Automatic data editing functions for establishment surveys

Jeroen Pannekoek, Sander Scholtus, Mark van der Loo*

Abstract

Data editing is arguably one of the most resource-intensive processes at NSIs. Forced by ever increasing budget pressure, NSIs keep searching for more efficient forms of data editing. Efficiency gains can be obtained by selective editing, that is limiting the manual editing to influential errors, and by automating the editing process as much as possible. Besides making the data editing process more efficient, there is also a need for increasing the cost-effectiveness of designing and implementing data editing systems. In this paper we propose a hierarchical decomposition of the data editing process into six different types of tasks, called *statistical functions*. Identifying the in- and output parameters of these abstract functions allows one to move towards a modern approach to process design, based on reusable components that connect in a plug-and-play manner.

Key words: automatic editing; generalised systems; process design

1 Introduction

The quality of raw data available to National Statistical Institutes (NSIs) is rarely sufficient to allow for the immediate production of reliable statistics. As a consequence, NSIs often spend considerable effort to improve the quality of micro-data before further processing can take place.

Statistical data editing encompasses all activities related to the detection and correction of inconsistencies in micro-data, including the imputation of missing values. Data editing has traditionally been performed manually by data editing staff with subject-specific expert knowledge. The manual follow-up of a large number of detected inconsistencies is, however, very time-consuming and therefore expensive and it decreases the timeliness of publications. Therefore, several approaches have been developed to limit this very resource-consuming manual editing.

One approach is selective editing (Latouche and Berthelot, 1992). This is an editing strategy in which manual editing is limited or prioritised to those errors where this editing has a substantial effect on estimates of the principal parameters of interest. This strategy can be successful because it has been well-established (see the review by Granquist and Kovar (1997)) that for many economic surveys only a minority of the records contains influential errors that need to be edited.

An alternative route to reducing manual editing is to perform the editing automatically, which is the main focus of this paper. Automatic editing is not a single method but consists of a collection of formalised actions that each perform a specific task in the overall editing process. Some well-known tasks that are performed in automatic editing are the evaluation of edit rules to detect inconsistencies in the data, the localisation of fields that cause these inconsistencies, the detection and correction of systematic errors such as the well-known thousand error, and the imputation of missing or incorrect values, see e.g. De Waal et al. (2011). Once implemented, automatic editing is fast, uses hardly any manual intervention and is reproducible. For reasons of efficiency, it should therefore be at least an important part of any editing application, with selective manual editing as a necessary addition for errors that cannot be treated automatically.

*Statistics Netherlands, P.O. Box 24500, 2490 HA The Hague, The Netherlands. E-mail: jpnk@cbs.nl, sshs@cbs.nl, mplo@cbs.nl. The views expressed in this article are those of the authors and do not necessarily reflect the policies of Statistics Netherlands.

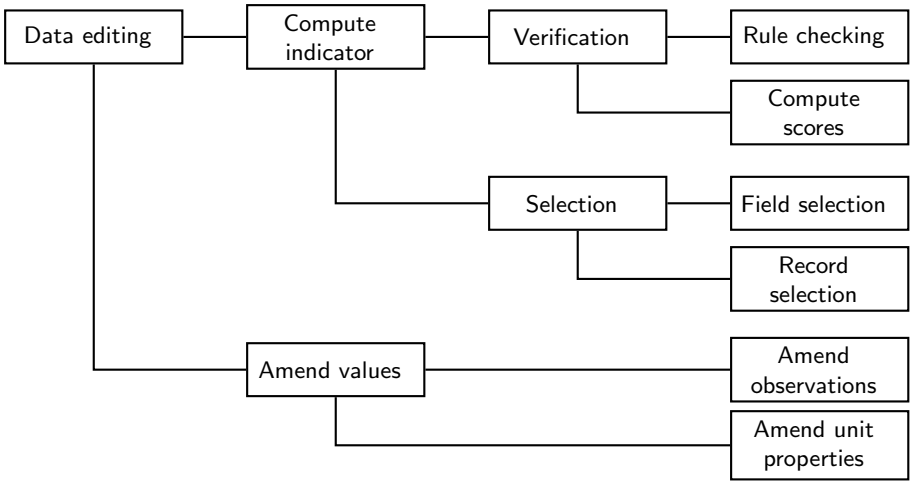

Figure 1: A taxonomy of data editing functions. Each data editing function has its own minimal input-output profile which determine how they may be combined in a data editing process (Table 1).

Besides making the data editing process more efficient, there is a need for increasing the cost-effectiveness of designing and implementing data editing systems. In this paper we propose a hierarchical decomposition of the data editing process into six different task types, called *statistical functions*. This view of the overall process builds on previous work by Camstra and Renssen (2011) and Pannekoek and Zhang (2012) and is treated in much more detail in Pannekoek et al. (to appear). Identifying the in- and output parameters of these abstract functions allows one to move towards a modern approach to process design, based on reusable components that connect in a plug-and-play manner. It also allows the definition of evaluation functions for each task with which the editing process can be monitored task-by-task and alternative methods or parameter settings for each of the tasks can be compared to optimise the overall process.

2 A taxonomy of data editing functions

A typical data editing process consists of a number of automated editing steps and a possibility for manual intervention on selected records. Each part of such a process has its own input, output and control parameters that influence how it can be combined with other steps to build a full process.

To design, compare and evaluate data editing processes it is useful to have a common terminology for the *types of activities* that are instrumental in realising the end result of a data editing process. In this section we describe a decomposition of the overall data editing process in a taxonomy of statistical functions that are characterised by the kind of task they perform and the kind of output they produce. The effects of these statistical functions can be evaluated by inspecting their characteristic output.

A statistical function describes *what* type of action is performed but leaves unspecified *how* it is performed. To implement a statistical function for a specific data editing task (a process step) a method for that function must be specified and configured. It should be noted that the same statistical function can, and often will, be implemented by several methods even within the same application. For instance, a number of different methods for detecting erroneous fields will often be applied one after another so as to catch as many errors as possible. This may be seen as the repeated application of the function *field selection* (see below).

An actual implementation of a data editing process can now be seen as a collection of implementations of statistical functions (process steps). The choice of methods to be used in the process steps and the order in which the process steps are executed will depend on the properties and requirements of the specific application at hand, see Pannekoek and Zhang (2012) for a discussion of these choices.

Table 1: The minimal input and output for data editing functions. The input data consist of N units. Each unit is subject to K mandatory edit rules.

Function	**input**	**output**
Rule checking	data, rules	$N \times K$ edit failure indicator
Compute scores	data	N-vector of score values
Field selection	data, rules	field selection indicator
Record selection	data	N-vector of subset indicators
Amend observations	data	data
Amend unit properties	unit properties	unit properties

In Figure 1 we decompose data editing tasks hierarchically, in three levels, into ultimately six low-level statistical functions. At the first level of the decomposition we distinguish between functions that leave the input data intact (*compute indicator*) and those that alter the input data (*amend values*). At the second level, functions are classified according to their purpose. We distinguish between indicators that are used to verify the data against quality requirements (*verification*) and indicators that are used to separate a record or dataset into subsets (*selection*). *Verification* functions are separated further into functions that verify hard (mandatory) edit rules (*rule checking*) and functions that compute softer quality indicators (*compute scores*). The *selection* function allows for different records (*record selection*) or different fields in a record (*field selection*) to be treated differently. There is no separation based on purpose for the *amendment* function; *amendment* functions are only separated into functions that alter observed values (*amend observations*) and functions that alter unit properties (*amend unit properties*) such as classifying or frame variables. This may be interpreted as a decomposition based on a record-wise or field-wise action.

The lowest-level statistical functions defined here each have their own minimal input-output specification which is independent of the chosen statistical method or implementation thereof. This facilitates the building of an overall editing system with a connected set of editing functions and the definition of performance indicators based on the minimal output. Table 1 denotes this set of minimal in- and output parameters for every low-level statistical function of the taxonomy. Any extra in- or output parameter used in a particular process will be related to the specific method chosen to implement a function.

Below, the six lowest-level data editing functions are further clarified by giving examples of well known data editing methods for each of the functions.

Rule checking. This verification function checks, record by record, whether the value combinations in a record are valid. The valid values are defined by a set of edit rules which specify the admissible values. For establishment survey data, many of these rules take the form of linear equalities or inequalities. For example *turnover* $\geq$ *0*, *profit*+ *total costs - turnover = 0* and *total costs = employee costs + costs of purchases + other costs*. By checking each edit rule for each record, we obtain a $N \times K$ failed edit indicator matrix, with N the number of records and K the number of edits.

Compute scores. A score function computes a quality measure for a record or field. In selective editing scores are used for estimating the potential effect that editing a record may have on estimated totals or other parameters of interest. Each score function results in an N-vector of quality measures. The output of score functions is often input for record selection functions, such as selection for manual review (selective editing).

Field selection is used to point out fields in records that need a specific treatment. Examples of methods include detection of systematic errors such as thousand errors and other errors with a known cause such as typing errors or rounding errors (see, Scholtus (2009), Scholtus (2011)). Apart from these so-called generic systematic errors there are also domain specific systematic errors. These pertain to specific variables and are usually detected and corrected by simple if-then type of rules. Another important example of *field selection* is Fellegi and Holt's method for error localisation of random

errors (Fellegi and Holt, 1976). These selection functions each yield a field selection indicator, which is used to decide on a specific follow-up action; e.g. dividing by 1000 for thousand errors, imputation for random errors and appropriate corrections for typing and rounding errors. These follow-up actions themselves are no *field selection* functions but *amendment functions*.

Record selection aims to select records from a data set that need separate processing. This can be done automatically, for example by comparing the value of a score function to a threshold value to select records for manual review, or by manual methods such as sorting on a score function, reviewing aggregates, and graphical analyses.

Amend observations. This function aims to improve data quality by altering observed values or by filling in missing values. The follow-up actions mentioned under *Field selection* are all amendment methods since they actually modify the observed data. Also the imputation of originally missing data is seen as an amendment method since it improves the data by changing missing values in non-missing ones by filling in predictions. Since imputed values will not always be consistent with the edit rules, imputation can be followed by an amendment step in which the imputed values are adjusted, as little as possible, to ensure this consistency. The amendment function can also be performed manually, for example by data editing staff who may recontact respondents.

Amend unit properties. This function does not alter the value of observed variables but amends auxiliary properties relating to the observed unit. In business statistics, this function entails tasks like changing erroneous NACE codes and is often performed manually. Another commonly performed task falling into this category is the adjustment of estimation weights for representative outliers.

3 Numerical illustration

In this section, we illustrate the effects of applying a sequence of automatic editing functions by using part of a data set concerning Dutch child care institutions. It contains 40 variables on employment, costs and revenues similar to those in an SBS questionnaire. We have applied the automatic editing process steps listed in Table 2. Steps 1a-1d are generic and domain specific error correction methods. These steps combine a *field selection* method (detection of a specific type of error) with an appropriate amendment method. Step 2 is automatic error localisation under the Fellegi-Holt paradigm and step 3 and 4 together ensure that all missing values are filled in with values consistent with the edit rules. The numerical calculations have been performed using several recently developed R-packages (De Jonge and Van der Loo (2012), Van der Loo et al. (2011), Van der Loo (2012)).

The second column of this table shows the number of changed data values at each process step. In the third column are the numbers of failed edits at each process step, which can be obtained directly from the failed-edits matrix. Some edits cannot be evaluated for some records because the edit contains variables with missing values in that record. The corresponding elements of the failed-edits matrix are then missing and the number of such missing elements is in the column *Not evaluated edits*. The number of missing data values is in the last column.

The first line of Table 2 shows that before automatic editing there are, in the whole data set, 258 edit violations and 158 edits that cannot be evaluated because of 124 missing values. Steps 1a-1e apply correction of several systematic errors. The number of changed values indicates the number of detected errors of each type and consequently the number of amendments made. Both the number of violated edits and the number of systematic errors are quality indicators for the raw data set. However, some of the detections of systematic errors may be incorrect because the number of violated edits increases (step 1a and 1b). Changes that cause edit failures should be followed up manually not only to correct the data but also to review the correction rules and modify them so that they are consistent with the edit rules. The corrections 1d and 1e are very effective in removing errors as the number of edit failures is reduced substantially.

At this stage the possibilities for correction of generic and domain-specific systematic errors are exhausted. The remaining inconsistencies and missing values are resolved by applying steps 2 through 4. Error localisation identifies 215 values that need to

Table 2: Numbers of values changed, edit violations and missings at each step of a sequence of automatic editing functions

Process step	*Changed values*	*Violated edits*	*Not eval. edits*	*Missings*
0. None	0	258	158	124
1a. Rules for false minus signs	9	249	158	124
1b. Thousand errors	17	250	158	124
1c. Domain specific errors	43	252	158	124
1d. Simple typing errors	53	187	158	124
1e. Rounding errors	102	147	158	124
2. Error localisation	215	0	477	339
3. Model-based imputation	178	109	0	0
4. Adjustment of imputed values	144	0	0	0

be changed in order to be consistent with all edit rules. These values are treated as missing in the following process steps. The increase of missing values also increases the number of not evaluated edits to a great extent. These values are imputed (by regression imputation). These imputed values result again in edit violations. Therefore we adjust the imputed values as little as possible and solve the 109 edit violations and a complete and consistent data set results.

References

Camstra, A. and R. Renssen (2011). Standard process steps based on standard methods as part of the business architecture. In *Proceedings of the 58th World Statistical Congress (Session STS044)*, pp. 1–10. International Statistical Iinstitute.

De Jonge, E. and M. Van der Loo (2012). *editrules: R package for parsing and manipulating of edit rules and error localization.* R package version 2.5.

De Waal, T., J. Pannekoek, and S. Scholtus (2011). *Handbook of statistical data editing and imputation.* Wiley handbooks in survey methodology. John Wiley & Sons.

Fellegi, I. P. and D. Holt (1976). A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association 71*, 17–35.

Granquist, L. and J. G. Kovar (1997). Editing of survey data: how much is enough? In L. Lyberg, P. Biemer, M. Collins, E. de Leeuw, C. Dippo, N. Schwartz, and D. Trewin (Eds.), *Survey measurement and process quality*, Wiley series in probability and statistics, pp. 416–435. Wiley.

Latouche, M. and J.-M. Berthelot (1992). Use of a score function to prioritize and limit recontacts in editing business surveys. *Journal of Official Statistics 8*, 389–400.

Pannekoek, J., S. Scholtus, and M. van der Loo. Automated and manual data editing: a view on process design and methodology. *Journal of Official Statistics*. to appear.

Pannekoek, J. and L.-C. Zhang (2012). On the general flow of editing. Working Paper No. 10, UN/ECE Work Session on Statistical Data Editing, Oslo.

Scholtus, S. (2009). Automatic correction of simple typing errors in numerical data with balance edits. Technical Report 09046, Statistics Netherlands, Den Haag.

Scholtus, S. (2011). Algorithms for correcting sign errors and rounding errors in business survey data. *Journal of Official Statistics 27*, 467–490.

Van der Loo, M. (2012). *rspa: Adapt numerical records to fit (in)equality restrictions with the Successive Projection Algorithm.* R package version 0.1-1.

Van der Loo, M., E. De Jonge, and S. Scholtus (2011). *deducorrect: Deductive correction, deductive imputation, and deterministic correction.* R package version 1.3-1.