I. INTRODUCTION

1. Statistics agencies collect surveys and publish results. Sometimes surveys contain "mistakes" and "missing values" which should be detected (editing) and corrected (imputation). This editing-and-imputation operation is a very complex process since it attempts to correct errors in the data whose true values are unknown. In addition the modern surveys often contain enormous amounts of data. So it is crucial the use of advances in technology in the whole process, particularly automatic tools based on Computer Science (computers, programming languages, databases, graphic environments, etc).

2. To contribute to the above lack of technology we have designed and implemented a new software called TEIDE (Techniques for the Editing-and-Imputation of Statistical Data). It is multi-platform (running with Linux, Windows, …), free (not commercial library or compiler is required to rebuild an executable from the source), based on XML formats (and also working with other standard formats from Microsoft Access and ORACLE), working on social and economic surveys (i.e., dealing with quantitative and qualitative variables) and also working on periodically collected surveys. TEIDE has been extensively tested on surveys concerning households and individuals with reading of data through databases Microsoft Access and Oracle, and it has been already presented in the previous edition of this conference. Now we mainly present a new independent module, TEIDE_OLA, which allows treating surveys collected periodically and saved in XML files.

II. TEIDE_OLA

A. Introduction

3. TEIDE_OLA works on economic data mainly through variants of methods based on numerical regression. All methods implemented in TEIDE_OLA have been motivated by working on data from the survey of the “Shopping Cart” in Canary Islands. In these surveys registers refer to the prices of a given product in a given market, collected in different months. Imputing a wrong or missing value takes into account other prices.

B. Development

4. This section addresses the methodology of editing and imputation generally applied to the specific case of the system TEIDE_OLA with their particularities.
4.1. Variables
The variables to be checked and potentially corrected by TEIDE_OLA are of numerical type. In other words, the tool has been designed to work with quantitative variables. Still, two variables are managed: the “price” is a quantitative variable and the “quantity” is a qualitative variable.

4.2. Process reading of data
The process of reading data by TEIDE_OLA consists in first reading a metafile (indicate the file containing the data), and second checking whether there are products which needs modifications. For each month we identify all the different products. This results in a table with multiple tabs (as many as months). This table shows all the products in this month with information concerning this product.

4.3. The process of editing
For every product the information on price and quantity is obtained for each month: When we do not have such information in a given month it is considered a missing value. The detection of inconsistencies (in quantities and/or prices) is also done.

4.4. The process of imputation
The process of imputation in TEIDE_OLA operates on the basis of the information generated by the editing process. It is divided into three phases that are carried out sequentially, one after another. The first of these phases is the imputation of quantities, the second is the imputation of prices, and the third is the imputation of missing values.

4.4.1. Imputation of quantities
The quantities equal to zero are always changed by default.

4.4.2. Imputation of prices
It imputes each of the prices taking into account a linear regression of all the price/quantity of product without counting the wrong price/quantity. If a price is within a given percentage (defined through the user options or a default 15%) with respect to the price estimated by the regression, the price is considered as right. Otherwise, it is incorrect and it will be estimated in another phase.

For the calculation of the linear regression has used the following formula:

\[ y = a + b \cdot x \]

Taking into account the following

<table>
<thead>
<tr>
<th>X</th>
<th>Y price/amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>y1</td>
</tr>
<tr>
<td>x2</td>
<td>y2</td>
</tr>
<tr>
<td>x3</td>
<td>y3</td>
</tr>
<tr>
<td>x4</td>
<td>y4</td>
</tr>
</tbody>
</table>

For the linear regression on a given month, we make use of the values of each product in other months.

4.4.3. Imputation of missings
At the time of changing a missing value we take into account the month as follows:
- If the missing value is in the first month, it is not imputed (since it is considered to have no past and does not exist yet).
- If the month is not the first month, but in previous months is also a missing value, it is not imputed (since it is still considering that such product does not exist).
- If the missing to be imputed is in an intermediate month (i.e it has an earlier month and a latter month) with values (no missing), we then make a linear regression with the previous point and the following point (estimated prices) and we impute the value.
- In the case that do not have a value for a month, but we have the values in the past two months, we imputed the price.
- It also takes into account that if a product has value empty in the three or more recent months, the values in these months are not changed since it is considered that the product will disappear.

The linear regression is calculated as explained above, except that now it only uses two values.

4.5. Display of errors
Once the imputation of prices has finished, the modifications are displayed in red.

4.5.1. Prices negative
Are those products in which a month has a price lower than zero. They do not use to appear in practice, but in case they are displayed then the parameters for the imputation must be changed and the process repeated.

4.5.2. Pending refusal
We the price decreases we also display a warning.

4.5.3. Prices different
Are those products that do not meet the formula:
\[
\frac{\text{price}_{\text{max}} - \text{price}_{\text{min}}}{\text{price}_{\text{min}}} \geq X
\]

4.5.4. Quantities different
Are those products that do not meet the formula given below:
\[
\text{amount}_{\text{max}} - \text{amount}_{\text{min}} > X \times \text{amount}_{\text{min}}
\]

III. TEIDE2

A. Introduction

5. TEIDE2 was designed to work with the survey of Income and Living Conditions in the Households in Canary Islands (EICVHC). The units of analysis are households resident in Canary Islands who inhabit family homes: people in a family home sharing a common budget for food and housing costs, and individuals over 16 years household members.

This application performs the editing-and-imputation of data obtained from this survey.
B. Development

6. The errors in the data can be detected specifying certain restrictions that must be satisfied by the individual records, i.e., the data for individual informants. These restrictions are called rules of consistency or edits.

6.1. Edits
    TEIDE2 works with the total edits divided into three subgroups:

6.1.1. Edits Range
    An Edit of range is no more than the specification of the range that can take a particular variable.

6.1.2. Filter Edits
    A filter edit are motivated by the need to put certain "keys" issues dependent. In this way, the filter edits are designed with the following structure: there is a condition, such that is not met, the variable can only take a value, and this value is "Not applicable".

6.1.3. Edits Explicit
    Structure of IF – THEN type.

6.2. Variables
    The variables are of numerical type, and within them, basically the tool is designed to work with qualitative variables.

6.3. The process of editing
    It is divided into two phases inter-related.

6.3.1. Editing ranges
    There edits are the set of range values and the whole edits of filters. Suppose that exist \( n \) records and \( m \) variables. The records has been given by the set \( M = \{ m_i \} | i = 1, ..., n \). The variables are given by the set \( V = \{ v_j \} | j = 1, ..., m \). Each variable \( v_j \) has an associated edit range \( \gamma_j(m_i) \) and a filter edit \( \delta_j(m_i) \). These edits are functions that return true or false depending on whether the variable meets the edits or not for the values of the variable in the record. The function that expresses the evaluation of range is \( r_j(m_i) \).
    The process of editing ranges would be as follows:
    \[
    \forall i = 1, ..., n \\
    \forall j = 1, ..., m \\
    r_j(m_i) = true \leftrightarrow \gamma_j(m_i) \text{ and } \delta_j(m_i) = true
    \]
    This generates a subset \( M_1 \subseteq M \) such that \( M_1 = \{ m_i \} | r_j(m_i) = true \forall i = 1, ..., n, \forall j = 1, ..., m \). That is, this subset \( M_1 \) is the subset of those records that have all the correct ranges.
    In turn, the complementary set \( M_1 \) is the set of records with an error. There is a set of variables erroneous associated with each record in \( M_1 \). We denote by \( ve_i(i) \) the list of variables with a wrong value in a record \( m_i \), with \( i = 1, ..., n \).

6.3.2. Editing explicit edits
    The objective is to evaluate all the explicit edits, i.e., the edits which are neither range values nor filters. Suppose that there are \( p \) explicit edits. The set of edits explicit is given by the set \( E = \{ e_k \} | k = 1, ..., p \). Each edit has associated a function \( \psi_k(m_i) \), which determines whether the record \( m_i \) meets the edit explicit \( e_k \). Let us denote by \( t_k(m_i) \) the final evaluation function in edits explicit.
    The process of editing edits explicit would be as follows:
    \[
    \forall i = 1, ..., n \\
    \forall k = 1, ..., p \\
    t_k(m_i) = true \leftrightarrow \psi_k(m_i) = true
    \]
    This generates a subset \( M_2 \subseteq M \) such that \( M_2 = \{ m_i \} | t_k(m_i) = true \forall i = 1, ..., n, \forall k = 1, ..., p \). In other words, this subset \( M_2 \) is the subset of those records that comply with all the edits explicit.
In turn, the complementary package $M_2$ is the set of records that have some error. There is a set of edits erroneous associated with each record. The function that returns the set of union of all variables present in the whole of erroneous edits to a record $i$ is denoted by $\text{ve}_e(i)$, with $i = 1, \ldots, n$.

6.4. The process of imputation
The general method of action is the imputation by registration donor.

6.4.1. Grafo from edits
The process of imputation makes use of a graph of edits. This graph is given by $G = (N, A)$, where $N = V$ and $A = \{ a_{ij} \mid \exists e \in S : v_i, v_j \in e; i, j = 1, \ldots, m \}$, with $S = \Gamma \cup \Delta \cup E$, where $\Gamma$ is the set of edits in range and $\Delta$ is the set of edits of filter. That is, the set of vertices of this graph is the set of variables and there is a ridge that links two vertices (variables) provided that these two variables exist in a same edit.

6.4.2. Records donors
To carry out the imputation, is necessary to have a set of records donors, i.e., a set of records entirely correct, that have met all the edits, both of range, filter as explicit.
This set of records is given for $M_d \subseteq M$ such that $M_d = M_1 \cap M_2$.

6.4.3. Variables basic and extended to impute
As opposed to the previous case, there is a set of records that contain some error. These records should enter into the process of charging and be "arranged". This set of records is given by $M_e \subseteq M$ such that $M_e = M_1 \cap M_2$.
Each record that belongs to this set has an associated set of basic variables to blame. The variables to impute are those who leave directly from the edition of ranges and edits explicit, i.e. those variables erroneous directly from the editing stage. For a record $m_i$, the set of basic variables is given by:
$$v_{bas}(i) = \text{ve}_e(i) \cup \text{ve}_r(i)$$
The variables extended to impute are those variables emerging from the union of the components related to the basic variables to blame. This is achieved in a set of variables which are interrelated, something very important for the algorithm of imputation that is described later. Suppose that the function $cc(j)$ returns the set of variables in the related component for the variable $v_j$. For a record $m_i$, the set of variables extended is given by:
$$v_{ext}(i) = \bigcup_{v_j \in cc(v_j)} v_{bas}(i)$$

6.4.4. Distance internal and external
It is necessary to define a measure of distance to compares the closeness of a registry. In this case, we identify two distances, arising from the concepts of basic variables to impute and variables extended to blame.
The interior distance is the number of fields that differs a registry of another taking into account the basic variables of attribution. Denoting $d_{int}(i,j)$ the function that return the interior distance between the records $m_i$ y $m_j$:
$$d_{int}(i,j) = |v_{bas}(i)| - \sum_{v_k \in v_{bas}(i)} \text{eq}(m_i(v_k), m_j(v_k))$$
where the function $\text{eq}(x, y)$ returns a 1 if $x$ and $y$ are equal, and returns a 0 in another case.
The distance outside is the number of fields that differs a registry of another taking into account the variables extended of imputation. If $d_{ext}(i,j)$ denotes the function that return the exterior distance between the records $m_i$ y $m_j$:
$$d_{ext}(i,j) = |v_{ext}(i)| - \sum_{v_k \in v_{ext}(i)} \text{eq}(m_i(v_k), m_j(v_k))$$
where the function $\text{eq}(x, y)$ returns a 1 if $x$ and $y$ are equal, and returns a 0 in another case.

6.4.5. Imputation Algorithm
We now describe the algorithm of imputation for a record $m_i$. 
After having calculated related components, and internal and external distances to all records, we must seek a registration $m_j$ that will be our donor for the record $m_i$. The record donor meets that $m_j \in M_d$. It must also implement that its distance is the minimum towards the record $m_i$. In the current implementation it is being used the distance external, but this approach may vary. That is, the registration donor $m_j$ would:

$$m_j = \min_{m_k \in M_d} \{d_{ext}(m_i, m_k)\}$$

We already have the record which "will donate" values to our record incorrect. Now the problem that arises is: How many values should donate the donor record to make the incorrect record meets the restrictions imposed?

The donation can be divided into two phases. In a first phase, we fix the incorrect record testing all combinations of basic variables to blame. It is important to note that these combinations are only carried out if the number of basic variables to impute is below a threshold defined. We have a number of basic variables to impute $t = |v_{bas}(i)|$, we check the following combinations:

$$\begin{pmatrix} t \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

If at any of these combinations complies with all the edits defined, the imputation is done. If not, turn to the second phase of algorithm.

In this second phase we are going to donate one to the rest of variables extended to charging that have been calculated for the record $m_i$. The order in which these variables are donated is not random, but are ordered from greater to minor appearance in the edits. In other words, if the function $ed(j)$ returns the set of edits in which appears the variable $v_j$, we try to donate in each iteration the variable to fulfil the following condition:

$$\max_{v_k \in \text{set}(i)} \{|ed(v_k)|\}$$

In the worst cases, this process ends by donating the whole of basic variables to impute and the whole set of variables extended to attribute, thus ensuring that the incorrect register turns to be correct.
IV. DISPLAY OR RESULTS

7. Once we have finished editing and imputation of the survey data, the application shows the results as well as the statistics computing, i.e. the errors detected and the corrections made. At this stage the realization of the tasks of execution is analysed and the data obtained. It produces a view to decide whether the results have a level of acceptable quality. It analyses the coverage obtained, the level of non-response, the accuracy of the responses, the mistakes produced during recording, the level and distribution of errors detected and the charges made. Both TEIDE_OLA as TEIDE2 have been tested and debugged with data "almost real", which represent a very small sample survey of the Shopping Cart and survey of Income and Living Conditions in the Households in Canary Islands respectively.

Teide_Ola: Statistics obtained from the editing and imputation of data, as well as histograms of them.
Teide2: Statistics obtained from the editing and imputation of data, as well as histograms of them.

V. CONCLUSIONS AND FUTURE

8. TEIDE_OLA is intended to simplify the work of a person devoted to debug a survey of regular nature, "the survey of the shopping cart". In this survey each record is a product whose price and format is evaluated each month. The application is very simple to use. It is very graphical in all steps. Basically, it opens a metafile with data, displays the prices and quantities of all products and months on screen, by means of a table, emphasising prices that are taken by different products in different months, allowing navigate through the table in a comfortable and enjoyable way.

9. The user may perform a manual imputation (i.e., he/she can insert the new value through the keyboard). In the case that he/she wants to do the automatic charging, the application will produce -for every price to impute- the new values to take, as well as the option to the user for doing or undoing an action.

10. This application may do everything automatically, with the necessary cooperation and supervision of statisticians. It also admits to consider different possible corrections, as well as to provide easy movement through the changes made by touching two keys on the keyboard.

11. TEIDE2 has been successfully used for debugging the survey of Income and Living Conditions in the Households in Canary Islands. The application shows data in tables with easy access. It also allows that the entire editing-and-imputation is carried out step-by-step, so the statistician may follow all the details.

12. The application can be improved:
   - TEIDE_OLA: It will read data from Oracle; now it only takes data from XML.
   - TEIDE2: It will read data from XML; now it only takes data from Microsoft Access or of Oracle. We also need to reduce the time for processing.
13. We thank ISTAC (Instituto Canario de Estadística) and IEA (Instituto Andaluz de Estadística) for supporting our work on TEDE. We also thank the research project MTM2006-14961-C05-03.