Working Paper 3.

**UNITED NATIONS**
**ECONOMIC COMMISSION FOR EUROPE**

**CONFERENCE OF EUROPEAN STATISTICIANS**

**Work Session on Statistical Data Editing**
(The Hague, Netherlands, 24-26 April 2017)

# Simplifying constraints in data editing

Prepared by Jacco Daalmans[1], Statistics Netherlands

## I. Introduction

1. Statistical offices have the task of publishing indisputable information about a diversity of subjects. Unfortunately, collected micro data usually contain errors, e.g. pregnant men, average salary of 5 million, components of a total that do not add up to that total. Correction of such errors is often necessary to prevent flaws and inconsistencies in statistics to be published. The process of checking and correcting data is called data editing, see e.g. De Waal (2008), De Waal *et al.* (2011) and Pannekoek *et al.* (2013). Data editing consists of two main steps: error detection and error correction. The focus of this paper will be entirely on detection of random errors (i.e. errors without a known cause) in numerical data.

2. Data editing is more and more automated, using computer programs without user intervention. The basis of the detection of random errors is edit rules: formal relationships between variables. A simple example is that total profits is equal to turnover minus costs. Usually, in business statistics, many edit rules are defined, that have many variables in common. In this way, connected systems of edit rules are obtained. Random error detection is often done by solving a mathematical optimization, in which edits are translated into constraints of an optimization problem, see e.g. Fellegi and Holt (1976) and De Waal *et al.* (2011).

3. The ongoing automation of data editing means that more and more subject matter knowledge is translated into edit rules. Specification of edit rules is often done by subject matter experts, who repeatedly add new edits to the set of existing rules. Thus, the number of edit rules for one survey can grow to a very large number. A large number of edits may cause problems. Firstly, computational problems may arise: computation time can be long or – in the worst case – automatic editing of a record can be impossible, see e.g. De Waal *et al.* (2011) . Secondly, when the number of edits is large, it may become difficult to understand the joint effect of the defined rules. Undesirable, non-intended implications can occur, for example one rule that contradicts another rule. Problems of this kind can be avoided by removing erroneous and redundant rules and simplifying rules as much as possible.

4. Erroneous edits are rules that have different implications than intended. For example, the following type of edit was found in an edit set, actually used by Statistics Netherlands:

If Questionaire_ID $\neq$ 1 OR Questionaire_ID $\neq$ 2 THEN VariableX = VariableY

[1] The author is grateful to Sander Scholtus for valuable comments on a previous version of this text.

Here, the OR-operator was meant to be an AND-operator. One can easily show that the conditional part of this edit is always satisfied.

5. Redundant edits are rules that are declaratively implied by other edits. Such edits can be removed from an edit set, without affecting results. The easiest example is a duplication of rules.

6. It can be very difficult to identify redundant, erroneous and unnecessarily complicated rules by hand, especially if the number of edit rules is large. Fortunately, mathematical methods are available for this purpose. An advantage of using formal, mathematical methods is that users do not have to bother about redundancy and the complexity of edit rules; they can specify as many rules as desired, simplification and error detection are done automatically out of sight.

7. Edits can be more formally specified as constraints on values. Therefore, the problem of edit rule simplification can be more generally stated as a problem of constraint simplification. A large number of methods for constraint simplification are available from the literature on Mathematical Optimization and Artificial Intelligence, see for instance Paulraj and Sumathi (2010), Chmeiss *et al.* (2008) and Piette (2008).

8. It is remarkable that algorithms for constraint simplification are not often mentioned in the context of data editing. Despite that Statistics Canada developed the SAS application BANFF for economic surveys, with certain methods for constraint simplification (see BANFF, 2005, Chapter 2), their methods do not allow for conditional (IF-THEN) edits that are frequently used in Official Statistics.

9. This paper contributes to fill the gap of constraint simplification techniques for data editing purposes. A number of methods will be proposed for data editing, based on known algorithms from Operations Research and Artificial Intelligence. Further, it will be demonstrated that these algorithms can actually be used to simplify real-life edit sets.

## II. Edits

10. The purpose of this section is to explain the format of edit rules that are considered in the remainder of this paper. A distinction will be made between conditional and unconditional edits.

11. Unconditional edits.
Examples of unconditional edits are:

Number of employees $\geq 0$;
Total turnover = Domestic turnover + Foreign turnover.

Mathematically, these are rules that can be written as a system of linear equalities and / or inequalities. A general form is given by:

$\boldsymbol{A_E x = b_E}$,
$\boldsymbol{A_I x \leq b_I}$,

where $\boldsymbol{A}$ is a matrix of coefficients and $\boldsymbol{b}$ a vector of constants. The subscript $E$ is used for equality constraints and $I$ for inequality constraints.

12. Conditional edits
We will consider ‘simple’ conditional edits in the form

IF <statement 1 > THEN <statement 2>

(where each statement is a single linear equality or inequality) and more complicated 'compound' rules that may contain

a) AND-statements in the IF-clause and/or;
b) OR-statements in the THEN-clause.

An example of a compound edit is:

IF number of employees > 0 AND turnover > 0 THEN
wages > 0 OR labour costs > 0.

As explained in Daalmans (2015), all 'simple' and 'compound' conditional edits can be converted into Disjunctive Normal Form (DNF). This form is defined by

$C_1$ OR $C_2$ OR $C_3$ OR…

where $C_i$ denote single inequality and equality constraints.
For example, our previous example can be written in DNF as follows:

number of employees ≤ 0 OR turnover ≤ 0 OR wages > 0 OR labour costs > 0.

In the remainder of this paper it will be assumed that unconditional edits are expressed as a system of linear (in)equalities and that conditional edits are written in disjunctive normal form.

## III. Simplification of conditional edits

13. Our aim is to simplify conditional rules by reducing as much as possible the number of components. If all components can be removed but one, a conditional rule is converted into an unconditional rule. Replacing conditional by unconditional edits is very beneficial, because processing of unconditional edits usually requires less computational effort.

14. Two methods will be considered for the simplification of conditional edits: 1) Removal of non-relaxing components and 2) Removal of non-constraining components. Below we will first illustrate the idea of these methods and thereafter we present pseudo-code for their implementation.

### A. Non-relaxing components

15. As mentioned in Dillig *et al.* (2010), non-relaxing components are statements within a conditional edit that cannot occur, given the other edits from an available set of edits. Non-relaxing components can be removed from the DNF representation of a conditional edit. An example is given below

Edit 1: $x_1 > 0$ OR $x_2 > 0$ OR $x_3 > 0$,
Edit 2: $x_2 < 0$.

Edit 2 implies that the statement $x_2 > 0$, within Edit 1, cannot possibly be satisfied by a feasible record: it is 'always FALSE'. Therefore it can be deleted from the edit.
We obtain:

Edit 1': $x_1 > 0$ OR $x_3 > 0$.

The number of terms in a compound edit statement diminishes by this kind of simplification. As mentioned above, a conditional edit is replaced by an unconditional one if only one component remains.

16. A more formal definition is given below:

*Definition*: A component $e_{ij}$ of a compound edit $e_i$ within an edit set $E$ is non-relaxing if $E \cup e_{ij}$ is infeasible.

17. Here, $E \cup e_{ij}$ stands for the edit set that is obtained by extracting a compound edit's component $e_{ij}$ from $e_i$ and adding it to the set $E$, as if it were a single edit.

18. The following algorithm can be applied for identifying non-relaxing components:

**Algorithm 1**: Identification & removal of non-relaxing components

**Input:** Edit set $E$
**Output:** Edit set $E$, without non-relaxing components.
**1** For each compound edit $e_i \in E$ do
**2** For each component $e_{ij} \in e_i$ do
**3** $E^* \leftarrow E \cup e_{ij}$;
**4** IF isFeasible($E^*$) = FALSE THEN $e_i \leftarrow e_i \setminus e_{ij}$
**5** Next
**6** Next

In each step of the algorithm one edit is added to an edit set $E$. Subsequently, the feasibility of the adapted edit set is checked. A similar structure will be applied in the Algorithms 2 and 3 below.

19. The function isFeasible() checks the feasibility of the constraints implied by edit set. As explained in Daalmans (2015), this function can be implemented by means of a MIP-solver, where MIP stands for Mixed Integer Programming, a well-known technique in Operations Research.

## B Non-constraining components

20. Non-constraining components of a compound edit are components that are always satisfied by a feasible record, given the edits of an edit set. Compound edits with a non-constraining component can be replaced by a single edit. Consider the following example:

Edit 1: $x_1 < 50$ OR $x_2 > 100$,
Edit 2: $x_1 > 100$ OR $x_2 > 0$.

Here, it follows that $x_2 > 0$ is a non-constraining component. For all possible $x_1$ values, at least one of the statements $x_1 < 50$ and $x_1 > 100$ is not satisfied. Thus, it follows from Edits 1 and 2 that either $x_2 > 0$, or the even stronger condition $x_2 > 100$, needs to be true. As a result, we obtain that $x_2 > 0$ always needs to hold.

21 Because non-constraining components are always fulfilled, non-constraining components of a compound edit can be added to an edit set as if they were a single, unconditional edit. In our example, we can add "$x_2 > 0$" to the edit set.

22. After doing this, all compound edits with a non-constraining component can be removed, as it follows that the added unconditional rules imply that these compound edits are always satisfied. In our previous example, Edit 2 can be omitted, after inserting "$x_2 > 0$" to the edit set.
The resulting edit set of our example is given by

Edit 1: $x_1 < 50$ OR $x_2 > 100$,
Edit 2': $x_2 > 0$.

23. A more formal definition is stated below:

*Definition*: A component $e_{ij}$ of a compound edit $e_i$ within an edit set *E* is non-constraining if $E \cup \neg\, e_{ij}$ is infeasible. (where ¬ stands for negation)

24. This definition makes use of the equivalence between the statement that a compound edit's component is always satisfied and the statement that the opposite of that component cannot occur.

25. The following algorithm can be applied for identifying non-constraining components:

**Algorithm 2**: Identification of non-constraining components

**Input:** Edit set *E*
**Output:** Edit set *E*, without non-constraining components.
**1** FOR each compound edit $e_i \in E$ DO
**2** FOR each component $e_{ij} \in e_i$ DO
**3** $E^* \leftarrow E \cup \neg\, e_{ij}$ ;
**4** IF isFeasible($E^*$) = FALSE THEN $E \leftarrow \{E \backslash e_i\} \cup e_{ij}$ END IF
**5** NEXT
**6** NEXT

## IV. Redundant edit removal

26. Redundant edits are non-constraining edits. These edits can be left out of an edit set, without affecting results. Consider the following example:

Edit 1: $x_1 < 10$ ,
Edit 2: $x_1 < 12$.

Since Edit 2 is implied by Edit 1, Edit 2 does not reduce the 'feasible set' and it can be removed accordingly. Another simple example of a redundant edit is the edit "5 > 4".
Because redundant edits may emerge as a result of simplification of conditional edits, it is important that redundant edit removal is performed after simplification of conditional edits.

27. An edit is redundant if all other edits imply that the edit is 'always satisfied'. This is equivalent to the statement that the opposite of the edit (in mathematical terms: negation) cannot occur. This leads to the following definition, that was also described before in Chmeiss *et al.* (2008) and Bruni (2005):

*Definition*: An edit $e_i$ from an edit set $E$ is redundant, if $\{E \backslash e_i\} \cup \neg e_i$ is infeasible.

The edit set $\{E \backslash e_i\} \cup \neg e_i$ is obtained from $E$, by replacing Edit $e_i$ by its negation.

28. The following pseudo-code can be used to remove redundant edits

**Algorithm 3:** Identification & removal of redundant edits
**Input:** Edit set $E$
**Output:** Edit set $E$, without redundant edits
**1** FOR each Edit $e_i \in E$ DO
**2** $E^* \leftarrow \{E \backslash e_i\} \cup \neg e_i$ ;
**3** IF isFeasible($E^*$) = FALSE THEN $E \leftarrow E \backslash e_i$
**4** NEXT

# V. Applications

29. The aim of this section is to apply constraints simplification methods on 'real-life' and 'fictitious' edit sets. By doing this we would like to show that the described methods are practically useful. The following data sets were used:

(a) Wholesale: Real-life edit set used for the 2007 Dutch Production Statistics for wholesale in agricultural products and livestock, for businesses with 10 employed persons or more;
(b) Health-care: Real-life edit set used for a Dutch survey among welfare and childcare institutions;
(c) Finance of Enterprises: Edit set under development, meant to be used for a Dutch survey on financing of enterprises;
(d) Bruni & Bianchi: Small, fictitious edit set derived from the Bruni and Bianchi (2012) paper, Section 4.

30. Table 1 summarizes the main properties and results for these data sets

**Table 1. Application to four edit sets**

| | Wholesale | Health care | Finance of Enterprises | Bruni & Bianchi |
|---|---|---|---|---|
| **Original edits** | | | | |
| Number of edits | 118 | 196 | 339 | 10 |
| of which conditional: | 12 | 114 | 5 | 1 |
| Number of variables in edits | 89 | 75 | 343 | 3 |
| **Simplification** | | | | |
| Redundant edits | 18 | 10 | 35 | 4 |
| Conditional replaced by unconditional | 1 | 7 | 0 | 0 |
| **Cleaned edits** | | | | |
| Number of edits | 100 | 186 | 304 | 6 |
| of which conditional: | 11 | 104 | 5 | 1 |

31. Several examples were found in which a conditional edit can be replaced by an unconditional edit: one case for 'wholesale' and seven cases for 'health care'.
In most cases, this can be easily understood. An example, based on the health care survey:

$A > 0$,
IF $A > 0$ THEN $B > 0$.

where $A$ and $B$ stand for a variable with a complicated name, not worthwhile to mention. It is quite obvious that the second edit can be replaced by "$B > 0$". The second edit is an unnecessarily complicated way of stating that $B$ has to be larger than zero.

32. In other cases, it is more difficult to understand why a conditional edit can be replaced by an unconditional edit. Consider the following example, taken again from health care:

IF $A > B$ THEN $A \leq 0$,
$B \geq 0$.

The first unconditional edit can be replaced by $A \leq B$. To explain this, suppose that $A > B$. Then, according to the first edit it follows: $A \leq 0$. But, it also follows that $B < 0$, because of the assumption $A > B$. This, however, contradicts the second edit.

33. For 'Wholesale' one difficult-to-understand case is found of a conditional edit that can be replaced by an unconditional edit. This follows from the joint effect of no less than nine edits, see Daalmans (2015). This illustrates the important merit of automated procedures that unnecessarily complicated constraints can be identified that would be very hard to detect by hand.

34. Redundant edits were detected in all data sets. For "Health care" redundant conditional edits were found, that are implied from other conditional edits. An illustrative example is:

Edit 1: IF $A \geq 10B$ THEN $C \leq 0$,
Edit 2: IF $C < 1$ THEN $B \leq 0$,
Edit 3: IF $A \geq 10B$ THEN $B \leq 0$.

Here, Edit 3 is implied by Edit 1 and 2. It can be deleted, accordingly.

35. A frequently occurring case is redundancy of lower and upper bounds. A typical example is when one variable is a sum of other variables and each variable, including the sum, has to be at least zero. In that case, the lower bound for the sum is redundant.

36. Nevertheless, users may find it informative to specify bounds for all variables. A great advantage of automated procedures is that removal of redundant constraints can be done out of sight, so that users can still specify all possible bounds, without ending up with an inefficient edit set.

## VI. Discussion

37. From software verification to soccer league planning; a great variety of applications of automated constraint handling problems is described in literature. An enormous number of constraints may be handled in those applications. Many authors have indicated that performance of these applications can be improved by conducting a constraint simplification step. However, to the best knowledge of the author, constraint simplification is not often applied in the field of data editing.

38. This paper shows that automated data editing can benefit from available methods for constraint simplification. Three algorithms were proposed that are based on techniques from Operations Research and Artificial Intelligence. It was shown that real-life edit sets can actually be simplified by adopting these algorithms.

39. To keep this paper short, we restricted ourselves to three algorithms for simplification of conditional rules and the removal of redundant rules. However, algorithms for other purposes are also available in the literature. For example, Daalmans (2015) describes an algorithm for the detection of fixed values, i.e. variables that only have one admissible value, given the edits in an edit set. An edit set can be simplified by substituting fixed values in all edit rules. In this way, the number of variables in the edit rules reduces.

40. In this paper we implicitly assumed that edit sets are feasible, i.e. that there are no contradictory constraints. In reality, however, infeasible edit sets may occur, for instance as a result of wrongly formulated rules. It can be hard to find the cause of a contradiction, especially if the number of edit rules is large. For small edit sets, it is much easier to find out the reason for inconsistency. Therefore, ample literature is available on isolating a smallest possible subset of inconsistent constraints: a so-called Irreducible Inconsistent Subset (IIS), see e.g. Daalmans (2015) and Bruni and Bianchi (2012).

41. The focus of this paper was entirely on random errors without a known cause. The simplification problem that is discussed here, is also relevant for the correction of systematic errors. The basis of the editing of systematic edits are correction rules. A correction rule means that a 'correction variable' is adjusted according to a pre-specified correction rule for all records that satisfy a certain condition. An example of a correction rule is

IF Profits ≠ Turnover – Costs THEN Profits := Turnover – Costs,

profits are set equal to the difference of turnover and costs for all records that do not fulfil this relation. Correction rules give rise to corrections that are conducted in a particular sequence. It can be easily verified that the order of execution matters for the results.
Another complication is that, after all rules have been conducted once, the repeated application of a set of correction rules, may lead to new corrections.
Statistics Netherlands has conducted research to avoid the latter problem. That is: to find a rule for the order of execution that guarantees that no new corrections are made if a set of rules are repeatedly applied. The results of this study are available (in Dutch) in Daalmans (2016) and Scholtus (2016).

## VII. References


Banff (2005), *Functional Description of the BANFF System for Edit and Imputation*, Statistics Canada, internal document.

Bruni (2005), Error correction for massive data sets. *Optimization Methods and Software*, 291-310.

Bruni R. & G. Bianchi (2012). A Formal Procedure for Finding Contradictions into a Set of Rules. *Applied Mathematical Sciences*, 6, 6253 - 6271.

Chmeiss, A., V. Krawczyk & L. Sais (2008). Redundancy in CSPs. In: Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008), IOS Press, Amsterdam.

Daalmans J. (2015), Simplifying constraints in data editing, Discussion Paper 201518, Statistics Netherlands. https://www.cbs.nl/-/media/imported/documents/2015/48/2015-simplifying-constraints-in-data-editing.pdf

Daalmans J. (2016), *Van coecrrteireegls naar correctieregels* (Dutch), Internal report, Statistics Netherlands.

De Waal T. de (2008). *An overview of statistical data editing*, Discussion Paper 08018, Statistics Netherlands. http://www.cbs.nl/NR/rdonlyres/693E4B18-9322-4AC2-99FD-DB61F03637B2/0/200818x10pub.pdf

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

Dillig I., T. Dillig and A. Aiken (2010). Small formulas for large programs: On-line constraint simplification in scalable static analysis. In: Cousot R. and M. Martel, (eds), *SAS 2010,* LNCS 6337, 236–252. Springer. http://theory.stanford.edu/~aiken/publications/papers/sas10.pdf

Fellegi, I. P. and D.Holt (1976), A Systematic Approach to Automatic Edit and Imputation, *Journal of the American Statistical Association*, 71, 17-35.

Pannekoek, J., S. Scholtus and M. van der Loo (2013), Automated and manual data editing: a view on process design and methodology, *Journal of Official Statistics*, 29, 4, 511–537.

Paulraj S. and P. Sumathi (2010). A Comparative Study of Redundant Constraints Identification Methods in Linear Programming Problems. *Mathematical Problems in Engineering*, 2010, Article ID 723402.

Piette,C. (2008). Let the solver deal with redundancy. In: *20th IEEE International Conference on Tools with Artificial Intelligence* (ICTAI'08), Dayton, Ohio, 67–73.

Scholtus S. (2016), *Over het volgordeprobleem bij correctieregels* (Dutch), Internal report, Statistics Netherlands