Manipulation of categorical data edits and error localization with the editrules package

package version 1.5.2

Mark van der Loo and Edwin de Jonge

September 26, 2011

Abstract

This vignette is far from finished. Version 2.0 fo the package will have the full vignette. At the moment, **functionality for treating categorical data has beta status** *so bugs are likely.*

Refer to the accompanying paper De Jonge and Van der Loo (2011) for manipulation of linear edits.

Contents

1 Introduction

The value domain of categorical data records is usually limited by domain rules interrelating these variables. The simplest examples are cases where the value of one variable excludes values of another variable. For example: if the age class of a person is "child", then (by law) the marital status cannot be "married". In survey or administrative data, violations of such rules are frequently encountered. Resolving such violations is an important step prior to data analysis and estimation.

A categorical data record r with n variables may be defined as an element of the cartesian product space D:

$$D = D_1 \times D_2 \times \ldots \times D_n, \tag{1}$$

where each D_k is a finite set of d_k possible categories for variable k We label the categories as follows:

$$D_k = \{c \in D_k \,|\, c = 1, 2, \ldots, d_k\}. \tag{2}$$

Each restriction e is a subset of D and we say that that *if* $r \in e$ *then* r *violates* e. Conversely, when $r \notin e$ we say that r *satisfies* e. In data editing literature, such rules are referred to as *edit rules* or *edits*, in short. In the context of contingency tables they are referred to as *structural zeros* since each rule implies that one or more cells in the $d_1 \times d_2 \times \ldots \times d_n$ contingency table must be zero. A record is *valid* if it satisfies every edit imposed on D.

Large, complex surveys may consist of hundreds of interrelated rules and variables, which impedes resolution of edit violations and renders manual manipulation infeasible. Winkler (1999) mentions practical cases where statistical offices handle 250, 300 or 750 categorical edit rules for surveys.

The R package editrules offers functionality to define, manipulate and maintain sets of edit rules with relative ease. It also implements error localization functionality based on the generalized principle of (Fellegi and Holt, 1976), which states that one should find the smallest (weighted) number of variables whose values can be adapted such that all edits can be satisfied. Fellegi and Holt's principle should be considered as the last resort of data editing. It is useful in situations where a record violates one or more edits and there is no information about the cause of the error. In certain cases, the cause of error can be estimated with near certainty, for example in the case of typing errors in numerical data. We refer the reader to Scholtus (2008, 2009) and Van der Loo et al. (2011) for such cases.

The purpose of this paper is to give a technical overview of the representation and manipulation of edits in the editrules package, as well as some coded examples to get new users started.

2 Defining and checking categorical constraints

In the next section we describe the representation of edits and records as implemented in the editrules package. Readers not interested in the underlying principles may skip Section 2.1.

2.1 Boolean representation of records and edits

Categorical records may be represented as a vector of boolean values. A boolean vector of dimension d is an element of the boolean algebra

$$\mathbb{B}^d = \left(\{0,1\}^d, \wedge, \vee, \neg\right), \tag{3}$$

where 0 and 1 have the usual interpretations FALSE and TRUE and the logical operators work elementwise on their operands. To facilitate the discussion we will also allow the standard arithmetic operations addition and subtraction on boolean vectors (this is also consistent with the way R handles vectors of class logical).

To represent a record $r = (r_1, r_2, \ldots, r_n)$, assign to every category c in D_k a unique standard basisvector $\vec{\delta}_k(c)$ of $\mathbb{B}^{d_k}$. The boolean representation $\rho(r)$ of the full record is the direct sum

$$r \xrightarrow{\rho} \vec{\delta}_1(r_1) \oplus \vec{\delta}_2(r_2) \oplus \ldots \oplus \vec{\delta}_n(r_n), \tag{4}$$

which we will write as

$$\rho(r) = \mathbf{v}_1 \oplus \mathbf{v}_2 \oplus \cdots \oplus \mathbf{v}_n \equiv \mathbf{v}. \tag{5}$$

The dimension d of $\rho(r)$ is given by the total number of categories of all variables

$$d = \sum_{k=1}^{n} d_k. \tag{6}$$

When each record in a dataset is represented this way, summing the vectors yields the d-dimensional vecorized representation of the $d_1 \times d_2 \times \ldots \times d_n$ contingency table of the dataset. This is sometimes called the complete disjunctive table.

An edit e is a subset of D which can be written as the cartesian product

$$e = A_1 \times A_2 \ldots \times A_n, \text{ where } A_k \subseteq D_k,\ k = 1, 2, \ldots n. \tag{7}$$

The interpretation of an edit is that if a record $\mathbf{v} \in e$, then $\mathbf{v}$ is considered invalid. The following properties follow immediately.

Remark 2.1. *If $e \subset D$ and $e' \subset D$ are edits, then $e \cup e' = \{e, e'\}$ and $e \cap e' = A_1 \cap A_1' \times A_2 \cap A_2' \times \cdots \times A_n \cap A_n'$ are also edits.*

An edit, expressed as in Eq. (7) is said to be in normal form. A variable v_k is *involved* in an edit if $A_k \subset D_k$. Conversely, we say that e *involves* v_k if v_k is involved in e. A variable v_k for which $A_k = D_k$ is not involved in e. Since every category i of D_k is mapped to a unique basis vector $\vec{\delta}_k(i)$, edits have a boolean representation $\rho(e)$, given by

$$e \quad \xrightarrow{\rho} \quad \bigvee_{i \in A_1} \vec{\delta}_1(i) \oplus \bigvee_{i \in A_2} \vec{\delta}_2(i) \oplus \ldots \oplus \bigvee_{i \in A_n} \vec{\delta}_n(i), \tag{8}$$

which may simply be written as

$$\rho(e) = \mathbf{a}_1 \oplus \mathbf{a}_2 \oplus \ldots \oplus \mathbf{a}_n \equiv \mathbf{a}. \tag{9}$$

A simple example is given in Figure 1. It is easy to check that an edit contains variable k if and only if the inner product $\mathbf{1}_{d_k} \cdot \mathbf{a}_k < d_k$, where $\mathbf{1}_{d_k}$ is a d_k vector of ones.

A record $\mathbf{v}$ violates an edit if $v_k \in A_k$, $k = 1, 2, \ldots n$. In the boolean representation this can be written as a condition on the standard inner product between the boolean representation of a record and an edit:

$$\sum_{k=1}^{n} \vec{\delta}_k(v_k) \cdot \mathbf{a}_k = \rho(\mathbf{v}) \cdot \mathbf{a} = n. \tag{10}$$

Suppose that E is a set of edits of the form (7). It is not difficult to verify that an edit $e \in E$ is redundant if

$$A_k \quad = \quad \varnothing, \text{ for any } k \in 1, 2, \ldots, n \tag{11}$$

or

$$e \quad \subseteq \quad e' \text{ with } e' \in E. \tag{12}$$

In (11), e is redundant since it cannot contain any records. It can be tested by checking if any $\mathbf{1}_{d_k} \cdot \mathbf{a}_k = 0$. In the case of (12), e is redundant because any edit violating e also violates e'. Using $\rho(e) = \mathbf{a}$ and $\rho(e') = \mathbf{a}'$, this can be tested by checking if $\mathbf{a} \wedge \mathbf{a}' = \mathbf{a}$ or equivalently if $\mathbf{a} \vee \mathbf{a}' = \mathbf{a}'$.

In the editrules the boolean representation is mainly used to store edits and to manipulate them with methods like variable substitution and elimination. Data records can be stored in data.frame objects, as usual.

2.2 The editarray object

In the editrules package, a set of categorical edits is represented as an editarray object. Formally, we denote an editarray E for n categorical variables and m edits as (brackets indicate a combination of objects)

$$E = \langle \mathbf{A}, \mathbf{ind} \rangle, \text{ with } \mathbf{A} \in \{0, 1\}^{m \times d} \text{ and } d = \sum_{k=1}^{n} d_k, \tag{13}$$

Each row **a** of **A** contains the boolean representation of one edit, and the d_k denote the number of categories of each variable. The object **ind** is a nested list which relates columns of **A** to variable names and categories. Labeling variables with $k \in 1, 2, \ldots, n$ and category values with $c \in 1, 2, \ldots, d_k$, we use the following notations:

$$\mathbf{ind}(k, c) = \sum_{l<k} d_l + c \tag{14}$$

$$\mathbf{ind}(k) = \{\mathbf{ind}(k, c) \,|\, c \in D_k\}. \tag{15}$$

So $\mathbf{ind}(k, c)$ is the column index in **A** for variable k and category c and $\mathbf{ind}(k)$ is the set of column indices corresponding to the categories of variable k. The editarray is the central object for computing with catgegorical edits, just like the editmatrix is the central object for computations with linear edits.

It is both tedious and error prone to define and maintain an editarray by hand. In practice, categorical edits are usually stated verbosely, such as: "a male subject cannot be pregnant", or "an under-aged subject cannot be married". To facilitate the definition of edit arrays, editrules is equipped with a parser, which takes R-statements in character format, and translates them to an editarray.

Figure 1 shows a simple example of defining an editmatrix with the editrules package. The first two edits in Figure 1 define the data model. The editarray function derives the datamodel based on the variable names and categories it finds in the edits, whether they are univariate (defining domains) of multivariate. This means that if all possible variables and categories are mentioned in the multivariate edits, the correct datamodel will be derived as well.

The function datamodel accepts an edit array as input and returns an overview of variables and their categories for easy inspection. When printed to the screen, the boolean array is shown with column heads of the form

<abbreviated var. name><separator><abbreviated cat. label>,

where both variable names and categories are abbreviated for readability, and the standard separator is a colon (:). The separator may not occur as a symbol in either variable or category name, and its value can be determined by passing a custom sep argument to editarray.

Internally, editarray uses the R internal parse function to transform the character expressions to a parse tree, which is subsequently traversed recursively to derive the entries of the editmatrix. The opposite is also possible. The R internal function as.character has been overloaded to derive a character representation from a boolean representation. When printed to the screen, both the boolean and textual representation are shown.

The character expressions that can be read by editarray, such as

```
> E <- editarray(c(
+     "gender %in% c('male','female')",
+     "pregnant %in% c('yes','no')",
+     "if (gender == 'male') pregnant == 'no'"
+     )
+ )
> E

Edit array:
     levels
edits gndr:feml gndr:male prgn:no prgn:yes
   e1     FALSE      TRUE   FALSE     TRUE

Edit rules:
d1  :  gender %in% c('female', 'male')
d2  :  pregnant %in% c('no', 'yes')
e1  :  if( gender == 'male' ) pregnant != 'yes'

> datamodel(E)

  variable  value
1   gender female
2   gender   male
3 pregnant     no
4 pregnant    yes
```

Figure 1: Defining a simple editarray with the editarray function. The array is printed with abbreviated column heads, which themselves consist of variable names and categories seperated by a colon (by default). When printed to screen, a character version of the edits is shown as well, for readability.

```
"if ( gender == 'male' ) pregnant = 'no'"
```

follows standard R syntax, which should be already familliar to the reader. Note that double quotes are used to enclose the whole string, while single quotes are used for category names. Table 1 shows which operators and functions can be used to specify categorical edit rules.

Categories may be litaral characters, or booleans. It is worth noting that expressions on the right hand side of the %in% and == operators are evaluated. One useful application of this is that the categories, or data model can be defined outside of the edits:

```
> xval <- letters[1:4]
> yval <- c(TRUE,FALSE)
> editarray(c( "x %in% xval","y %in% yval","if ( x %in% c('a','b') ) !y "))

Edit array:
     levels
edits  x:a  x:b   x:c   x:d y:FALS y:TRUE
   e1 TRUE TRUE FALSE FALSE  FALSE   TRUE

Edit rules:
d1  :  x %in% c('a', 'b', 'c', 'd')
d2  :  y %in% c(FALSE, TRUE)
e1  :  if( x %in% c('a', 'b') ) y == FALSE
```

Table 1: Functions and operators that may be used to define edits with editarray

Operator	Description
%in%	Set membership*
==	Equality*
if(<condition>) <expression>	conditional statement
c('<cat1>','<cat2>',...)	categories, character or logical
&&, &	logical AND
||, |	logical OR
!	logical NOT

*Right-hand side is evaluated.

The above example also illustrates the use of boolean categories.

2.3 Coercion, checking, redundancy and infeasibility

Table 2 lists basic functions of editarray objects. The datamodel function retrieves the variables and categories in an edit array, and returns them as a two-column data.frame. With as.data.frame or as.character one can coerce an editarray so that it can be written to a file or database. Character coercion is used when edeits are printed to the screen. Optionally, coercing the datamodel to character form can be switched off. The result of as.data.frame version contains columns with edit names, a character representation of the edits and a column for remarks.

The function violatedEdits takes an editarray and a data.frame as input and returns a logical matrix indicating which record (rows) violate which edits (columns). It works by parsing the editarray to R-expressions and evaluating them within the data.frame environment. By default, the records are checked against the data model. This can be turned off by providing the optional argument datamodel=FALSE.

When manipulating edit sets, redundant edits of the form of Eq. (11) may arise. Such redundancies can be detected in the boolean representation with isObviouslyRedundant. By default, this function also checks for duplicate edits, but this may be turned off. The function duplicated is overloaded from the standard R function and the function isSubset (pseudocode in Algorithm 1) detects which edits are a subset or duplicate of another one. In the actual R implementation, the only explicit loop is a call to R's vapply function. The other loops are avoided using R's indexing and vectorization properties.

Manipulations may also lead to edits of the form $e = D$, in which case every possible record is invalid, and the editarray has become impossible to satisfy. The function isObviouslyInfeasible detects whether any such edits are present. They can be detected by checking if $\sum_{j=1}^{d} \rho(e)_j = d$.

Table 2: Basic functions for objects of class editarray. Only mandatory arguments are shown, refer to the built-in documentation for optional arguments.

Function	description
datamodel(E)	get datamodel
getVars(E)	get a list of variables
as.data.frame(E)	coerce edits to data.frame
contains(E)	which edits contains which variable
as.character(E)	coerce edits to character vector
violatedEdits(E,x)	check which edits are violated by x
isObviouslyRedundant(E)	find redundancies [Eq. (11)], duplicates
duplicated(E)	find duplicate edits
isSubset(E)	find edits, subset of another edit [Eq. (12)]
isObviouslyInfeasible(E)	detect contradictions
substValue(E,var,value)	substitute a value

Algorithm 1 isSubset(E)

Input: An editarray $E = \langle \mathbf{A}, \mathbf{ind} \rangle$.

$\mathbf{s} \leftarrow (\text{FALSE})^m$
for $(\mathbf{a}^{(i)}, \mathbf{a}^{(i')}) \in \text{rows}(\mathbf{A}) \times \text{rows}(\mathbf{A})$ **do**
  if $\mathbf{a}^{(i)} \vee \mathbf{a}^{(i')} = \mathbf{a}^{(i')}$ **then**
    $s_i \leftarrow \text{TRUE}$

Output: Boolean vector $\mathbf{s}$ indicating which edits represented by $\mathbf{A}$ are a subset of another edit.

3 Manipulation of categorical restrictions

The basic operations on sets of categorical edits are value substitution and variable elimination. The former amounts to adapting the datamodel underlying the edit set while the latter amounts to deriving relations between variables not involving the eliminated variable.

3.1 Value substitution

If it is assumed that in a record, one of the variables takes a certain value, that value may be substituted in the edit rules. In the boolean representation this amounts to removing all edits which exclude that value, since the record cannot violate those edits. Secondly, the columns related to the substituted variable, but not to the substituted category are removed, thus adapting the datamodel to the new assumption. Algorithm 2 gives the pseudocode for reference purposes.

In the editrules package, value substitution is performed by the substValue function, which accepts variable and category names. In the following

Algorithm 2 substValue(E,k,v)

Input: an editarray $E = \langle \mathbf{A}, \mathbf{ind} \rangle$, a variable index k and a value v

$i \leftarrow \mathbf{ind}(k, v)$

$\mathbf{A} \leftarrow \mathbf{A} \backslash \{\mathbf{a} \in \text{rows}(\mathbf{A}) \mid a_i = \text{FALSE}\}$ ▷ Remove rows not involving v

$\mathbf{A} \leftarrow \mathbf{A} \backslash \{\mathbf{a}_j^t \in \text{columns}(\mathbf{A}) \mid j \in \mathbf{ind}(k) \backslash i\}$ ▷ Remove categories $\neq v$

Update **ind**

Output: $\langle \mathbf{A}, \mathbf{ind} \rangle$ with v substituted for variable k.

example the editmatrix defined in Figure 1 is used.

```
> substValue(E, "gender", "female")

Edit array:
      levels
edits  gndr:feml prgn:no prgn:yes

Edit rules:
d1  :  gender == 'female'
d2  :  pregnant %in% c('no', 'yes')
```

In this case, the variable "gender" is substituted by the value "female". With the gender is fixed, the datamodel reduces to $\{\text{male}\} \times \{\text{no}, \text{yes}\}$ and the restriction "if male then pregnant = true" becomes meaningless and is therefore removed.

The R implementation of substValue has an extra option, allowing to choose if the datamodel is reduced or not, which by default is set to TRUE .

3.2 Variable elimination by category resolution

Given two edits e and e', with boolean representations $\mathbf{a}$ and $\mathbf{a}'$ respectively. We define the *resolution operator* $\mathfrak{R}_k$ as:

$$\begin{aligned} \mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}' = {} & \mathbf{a}_1 \wedge \mathbf{a}'_1 \oplus \ldots \oplus \mathbf{a}_{k-1} \wedge \mathbf{a}'_{k-1} \\ \oplus \quad & \mathbf{a}_k \vee \mathbf{a}'_k \oplus \mathbf{a}_{k+1} \wedge \mathbf{a}'_{k+1} \oplus \ldots \oplus \mathbf{a}_n \wedge \mathbf{a}'_n \end{aligned} \tag{16}$$

For two edit sets $\mathbf{A}$ and $\mathbf{A}'$, we also introduce the notation

$$\mathbf{A}\,\mathfrak{R}_k\,\mathbf{A}' = \{\mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}' \mid (\mathbf{a}, \mathbf{a}') \in \text{rows}(\mathbf{A}) \times \text{rows}(\mathbf{A}')\}. \tag{17}$$

Observe that the resolution operator has the following properties, relevant for record checking.

$$\rho(\mathbf{v}) \in \mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}' \quad \Rightarrow \quad \rho(\mathbf{v}) \in \mathbf{a} \vee \rho(\mathbf{v}) \in \mathbf{a}' \tag{18}$$

$$\rho(\mathbf{v}) \in \mathbf{a} \quad \Rightarrow \quad \rho(\mathbf{v}) \in \mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}' \vee \mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}' = \varnothing \tag{19}$$

That is, if a record violates $\mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}'$, it does so because it violates $\mathbf{a}$ and/or $\mathbf{a}'$. Therefore, $\mathbf{a}\,\mathfrak{R}_k\,\mathbf{a}'$ is also an edit in the sense that a record is invalid if it is

Algorithm 3 ELIMINATE(E,k)

Input: an editarray $E = \langle \mathbf{A}, \mathbf{ind} \rangle$, a variable index k

for $j \in \mathbf{ind}(k)$ **do**

$\mathbf{A}^+ = \{\mathbf{a} \in \text{rows}(\mathbf{A}) : a_j = \text{TRUE}\}$

$\mathbf{A}^- = \{\mathbf{a} \in \text{rows}(\mathbf{A}) : a_j = \text{FALSE}\}$

if $\mathbf{A}^+ = \varnothing$ **then**

$\mathbf{A} \leftarrow \varnothing$

break

$\mathbf{A} \leftarrow \mathbf{A}^+ \cup \mathbf{A}^+ \,\mathfrak{R}_k\, \mathbf{A}^-$

Delete rows which have ISSUBSET($\langle \mathbf{A}, \mathbf{ind} \rangle$) = TRUE.

Output: editarray $\langle \mathbf{A}, \mathbf{ind} \rangle$ with variable k eliminated

falls in the derived edit.When $\mathbf{a}_k = \mathbf{a}'_k$, the resulting edit is the intersection of the original edits, in which case the resulting edit is redundant.

The operator is called resolution operator since its action strongly resembles a resolution operation from formal logic. If $\mathbf{a}_k \vee \mathbf{a}'_k = (\text{TRUE})^{d_k}$, the operator "resolves" or eliminates the k^{th} variable and we are left with a relation between the other variables, regardless of the value of variable k. The edit resulting from a resolution operation on two explicitly defined edits is called an *implied edit*. If the resolution operation happens to eliminate one of the variables, it is called an *essentially new implied edit*. These terms were introduced by Fellegi and Holt (1976) who first solved the problem of error localization for categorical data.

The resolution operator can be used to eliminate a variable k from a set of edits represented by $\mathbf{A}$ category by category as follows (Algorithm 3). Suppose that j is the column index of the first category of k. Collect all pairs of $(\mathbf{a}^+, \mathbf{a}^-)$ obeying $a_j^+ = \text{TRUE}$ and $a_j^- = \text{FALSE}$. If there are no edits of type $\mathbf{a}^+$, the variable cannot be eliminated and the empty set is returned. Otherwise, copy all $\mathbf{a}^+$ to a new set of edits and add every $\mathbf{a}^+ \,\mathfrak{R}_k\, \mathbf{a}^-$. By construction, these new edits all have $a_j = \text{TRUE}$. Possibly, redundant edits have been produced, and these may be removed. The procedure is iterated for every category of k, adding a category for which each $a_j = \text{TRUE}$ at each iteration.

4 Error localization in categorical data

See the help on localizeErrors and errorLocalizer.

References

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.

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

Scholtus, S. (2008). Algorithms for correcting some obvious inconsistencies and rounding errors in business survey data. Technical Report 08015, Statistics Netherlands, Den Haag. The papers are available in the inst/doc directory of the R package or via the website of Statistics Netherlands.

Scholtus, S. (2009). Automatic correction of simple typing error in numerical data with balance edits. Technical Report 09046, Statistics Netherlands, Den Haag. The papers are available in the inst/doc directory of the R package or via the website of Statistics Netherlands.

Van der Loo, M., E. de Jonge, and S. Scholtus (2011). *deducorrect: Deductive correction of simple rounding, typing and sign errors.* R package version 1.0-1.

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.

A Notation

Symbol	Explanation and reference
$\mathbf{a}$	An edit, in boolean representation: $\mathbf{a} = \rho(e)$, Eq. (9).
$\mathbf{a}_k$	Boolean representation of one variable in $\mathbf{a}$. $\mathbf{a} = \oplus_{k=1}^{n} \mathbf{a}_k$.
$\mathbf{A}$	Set of edits, in $m \times d$ boolean representation.
c	Label for a single category of D_k.
D	Set (domain) of all possible categorical records, Eq. (1).
D_k	Set of possible categories for variable k. Eq. (2).
d	Number of categories (in total), Eq. (6).
d_k	Number of categories in D_k.
e	An edit, in set representation: $e \subseteq D$, [Eq. (7)].
E	An editarray, Eq. (13), or a set of edits in set representation.
ind	Function relating categories c of variable k to columns in $\mathbf{A}$, Eqs.(14) and (15).
i	row index in $\mathbf{A}$ (labeling edits).
j	column index in $\mathbf{A}$ (labeling categories).
m	Number of edits.
n	Number of variables.
r	Categorical record, in set representation: $r \in D$.
$\mathfrak{R}_k$	Resolution operator Eq. (16).
ρ	Map, sending set representation to boolean representation.
$\mathbf{v}$	Categorical record, in boolean representation: $\mathbf{v} = \rho(r)$.
$\mathbf{v}_k$	Boolean representation of a single variable $\mathbf{v} = \oplus_{k=1}^{n} \mathbf{v}_k$.