.

CCountOn(*expr_base*,*numDigits*)

Returns the number of observations (or, the size) of each new group formed by truncating the eval result of *expr_base* numDigits digits off.

Example: CcountOn(DIST,0)
CcountOn(\$ID\$/100,0)
CcountOn(Gender,0)

CErr(*expr*)

CErrBy(*expr*,*numDigits*)

CErrOn(*expr*,*expr_base*,*numDigits*)

Similar to CCOUNT functions, return the standard error of each group.

CFirst(*expr*)

CFirstBy(*expr*,*numDigits*) where *numDigits* is a non-negative integer constant.

CFirstOn(*expr*,*expr_base*,*numDigits*)

Repeats the first evaluated *expr*'s value of each cluster (group) within that cluster (group). You may re-group observations by truncating cluster-ID numDigits MORE digits off, or, by truncating the eval-result of *expr_base* numDigits digits off.

CGet(*condi_expr*,*expr*)

If the *i*th value of *condi_expr* is true (1), gets the *i*th value of *expr* and repeats it within cluster. If there is no 1 at all within a cluster, the cluster of result will be filled with all missings. If there are more than one 1's of *condi_expr* found within a cluster, this function takes the 1st one and ignore the others.

CMax(*expr*)

CMaxBy(*expr*,*numDigits*)

CMaxOn(*expr*,*expr_base*,*numDigits*)

Similar to CCOUNT functions, return the maximum value of each group.

CMean(expr)

CMeanBy(expr,numDigits)

CMeanOn(expr,expr_base,numDigits)

Similar to CCOUNT functions, return the mean value of each group.

Example: CMeanBy(X+Y,2)
CMeanOn(X+Y,DIST,2)

CMin(expr)

CMinBy(expr,numDigits)

CMinOn(expr,expr_base,numDigits)

Similar to CCOUNT functions, return the minimum value of each group.

CSum(expr)

CSumBy(expr,numDigits)

CSumOn(expr,expr_base,numDigits)

Similar to CCOUNT functions, return the sum of all values in each group.

WCMean(expr,expr_w)

WCMeanBy(expr,expr_w,numDigits)

WCMeanOn(expr,expr_base,expr_w,numDigits)

This set of functions are different from their un-weighted counter part CMean family by applying explicit weight.

Example: WCMean(X+Y,W)
WCMeanBy(X+Y,W,2)
WCMeanOn(X+Y,DIST,W,0)

WCSum(expr,expr_w)

WCSumBy(expr,expr_w,numDigits)

WCSumOn(expr,expr_base,expr_w,numDigits)

Similar to WCMean functions, return the mean of all values in each group.

CSerial(numDigits)

CSerialBy(expr,numDigits)

DRAFT for comment; not for citation

CSerial generates a sequential number starting from 1 for each group whose shifted ID (shifted ID by *numDigits*) is the same. In CSerialBy, the sequential number starts from 1 in each group whose shifted ID is the same, then increase by 1 when the value of *expr* changes. Both functions are designed for simplifying the ID structure.

3. Testing Functions

IsDup(*expr*)

Returns a 0-1 vector, in which 1 indicates the element is a duplicate of the prior element of *expr*.

IsFirst(*expr*)

Returns a 0-1 vector, in which 1 means the value of element of *expr* is the 1st one in a group of duplication. If an element is different from the prior and the next, also returns value 1. So, the result actually is the 1's complement of IsDup(), i.e., with the same *expr*, where IsDup returns a 1, here we get a 0; a 0, we get a 1, and vice versa.

IsMissing(*expr*)

Returns a 0-1 vector. 1: if the element of *expr* is a missing

GMask()

Returns the global mask as a 0-1 vector

LCount() 'L' stands for 'Left'

ICount() 'I' stands for 'Inner'

RCount() 'R' stands for 'Right'

Used when two data array are linked. The one on the left is always at more detail level than the one on the right. (such as individual \leftrightarrow household, or household \leftrightarrow village. The ID on the left and right data array may not be completely matched. LCount will return the count of each ID group based on the 'left data array'. RCount will return the count of each ID group based on the 'right data array'. ICount returns the count of ID existed both on the 'left data array' and the 'right data array'.

4. Conditional Functions

Bounding(*expr*,*L_bound*,*R_bound*) both bounds must be constants.

For each *v* in *expr*, return *L_bound* if $v < L_bound$; return *R_bound* if $v > R_bound$; otherwise, return value *v*.

Categorize(*expr*)

Cast the *expr* to be a categorical variable

Encode(*expr*, *Value_Set*)

Returns a categorical variable with re-grouped values of *expr* according to *Value_Set* (see definition below)

IIF(*condi_expr*, *true_expr*, *false_expr*)

Returns *true_expr* when *condi_expr* is true (value 1), or *false_expr* otherwise.

Example: IIF(IsMissing(X), X1, IIF(Y1 <> 0, 100*X1/Y1, 0))
IIF(Gender="Male", 65, 55)

Int(*expr*)

Returns the integer part of each element in *expr*.

Round(*expr*, *decimalPlaces*)

Returns *expr* rounded to a number of decimal places, specified by *decimalPlaces*. If *decimalPlaces* is negative, ROUND() returns a whole number containing zeros equal in number to *decimalPlaces* to the left of the decimal point.

Example: ROUND(17.5274, 0), result is 18
ROUND(17.5274, 2), result is 17.53
ROUND(17.5274, 1), result is 17.5
ROUND(17.5274,-1), result is 20

Seg(*expr*, *start*, *len*)

expr is supposed to be a multi-segment key, this function returns one of its *len* digits segment start at *start* (from right of the *expr*).

SetMissing(*expr*, *Value_Set*)

Returns a vector in which any value of *expr* beyond the range (specified by *Value-Set*) will be set to a missing, but the value within range will remain untouched. NOTE: There should be only one group in *Value-Set*

ValueOf(expr)

Casts the type of expr from categorical variable to numerical variable.

IIF($X \geq 3 \ \& \ X < 12$, X, MissingValue()) can be replaced as SetMissing(X, {{3,12}}) or $X @ \{[3,12]\}$, but

IIF($X \geq 3.2 \ \& \ X < 11.97$, X, MissingValue()) does NOT have the same style equivalent substitution like SetMissing(X, {{3.2, 11.97}}) or $X @ \{[3.2, 11.97]\}$, since decimal is invalid in

a Value-Set

IIF($X > 11.97$, 11.97, IIF($X < 3.2$, 3.2, X)) is equivalent to Bounding(X, 3.2, 11.97) except the latter is simpler and faster.

5. Operators and its Precedence

Operators of PovMap2 are listed fellow from hight to low:

(,)	Forces a different precedence on the expressions
(unary)+, -, !(~)	Positive, negative, logical NOT ↓
^	Power
*, /, %, &	Multiplication, division, modulo, logical AND
+, -,	addition, subtraction, logical OR
>, >=(=>), <, <=(=<)	comparison operators
=(or ==), !=(or <>, ><)	equality, inequality

6. Value set

Value set defines the conditions for generating sequential integer 1,2,3,... Definitions are limited with semi-column. It can be used in the place where the procedure language use a **switch** statement (in case of c,c++,Java) or **select case** statement as in the VB. The formal definition can be specified as follows:

```
Value-Set ::= {Group[; Group]}
Group ::= Enumerative-group | Range-group
Enumerative-group ::= Value [, Value]
Range-group ::= >Value Range group
Range-group ::= >=Value
Range-group ::= <Value
Range-group ::= <=Value
```

Range-group ::= ...

Range-group ::= LBracket Value,Value RBracket

LBracket ::= (

LBracket ::= [

RBracket ::=)

RBracket ::=]

Value ::= Integer

Value ::= yyyy/mm/dd

y ::= 0..9

m ::= 0..9

d ::= 0..9

It is user's responsibility to ensure the definition of groups cover the complete domain, values not fall in any groups will become a missing. When used in the SetMissing function, there should be just one group and that group could contain multiple range-groups and/or enumerative-groups.

Range-groups should not overlap each other, but enumerative-groups can overlap range-groups, i.e. enumerative groups have higher priority than range groups.

Examples: `EnCode(x, {1,3,5; 2,4,6})` will convert value 1,3,5 to 1 and 2,4,6 to 2

`EnCode(x, {<0;[0,10];(10,20);>20})` will convert any value less than 0 to value 1, value 0 to 10 to 2, values larger than 10 and up to value 20 into 3, last, any value greater than 20 into 4.

`Encode(X, {<0;[0,10];...;>=50})` is equivalent to
`Encode(X, {<0;[0,10];[10,20];[20,30];[30,40];[40,50];>=50})`

`Encode(X, {[1,3];...;[10,15];...;>=30})` is equivalent to
`encode(X, {[1,3];[3,5];[5,7];[7,9];[10,15];[15,20];[20,25];[25,30];>=30})`

`Encode(X, {15,[1,10];(10,20)})` will convert all value 15 into 1, all values between 1 and 10 (inclusively) into 2, any value larger than 10 and less or equal to 20 as 3, and anything else into missing.

`Encode(X, {>0})` is equivalent to `IIF(x>0, 1,NA())`

`Encode(Birthday, {[1970/04/01, 1990/09/30]})`