Automatic editing: a generalised Fellegi-Holt paradigm

Jacco Daalmans and Sander Scholtus
Contents

• Generalised paradigm for automatic data editing

• Towards practical implementation

• Simulation study

• Technical details
Automatic data editing

Step 1. Systematic errors

Step 2. Random errors: error localisation and imputation
Automatic data editing

Step 1. Systematic errors

Deterministic correction rules
‘If \(x < 0\) and \((-x + y = \text{Total})\) Then Replace \(x\) by \(-x\).’

Step 2. Random errors: error localisation and imputation

Error localisation based on edit rules:
‘profit = revenues minus costs’,
‘pregnant women cannot be younger than 10 etcetera.’

Fellegi-Holt paradigm:
Correct a minimum number of values to satisfy all edit rules
Problem

Not all systematic errors can be (easily) automatically corrected.

Data in step 2 (‘correction of random errors’) might still contain systematic errors.
Not all systematic errors can be (easily) automatically corrected

Data in step 2 (‘correction of random errors’) might still contain systematic errors

Fellegi-Holt based algorithms do not produce adequate solutions for systematic errors;

The generalised FH-algorithm by Scholtus aims to solve this problem
Generalised paradigm

Fellegi-Holt’s paradigm:

“find the minimum number of corrections, to make a record consistent with all edit rules”.
Generalised paradigm

**Generalised** Fellegi-Holt’s paradigm

*edit operations*

“find the minimum number of corrections, to make a record consistent with all edit rules”.  (Sander Scholtus)
Generalised paradigm

**Generalised** Fellegi-Holt’s paradigm

“find the minimum number of corrections, to make a record consistent with all edit rules”.  
(Sander Scholtus)

Edit operations:
- ‘Fellegi-Holt’: designate one value to be erroneous.
- **Special operation**: Change sign  -100 -> 100
- **Special operation**: Interchange values: (1, 2) -> (2,1)
- etcetera
Generalised paradigm

- can deal with systematic and random errors

- Locates erroneous values and might (or might not) uniquely determine an imputed value

- Scholtus proposed a branch-and-bound algorithm, that cannot be applied to realistic problems (too resource intensive).
Aim

Propose a practical feasible algorithm to implement the generalised paradigm

Based on Mixed Integer Programming (MIP) and simplifications of the original idea.
Simplifying assumptions

- Each variable involved with at most one special operation

- Interpretation: ‘Special operations’ before ‘Fellegi-Holt operations’
Simplifying assumptions

- At most one special operation is applied to each variable ...avoids order-dependency
  e.g.
  operation 1: interchange \((x, y)\)
  operation 2: interchange \((y, z)\)
  Order matters!

- Interpretation: ‘Special operations’ before Fellegi-Holt operations
Mixed Integer programming (MIP)

Minimizes an objective function, subject to constraints

**Objective function:** (weighted) number of corrections (including special operations)

**Constraints:** A corrected record exists that satisfies all rules

Well studied problem; efficient ‘solvers available’.

All admissible special operations need to be determined beforehand.
Simulation: setup

- Structural Business Survey (SBS) Transport:
  1080 records; manually edited (error free)
  140 variables

- Errors were added to these data (max 5)
  - Random errors to all variables
  - 15 special operations
  Special operations had a 3 times higher probability than random errors
  - This is reflected in the objective function (weights)
Simulation: computation time

Generalised approach feasible for realistic data

Computation time ca 1.5 times higher than for standard approach

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Computation time per record</th>
</tr>
</thead>
<tbody>
<tr>
<td>FH original</td>
<td>mean (s)</td>
</tr>
<tr>
<td></td>
<td>1.71</td>
</tr>
<tr>
<td>FH extended</td>
<td>3.02</td>
</tr>
</tbody>
</table>
Simulation: evaluation

Fraction of ‘true errors’ that is indeed detected as erroneous
True Positives / (True Positives + False Negatives)

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Records correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>FH original</td>
<td>59.7%</td>
<td>41.7%</td>
</tr>
<tr>
<td>FH extended</td>
<td>73.6%</td>
<td>56.8%</td>
</tr>
</tbody>
</table>
Simulation: evaluation

Fraction of ‘true errors’ that is indeed detected as erroneous
True Positives / (True Positives + False Negatives)

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Records correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>FH original</td>
<td>59.7%</td>
<td>41.7%</td>
</tr>
<tr>
<td>FH extended</td>
<td>73.6%</td>
<td>56.8%</td>
</tr>
</tbody>
</table>

In a second scenario, max. 10 errors not 5, the outcomes are:

<table>
<thead>
<tr>
<th></th>
<th>Sensitivity</th>
<th>Records correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>FH original</td>
<td>47.3%</td>
<td>28.6%</td>
</tr>
<tr>
<td>FH extended</td>
<td>50.7%</td>
<td>40.1%</td>
</tr>
</tbody>
</table>
Conclusions

Extended FH method detects ‘system errors’ that are not found by a standard FH-method.

The extended FH method can be efficiently implemented by a MIP-approach.

Future activities: More practical experiments to see whether the extended method can be used in production?