Error localization as a mixed integer problem with the `editrules` package

package version 2.2.0

Edwin de Jonge and Mark van der Loo

February 22, 2012

Abstract

This vignette is far from finished. Version 2.0 of the package will have the full vignette. At the moment, functionality for solving error localization problems with lp solvers is experimental.

Contents

1 Introduction

Analyses of real data are often hindered by occurrences of incomplete or inconsistent raw data records. The process of locating and correcting such errors is referred to as *data editing*, and it has been estimated that National Statistics Institutes spend up to 40% of their resources on this process (De Waal et al., 2011). For this reason, considerable attention is paid to the development of data editing methods that can be automated. Since data are often required to obey many interrelated consistency rules, data editing can be too complex to perform manually. Winkler (1999) mentions practical cases where records have to obey 250, 300 or even 750 internal consistency rules. Although the R statistical environment has numerous facilities for analyzing categorical data [See *e.g.* Husson et al. (2010)], the options for error localization and record correction are currently limited.

The R package editrules was developed to help closing the gap between raw data retrieval and data analysis with R. The main purpose of the editrules package is to provide a user-friendly environment for handling data restriction rules, to apply those rules to data, and to localize erroneous fields in data based on the generalized principle of Fellegi and Holt (1976). The package does not offer functionality for data correction. However, it does facilitate the identification of the set of solutions for an error correction problem. For a detailed description we refer to De Jonge and Van der Loo (2011) and Van der Loo and De Jonge (2011).

editrules offers a fairly complete toolbox to work with numerical and categorical edits. It contains a flexible object for localizing errors, which can be adapted by the user of editrules based on a branch and bound algorithm (De Waal et al., 2011). However, for the time to solve the error localization problem with a branch and bound algorithm grows exponentially with the number of (incorrect) variables. Many surveys have hundreds of variables, which results in (very) long processing times for records with many errors.

This paper describes the error localization problem for numerical and categorical edits as a mixed integer problem and its implementation in editrules. A brief description can be found in (De Waal et al., 2011) p. 75. Mixed integer programming is a special case of linear programming. Linear programming maximizes a linear objective function, which is subject to linear equality and linear inequality constraints. More formally:

$$\text{Maximize } \mathbf{c}^T\mathbf{x} \tag{1}$$

$$A\mathbf{x} \leq \mathbf{b} \tag{2}$$

$$\text{with } \mathbf{x} \geq 0 \tag{3}$$

where $\mathbf{x}$ is the vector of (numerical) variables to be optimized, $\mathbf{c}$ is a vector of weights, $\mathbf{b}$ is a vector of upper bounds and A is a coefficient matrix for the constraints. In mixed integer programming (mip) $\mathbf{x}$ is a mixture of continious and integer variables.

Error localization implemented as a mixed integer programming results in a fast procedure, which is typically much faster than the branch and bound method also available in editrules.

In section 2 we describe a mip formulation for numerical records. Section ref describes the mip formulation for categorical records. We end with a discussion on the implementation in editrules.

2 Numerical data errors

A numerical record $\mathbf{x}$ with reported values $(x_1^0, \ldots, x_m^0)$ has the following linear constraints:

$$A\mathbf{x} \odot \mathbf{b} \text{ with } \odot \in \{<, \leq, =\}^n, \tag{4}$$

It can easily be checked with editrules if $\mathbf{x}$ violates any of these constraints. If this is the case, the task is to find the minimal (weighted) number of adjustments to the reported values such that it complies to the edits. These edits are (usually) valid for all records of a data set. For each reported record we define a seperate mip, which contains the set of edits extended with edits that depend on the reported values.

In practice reported values are always bounded: No person is older than 200 years old or taller than 30 feet. So each for each variable x_i we assume a lower boundary $\mathrm{l_i}$ and an upper boundary $\mathrm{u_i}$[1].

$$\mathrm{l}_i \leq x_i \leq \mathrm{u}_i \tag{5}$$

For each x_i we introduce a binary variable $\Delta_i \in \{0, 1\}$:

$$\Delta_i = \begin{cases} 0 & if x_i = x_i^0 \\ 1 & if x_i \neq x_i^0. \end{cases} \tag{6}$$

where x_i^0 is the reported value for x_i. We now add the following edits to the edit set:

$$(\mathrm{l}_i - x_i^0)\Delta_i + x_i^0 \leq x_i \leq x_i^0 + (\mathrm{u}_i - x_i^0)\Delta_i \tag{7}$$

It can easily be checked that if $\Delta_i = 0$ these edits reduce to $x_i = x_i^0$, meaning that the reported x_i^0 is assumed correct. If $\Delta_i = 1$ the edits reduce to equation 5, meaning that the value of x_i can take any feasible value within its boundaries $\mathrm{l_i}$ and $\mathrm{u_i}$.

Using principle of Felligi Holt (Fellegi and Holt, 1976) which minimizes the weigthed sum of adaptations, the error localization problem can be written as:

[1]Note that equation 5 can also be written in the form of 4. We will use univariate or boundary constraints explicitly and exclude them from 4.

Minimize $\sum_{j=1}^{m} \mathrm{w_j}\Delta_j$, with:

$$
\begin{array}{rcrcl}
\sum_{j=1}^{m} a_{1j}x_j & & & \odot_1 & b_1 \\
\dots & & & \dots & \dots \\
\sum_{j=1}^{m} a_{nj}x_j & & & \odot_n & b_n \\
x_1 & - & (\mathrm{u_1} - \mathrm{x_1^0})\Delta_1 & \leq & x_1^0 \\
-x_1 & + & (\mathrm{l_1} - x_1^0)\Delta_1 & \leq & -x_1^0 \\
\dots & & \dots & & \dots \\
x_m & - & (\mathrm{u_m} - x_m^0)\Delta_m & \leq & x_m^0 \\
-x_m & + & (\mathrm{l_m} - x_m^0)\Delta_m & \leq & -x_m^0
\end{array}
\tag{8}
$$

with $\mathrm{w_j}$ a weight for variable x_j. This problem is equal to the following mixed integer problem:

$$
\begin{array}{rcl}
\text{Maximize } \mathbf{c}^T\hat{\mathbf{x}} & & \\
\hat{A}\hat{\mathbf{x}} & \leq & \hat{\mathbf{b}} \\
\text{with } \hat{\mathbf{x}} & \geq & 0 \\
\hat{\mathbf{x}} & = & (x_1 - \mathrm{l_1}, \dots, x_m - \mathrm{l_m}, \Delta_1, \dots, \Delta_m) \\
\mathbf{c} & = & (0, \dots, 0, -\mathrm{w_1}, \dots, -\mathrm{w_m}) \\
\hat{\mathbf{b}} & = & (b_1, \dots, b_m, x_1^0, -x_1^0, \dots, x_m^0, -x_m^0)
\end{array}
$$

2.1 Boundaries

The extended edit set derived in (8) depends on the boundaries $\mathrm{l_i}$ and $\mathrm{u_i}$. Mathematically the size of these coefficients does not matter. Computationally however their size is important. If any $|\mathrm{u_i} - \mathrm{c_i}| \gg |a_j|$ or $|\mathrm{l_i} - \mathrm{c_i}| \gg |a_j|$ the resulting mixed integer program may become numerical unstable. From the lpsolve manual (lps, 2012):

> The chance for numerical instability and rounding errors is considerably larger when the input data contains both large and small numbers. So to improve stability, one must try to work with numbers that are somewhat in the same range. Ideally in the neighbourhood of 1.
>
> You should realize, that you the user are probably in a better position to scale the problem than any computer algorithm.

For the error localization problem this means that the upper and lower boundaries of variables x_i shouldn't be too large. A reasonable approach is to take the minimum and maximum values for all variables in a dataset and multiply them with a factor f (e.g. 1000).

```
> E <- editmatrix(c(
+     "x + y == z",
+     "x > 0",
+     "y > 0",
+     "z > 0"))
> dat <- data.frame(
+     x = c(1,-1,1),
+     y = c(-1,1,1),
+     z = c(2,0,2))
> # localize all errors in the data using mip
> localizeErrors(E,dat, method="mip")

Object of class 'errorLocation' generated at Wed Feb 22 09:48:16 2012
call : localizeErrors(E, dat, method = "mip")
method : mip
slots: $adapt $status $call $method $user $timestamp

Values to adapt:
      adapt
record     x     y     z
     1 FALSE  TRUE FALSE
     2  TRUE FALSE  TRUE
     3 FALSE FALSE FALSE

 Status:
  weight degeneracy  user system elapsed maxDurationExceeded
1      1         NA 0.004      0   0.004               FALSE
2      2         NA 0.004      0   0.003               FALSE
3      0         NA 0.000      0   0.002               FALSE
```

Figure 1: Localizing errors in a data.frame using `method="mip"`

2.2 Implementation in editrules

editrules contains the function localizeErrors which can be used to localize errors in a `data.frame`. It default implemention using a branch and bound algorithm is described in detail in (De Jonge and Van der Loo, 2011). It accepts an editmatrix and a data.frame, and returns an object of class errorLocation. An errorLocation object contains the locations of errors for each record in the data.frame as well as logging information, solution weights and degeneracy.

We extended this function with a parameter `method`, that can be used to select the `localizer` or `mip` method. If the method "mip" is given, editrules internally creates for each record an extended editmatrix `E` that contains the conditional boundary conditions as defined in (8). Internally the package lpsolve and Konis (2011) is used to solve the resulting mixed integer program.

Figure 1 shows an example of localizing errors using mip.

The default boundary conditions used in `localizeErrors` use $\mathrm{u_i} = 1000 \cdot x_i^0$ and $\mathrm{l_i} = -1000 \cdot x_i^0$.

3 Categorical data errors

A categorical record $\mathbf{v}$ with reported values $(v_1^0, \ldots, v_m^0)$ has the following n constraints:

$$e^i = \begin{array}{ll} \textbf{if} & v_j \in F_j^i \text{ for } j = 1, \ldots, m \\ \textbf{then} & \text{FALSE} \end{array} \tag{9}$$

with $F_j^i \subseteq D_j$ where D_j is the possible set of categories $\{1, \ldots, k\}$ for each v_j and $i = 1 \ldots n$ the number of edits.

It can easily be checked with editrules if $\mathbf{v}$ violates any of these constraints. If this is the case, the task is to find the minimal (weighted) number of adjustments to the reported values such that it complies to the edits. Internally editrules uses an editarray object which is described in detail in Van der Loo and De Jonge (2011). In this paper we use a different, but equivalent, representation to allow for an easy conversion to a mixed integer program: cateditmatrix.

A record of m categorical variables can be written as an element of the cartesian product space D:

$$D = D_1 \times D_2 \times \cdots \times D_m, \tag{10}$$

where each D_j is the set of possible categories for variable j.

The number of categories for variable j is labeled d_j while the total number of categories is labeled d, given by

$$d = \sum_{j=1}^{n} d_j \tag{11}$$

Categorical record $\mathbf{v}$ can be written as in this space as

$$\begin{array}{rcl} \check{\mathbf{v}} & = & \oplus_{j=1}^{m}(\check{v}_{j1}, \ldots, \check{v}_{jd_j}) \\ & = & (\check{v}_{11}, \cdots, \check{v}_{1d_1}, \cdots, \check{v}_{m1}, \cdots, \check{v}_{md_m}) \end{array} \tag{12}$$

$$\text{with } \check{v}_{jk} = \begin{cases} 0 & \text{if } v_j \neq k \\ 1 & \text{if } v_j = k \end{cases} \tag{13}$$

Equation (9) can be written in the following form:

$$e^i = \neg \bigwedge_{j=1}^{m} v_j \in F_j^i = \bigvee_{j=1}^{m} \neg(v_j \in F_j^i) \tag{14}$$

$$= \bigvee_{j=1}^{m} v_j \notin F_j^i = \bigvee_{j=1}^{m} v_j \in T_j^i \tag{15}$$

with $T_j^i = D_j \setminus F_j^i$. This means that edit e^i is satisfied if at least one $v_j \in T_j^i$. e^i can therefore be written as:

$$\sum_{j=1}^{m} \left(\sum_{k=1}^{d_j} t_{jk}^i \cdot \check{v}_{jk} \right) \geq 1 \tag{16}$$

$$\text{with } t_{jk}^i = \begin{cases} 0 & \text{if } k \notin T_j^i \\ 1 & \text{if } k \in T_j^i \end{cases} \tag{17}$$

These edits are (usually) valid for all records of a data set. Note that equation (21) is in a form to be used in a mixed integer program where $\check{v}_{jk}$ are integer variables.

For each v_j its categories should exclude each other, so the following constraint must hold:

$$\left(\sum_{k=1}^{d_j} \check{v}_{jk} \right) \leq 1 \tag{18}$$

We introduce for each v_j a binary variable $\Delta_j \in 0, 1$:

$$\Delta_j = \begin{cases} 0 & \text{if } v_j = v_j^0 \\ 1 & \text{if } v_j \neq v_j^0. \end{cases} \tag{19}$$

where v_j^0 is the reported value for v_i. We now add the following edit to the edit set:

$$\check{v}_{jk_j^0} = 1 - \Delta_j \text{ with } k_j^0 = v_j^0 \tag{20}$$

It can be easily checked that if $\Delta_j = 0$ $\check{v}_{jk_j^0} = 1$, meaning that the reported value v_j^0 is assumed correct. If $\Delta_j = 1$, $\check{v}_{jk_j^0} = 0$, meaning that v_j should have a different value.

The categorical error localization problem now can be written as:

Minimize $\sum_{j=1}^{m} \mathrm{w_j} \Delta_j$, with:

$$\begin{array}{rcrcl} \sum_{j=1}^{m} \left(\sum_{k=1}^{d_j} t_{jk}^1 \cdot \check{v}_{jk} \right) & & & \geq & 1 \\ \cdots & & & \cdots & \cdots \\ \sum_{j=1}^{m} \left(\sum_{k=1}^{d_j} t_{jk}^n \cdot \check{v}_{jk} \right) & & & \geq & 1 \\ \sum_{k=1}^{d_1} \check{v}_{1k} & & & \leq & 1 \\ \check{v}_{1k_1^0} & + & \Delta_1 & = & 1 \\ \cdots & & \cdots & & \cdots \\ \sum_{k=1}^{d_m} \check{v}_{mk} & & & \leq & 1 \\ \check{v}_{mk_m^0} & + & \Delta_m & = & 1 \end{array} \tag{21}$$

```
> E <- cateditmatrix(
+        expression( gender %in% c("male", "female")
+                  , if (pregnant) gender == "female"
+                  )
+                   )
> as.data.frame(E)

  name                              edit
1 num1 gender:female + gender:male == 1
2 num2        pregnant <= gender:female

> getAb(E)

      var
rules  gender:female gender:male pregnant CONSTANT
  num1             1           1        0        1
  num2            -1           0        1        0
```

Figure 2: Using the cateditmatrix object

Note that parts of equation (16) can be written in a negated form:

$$\text{if } T_j^i \neq \emptyset \text{ then } \sum_{k=1}^{d_j} t_{jk}^i \cdot \check{v}_{jk} = 1 - \left(\sum_{k=1}^{d_j} f_{jk}^i \cdot \check{v}_{jk} \right) \tag{22}$$

$$\text{with } f_{jk}^i = \begin{cases} 0 & \text{if } k \notin F_j^i \\ 1 & \text{if } k \in F_j^i \end{cases} \tag{23}$$

3.1 The cateditmatrix object

To create a mip formulation for the categorical error localization problem, editrules uses internally the cateditmatrix object. cateditmatrix is an editmatrix object, which is described in De Jonge and Van der Loo (2011), but it "remembers" that it is categorical.

Figure 2 shows an example for the use of cateditmatrix. It also shows that categorical edits can be written in the form of a linear constraint to be used in an mixed integer problem solver. cateditmatrix can coerce an editarray. This method can be called explicitly by a user, but it it will be called implicitly in localizeErrors using `method="mip"`. An example is shown in figure 3

4 Discussion

Is a usefull addition to `editrules`, finds quickly solutions to error localization problems with hundreds to thousands of variables.

```
> dat <- data.frame(gender="male", pregnant=TRUE)
> localizeErrors(E,dat, method="mip")

Object of class 'errorLocation' generated at Wed Feb 22 09:48:16 2012
call : localizeErrors(E, dat, method = "mip")
method : mip
slots: $adapt $status $call $method $user $timestamp

Values to adapt:
      adapt
record gender pregnant
     1   TRUE    FALSE

 Status:
  weight degeneracy  user system elapsed maxDurationExceeded
1      1         NA 0.004      0   0.003               FALSE
```

Figure 3: Using localizeErrors method with categorical data

Solutions given by current lp solvers can be numerical unstable, which may result in a false positive or a false negative solution. Luckily `editrules` contains `substValue` and `isFeasible` that can be used together to check the validity of a solution. Furthermore several heuristics can be used to increase the numerical stability by using smaller boundaries for the variables.

4.1 Comparison to `backtracker`

errorLocalizer is more complete, offers a more complete tool box for finding an optimal solution. It can also find more equivalent solutions, which is not possible or difficult with MIP solvers.

References

(2012). *lpsolve reference guide*. lpsolve version 5.5.2.

De Jonge, E. and M. Van der Loo (2011). Manipulation of linear edits and error localization with the editrules package. Technical Report 201120, Statistics Netherlands, The Hague.

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 Americal Statistical Association 71*, 17–35.

Husson, F., S. Lê, and J. Pagès (2010). *Exploratory Multivariate Analysis by Example Using R.* Computer Sciences and Data Analysis. Chapman & Hall/CRC.

lpsolve and K. Konis (2011). *lpSolveAPI: R Interface for lpsolve version 5.5.2.0.* R package version 5.5.2.0-5.

Van der Loo, M. and E. De Jonge (2011). Manipulation of categorical data edits and error localization with the editrules package. Technical Report 201129, Statistics Netherlands.

Winkler, W. E. (1999). State of statistical data editing and current research problems. In *Working paper no. 29. UN/ECE Work Session on Statistical Data editing*, Rome.