Automatic editing of numerical data with R

Mark van der Loo,* Edwin De Jonge, and Sander Scholtus

Statistics Netherlands, dpt. of Methodology and Quality, PO box 24500, 2490 HA The Hague, The Netherlands

(Dated: August 11th, 2011)

In 2010 Statistics Netherlands adopted R as a strategic tool for statistics production. In this paper we discuss two R extensions developed by the authors, which offer functionality for automatic data editing. The first package is deducorrect, which implements methods to find and correct rounding, typing and sign errors in numerical data. The second package is called editrules and offers functionality to define and manipulate edits as well as error localization based on Fellegi and Holt's principle. Both packages are available for download and install via the Comprehensive R Archive Network. We conclude with some remarks on planned extensions to categorical data.

For presentation at the EESW11 workshop, 12-14 September 2011, Neuchâtel (Switzerland).

I. INTRODUCTION

R is a powerful open source statistical computing environment and programming language which has evolved to an important world-wide standard over the last decade. In 2010 Statistics Netherlands accepted R as a strategic tool for the development of production systems. The main virtues of R are its statistical, graphical, and data processing capabilities, the presence of an active international community of developers and shallow learning curve compared to traditional low-level programming languages.

R is a modular environment, consisting of the R core which holds the language interpreter and basic statistical functionality, and extension packages offering specific functionality which can be loaded when necessary. The core is developed and maintained by the R Development Core Team (2011) while packages can be developed by users. Packages which meet certain documentation and technical standards may be published via the Comprehensive R Archive Network (CRAN, 2011). At the time of writing there are more than 3000 packages available via CRAN, of which a selection may be used for production purposes at Statistics Netherlands.

In 2010 Statistics Netherlands completed the "methods series" project, which aims to offer a uniform description of all (statistical) methods used in data collection, processing and delivery. The R center of expertise at Statistics Netherlands is working on projects to make appropriate methods available in R. Many methods, such as weighting, sampling and imputation are part of standard R functionality, while some have to be developed from scratch. Recently, the authors have released two R packages via CRAN, offering functionality for automated data editing:

- deducorrect: A package for correcting rounding, typing and sign errors in numerical data under linear restrictions (Van der Loo et al., 2011).
- editrules: A package for manipulation of edits and error localization based on the generalized principle of Felligi and Holt (De Jonge and van der Loo, 2011).

In the remainder of this paper, an overview of the functionality and algorithms behind the packages is given.

II. DEFINING AND MANIPULATING LINEAR EDITS WITH THE EDITRULES PACKAGE

The value domain of numerical (survey) data with n variables is usually restricted to a subset of $\mathbb{R}^n$ by linear restrictions of the form

$$\mathbf{a}'\mathbf{x} \odot b, \text{ where } \mathbf{a}, \mathbf{x} \in \mathbb{R}^n,\ b \in \mathbb{R} \text{ and } \odot \in \{<, \leq, =, \geq, >\}. \tag{1}$$

Here, $\mathbf{a}$ is a vector of coefficients and $\mathbf{x}$ a record of n numerical variables. Examples of such rule sets include account balances, positivity demands and demands on ratios between variables. For computational processing, sets of equalities and inequalities are most conveniently stored and manipulated in matrix form, while statisticians usually formulate

*Corresponding author: m.vanderloo@cbs.nl

```
> E <- editmatrix(c(
+ "cost + profit == turnover",
+ "profit <= 0.6*turnover",
+ "cost >= 0",
+ "profit >= 0",
+ "turnover >= 0"
+ ))
> E

Edit matrix:
   cost profit turnover Ops CONSTANT
e1    1      1     -1.0  ==        0
e2    0      1     -0.6  <=        0
e3   -1      0      0.0  <=        0
e4    0     -1      0.0  <=        0
e5    0      0     -1.0  <=        0

Edit rules:
e1 : cost + profit == turnover
e2 : profit <= 0.6*turnover
e3 : 0 <= cost
e4 : 0 <= profit
e5 : 0 <= turnover
```

Figure 1: Defining a set of linear (in)equalities from the R command line with the editmatrix function. The R-internal function c() concatenates a series of strings to a list. Edits may also be read from files or databases, using R's database connectivity features. The function editmatrix returns an object of class editmatrix, which is printed to screen both in numeric and textual representation.

such demands in terms of the variables, *e.g.* cost + profit = turnover or turnover ≥ 0. Translating such demands to matrix form manually is both tedious and error prone, which is why the editrules package is able to parse equations in textual form to matrix representation. For example, the set of edits

$$\text{cost} + \text{profit} = \text{turnover} \tag{2}$$

$$\frac{\text{profit}}{\text{turnover}} \leq 0.6 \tag{3}$$

$$\text{cost} \geq 0 \tag{4}$$

$$\text{profit} \geq 0 \tag{5}$$

$$\text{turnover} \geq 0, \tag{6}$$

can be defined from the R command line as shown in Figure 1.

The editmatrix function of the editrules package accepts R-expressions in character form and parses them to an internal format consisting of an augmented matrix $[\mathbf{A}|\mathbf{b}]$, where each row contains the coefficients of one edit of the form (1) and a list of comparison operators. This so-called editmatrix object is the central object of the editrules package, and is used to represent edits in the deducorrect package as well. When printed to screen, both the matrix and textual representation are shown.

Using R's database connectivity features, with the editrules package, edits can be read from a (user-maintained) database, also offering the opportunity of naming and commenting the rules. Editrules also parses edits in edit matrix form to text, by overloading the R-internal as.character function. Internally, edits are parsed from text to matrix by taking advantage of the property that R-expressions are first-class objects[1]. The textual expressions are casted to a parse tree using R's internal parse capabilities, which is subsequently processed in a recursive manner to build the editmatrix representation.

The package contains functions to check which records in a record set violate which edits, functionality for the detection of redundancies equivalent to $0 < 1$ or contradictions equivalent to $-1 > 0$. A full feasibility test on a set of edits is included as well. The feasibility test is based on Fourier-Motzkin elimination and checks for any

[1] In computer programming, a first-class object is an object which may be created, manipulated and reassigned like a common variable.

contradictions within a set of edits. Furthermore, edit matrices may be split into independent blocks and equality-rules can be transformed to echelon form. Van der Loo et al. (2011) fully describe the editrules package with examples and detailed pseudocode of the underlying algorithms.

III. DEDUCTIVE CORRECTION WITH THE DEDUCORRECT PACKAGE

Among the most common errors occurring in administrative or survey data are human-caused errors such as typing errors, sign errors, value swaps and rounding errors. The errors mentioned here share the properties that they can be recognized, and in certain cases immediately resolved. Scholtus (2008, 2009) describes specialized algorithms to solve typing, rounding and sign errors in numerical data under balance restrictions. These algorithms, or generalizations thereof have been implemented in the deducorrect package (Van der Loo et al., 2011).

The deducorrect package offers three functions with a uniform interface and output. These are

- correctSigns To correct for sign flips and/or value swaps in numerical data violating balance restrictions.
- correctTypos To correct for typing errors in numerical data violating balance restrictions.
- correctRounding To correct rounding errors in numerical data violating balance restrictions.

Each function accepts an object of class editmatrix, as defined with the editrules package and an R data.frame with numerical records. The functions detect and try to correct violated *equality restrictions*. That is, inequality restrictions are ignored in the sense that no attempt is made to repair them, except as a coincidental consequence of repairing violated equality restrictions. Solutions which produce new violations of inequality restrictions are rejected.

Each function returns an object of class deducorrect which holds not only the corrected records, but also a log of applied corrections, a status indicator, a timestamp and the user name of the user running R.

The functions correctRounding and correctTypos attempt to correct as many edit violations as possible, and may label the correction status as "partial", meaning that some, but not all edit violations could be repaired. The correctSigns function only applies corrections to records if it completely resolves the violated edits, equalities as well as inequalities. Both correctSigns and correctTypos are able to correct errors which are masked by rounding errors, using a user-definable tolerance for detecting such errors.

The function correctSigns checks a record for violated balance restrictions, and identifies the involved variables. Next, using a binary tree algorithm, combinations of sign flips and value interchanges are generated in attempts to resolve the edit violations. Because the size of the search space grows combinatorially with the number of involved variables, a user has several options to limit the number of attempted repairs. The most important options are limiting the maximum number of variables to change or limiting the number of repair actions (sign flip or value interchange) to try. It is also possible to fixate or release certain variables explicitly for adaptation.

The function correctTypos analyses which edits are violated by a record and determines the involved variables. Next, by solving a linear system of equalities, it computes new suggestions for each variable involved and computes the Damerau-Levenshtein distance between the original and suggested values. The Damerau-Levenshtein distance counts the number of permutations, deletions, substitutions and additions necessary to turn one string into another (Damerau, 1964; Levenshtein, 1966). If this distance is below a user-definable limit (default: 1), it is assumed that the edit violation is caused by a typographical error and the suggestion(s) are accepted.

The function correctRounding first detects the equality relations which are violated by a user-definable difference (default: 2) small enough to be attributed to a rounding error. All others are removed from the edit set. There are usually many possible solutions to the problem of making small changes to one or more variables in order to solve the rounding errors. In correctRounding a set of variables involved in the remaining edits is chosen at random under the condition that the solution is unique after choice is made. Next, using a QR-decomposition, new values for the variables are computed. If this does not cause any inequality violations, the solution is accepted. Otherwise a new set of variables are drawn, up to a user-definable (default: 10) number of times.

IV. ERROR LOCALIZATION WITH THE EDITRULES PACKAGE

If a record violates edits, and the values in the record do not give any clue to the cause of error, new assumptions are necessary to localize the erroneous values. The most common way forward is to assume that errors are distributed randomly and uniformly over the variables. This then yields the following problem which was first described and solved by Fellegi and Holt (1976):

> Find the smallest (weighted) number of variables which can be adapted so that the record may satisfy every explicitly defined edit and edits derived thereof.

Depending on the variable types, this problem can be reformulated in various types of programming problems, including, mixed integer or binary programming. See De Waal (2003) or, more recently, De Waal et al. (2011) for several approaches to the problem.

The editrules package implements the binary branch-and-bound algorithm for numerical data as defined in De Waal and Quere (2003). In this algorithm a binary tree is generated where branching is defined by assuming a variable to be correct or erroneous. If a variable is assumed correct its value must be substituted in the edit matrix, if a variable is assumed erroneous it must be eliminated from the set of edits by Fourier-Motzkin elimination. For this purpose a new and fast Fourier-Motzkin elimination procedure was developed in R of which the main features are that it avoids explicit loops as much as possible and uses a rule of Černikov (1960) (but see Williams (1986)) to significantly reduce the memory (and therefore time) complexity of derived edit generation.

After defining the editrules and reading a numerical record, error localization is executed in two steps, shown below. The edit matrix E corresponds to Eqs. (2)-(6), as defined in Figure 1.

```
> record <- c(cost=40, profit=70, turnover=100)
> el <- errorLocalizer(E,record)
> el$searchBest()

$w
[1] 1

$adapt
    cost   profit turnover
   FALSE     TRUE    FALSE
```

Here in the first line, a numerical record is created, which obviously fails the edits of Eq. (2) and (3). In the second line, an object of class backtracker is created with the errorLocalizer function. The third line invokes the searchBest method of the backtracker object, resulting in the lowest-weight solution to the error localization problem. Indeed, replacing the value of "profit" with 60 would resolve all edit violations without causing new ones.

Apart from searchBest, there are other solvers in the backtracking object returned by errorLocalizer, namely

- searchNext Search the next solution in the binary tree.
- searchAll Return all solutions encountered while traversing the binary tree in a branch-and-bound manner.
- searchBest Search all lowest-weight solutions. Return a random lowest-weight solution if multiple are found.
- reset Reset backtracker object to initial state.

The user has the option to assign individual weights to variables when creating the backtracker object. The order of tree-traversal is adapted according to those weights so that there is a high chance of finding the optimal solution quickly. Furthermore, there are options to limit the search space by defining maximum weight, a maximum number of variables to change and to limit the search time (default: 10 minutes).

The errorLocalizer function returns a backtracker object containing the root node of the binary search tree associated to the error localization problem. After a call to searchNext the backtracker object contains a stack of binary tree nodes representing the found solution or NULL if no solutions can be found. A subsequent call to searchNext will result in a solution with equal or lower weight than the previous solution, or NULL.

To facilitate extension of error localization to categorical data as well as extensions to applications with "soft edits", the underlying functionality to define generic backtracker objects is made public in the form of the backtracker function. This function accepts as arguments a set of root initialization statements, a set of R-expressions to be executed at left child nodes, a set of R-expressions to be executed at the right child node and a set of R-expressions to judge the state of a node (and possibly perform some administrative tasks). With the backtracker function, users are able to easily and flexibly define and execute their own binary programming problems. See De Jonge and van der Loo (2011) for an example of its use and the underlying algorithms.

V. CONCLUSIONS AND OUTLOOK

In this paper we described the functionality and implementation of data editing packages for R: packages deducorrect and editrules. Both packages implement functionality previously unavailable yet in R but which may be of great interest to the official statistics community. Package deducorrect offers functionality to detect and solve various common human-caused errors in numerical data. Package editrules offers error localization functionality as well as a toolbox for investigating and maintaining large edit sets.

In the near future the editrules package will be extended to be able to parse and localize errors in categorical and mixed data as well. Furthermore, we are currently investigating the possibilities of interfaces with standard LP solvers such as lp_solve (Berkelaar and others, 2011) or GLPK (Lee and Luankesorn, 2011). Such interfaces are likely to yield some performance gain over the all-R solution, at the cost of some flexibility in choosing solutions.

References

Berkelaar, M. and others (2011). *lpSolve: Interface to Lp_solve v. 5.5 to solve linear/integer programs.* R package version 5.6.6.

Černikov, S. N. (1960). Contraction of systems of linear inequalities. In *Soviet mathematics DOKLADY 1.*

CRAN (2011). The Comprehensive R Archive Network is a network of ftp and web servers around the world that store identical, up-to-date versions of code and documentation for R. http://cran.r-project.org.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM 7*, 171–176.

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

De Waal, T. (2003). *Processing of erroneous and unsafe data.* Ph. D. thesis, Erasmus University, Rotterdam.

De Waal, T., J. Pannekoek, and S. Scholtus (2011). *Handbook of statistical data editing and imputation.* Wiley handbooks in survey methodology. Hoboken, New Jersey: Wiley.

De Waal, T. and R. Quere (2003). A fast and simple algorithm for automatic editing of mixed data. *Journal of Official Statistics 19*, 383–402.

Fellegi, I. P. and D. Holt (1976). A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association 71*, 353.

Lee, L. and L. Luankesorn (2011). *glpk: GNU Linear Programming Kit.* R package version 4.8-0.5.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady 10*, 707–710.

Scholtus, S. (2008). Algorithms for correcting obvious inconsistencies and rounding errors in survey business data. Technical Report 08015, Statistics Netherlands, Den Haag.

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

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Van der Loo, M., E. De Jonge, and S. Scholtus (2011). Correction of rounding, typing and sign errors with the deducorrect package. Technical Report 201119, Statistics Netherlands, The Hague.

Williams, H. (1986). Fourier's method of linear programming and its dual. *The American mathematical monthly 93*, 681–695.