The `editrules` vignette

Edwin de Jonge and Mark van der Loo

April 1, 2011

Abstract

`editrules` is a package to define, parse, manipulate and check linear and other data restrictions in R. Verbose restrictions can be entered at the commandline or stored as a (text) file or a database. This vignette is under construction.

1 Introduction

Data processing methods always impose restrictions on the type and format of data that may enter the method. Well-known cases include linear (in)equality restrictions in optimizations or the rules used in data checking and cleaning prior to analysis. Rules used for the purpose of data cleaning include, but are usually not limited to the linear (in)equalities mentioned before. They might include restrictions on certain value combinations of categorical variables for instance. These restrictions are usually referred to as *edit rules*, or *edits* in short. For the remainder of this paper we will use these terms as well.

`editrules` is a package to define, parse and manipulate edits with `R` in a convenient way. Moreover, it is possible to apply edits to a dataset to obtain a list of edit violations per record. The current version can handle linear (in)equality restrictions. Future versions with more versatile edits are planned.

2 A simple example

Suppose you have the following data,

```
> balance <- data.frame(
+     cost      = c( 75, 300, 70),
+     profit    = c(125,  40, 10),
+     turnover  = c(200, 320, 80))
```

subject to the linear restrictions:

$$\text{turnover} = \text{profit} + \text{cost} \tag{1}$$

$$\frac{\text{profit}}{\text{turnover}} \leq 0.6 \tag{2}$$

In the `editrules` package these rules can be checked as follows[1].

[1] The extra brackets around assign statements are included only to force R to print the result after assignment.

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

Find out which record violates any edit:

```
> (valid <- checkRows(E, balance))
```

```
[1] FALSE FALSE  TRUE
```

and list the errors.

```
> listViolatedEdits(E, balance)
```

```
[[1]]
e2
 2

[[2]]
e1
 1

[[3]]
named integer(0)
```

So the first record violates edit e2, the second record violates e1 and the third record is clean.

3 Verbose input of linear (in)equality constraints

In many statistical and optimization problems, one has to represent a set of linear (in)equalities in matrix a matrix form:

$$Ax = x_0, \text{ and/or } Ax = x_0 \text{ and/or } Ax \leq x_0. \tag{3}$$

However, in practice, these restrictions are often formulated verbatim and have to be translated to the matrix form by the statistician. The editrules package facilitates this by allowing a user to write linear (in)equality constraints in R language and translating it to an S3 object of type editmatrix. Constraints can be entered as a text vector, as in the example above, or read from a data.frame whith the columns:

name	character names an edit
edit	character contains the edit in the form of an R expression
description	character a description of the edit.

For example, the edits in the previous section can be entered as

```
> (edits <- data.frame(
+     name  = c("balance","suspect"),
+     edit  = c("turnover == cost + profit",
+               "profit <= 0.6*turnover"),
+     description = c("balance check",
+                     "suspiciously high turnover")))
```

```
     name                          edit                description
1 balance turnover == cost + profit              balance check
2 suspect    profit <= 0.6*turnover suspiciously high turnover
```

Translating these verbose rules can be done with the `editmatrix` command.

```
> (E <- editmatrix(edits))

Edit matrix:
        cost profit turnover CONSTANT
balance   -1     -1      1.0        0
suspect    0      1     -0.6        0

Edit rules:
balance : turnover == cost + profit [ balance check ]
suspect : profit <= 0.6*turnover [ suspiciously high turnover ]
```

Here, `E` is an object of class `editmatrix` which is a standard `R` `matrix` with some extra attributes to store edit information. It can be coerced to a normal `matrix` object with `as.matrix`.

The edit matrix `E` contains information on following system of linear (in)equalities.

$$Ex = C \tag{4}$$

$$Ex \leq C \tag{5}$$

$$Ex < C \tag{6}$$

`getC` retrieves the constant part of the system of linear (in)equalities.

```
> getC(E)

balance suspect
      0       0
```

`getOps` retrieves the comparison operator part of the system of linear (in) equalities.

```
> getOps(E)

balance suspect
   "=="    "<="
```

It can be used to check if all `ops` are the same or split `E` into several smaller edit matrices. Splitting an `editmatrix` can be done with normal subsetting.

```
> E[getOps(E) == "<="]
```

```
Edit matrix:
        cost profit turnover CONSTANT
suspect    0      1     -0.6        0

Edit rules:
suspect : profit <= 0.6*turnover [ suspiciously high turnover ]
```

Note that using a second index is pointless, since it is not clear what this would mean for the `editrules` of the editmatrix.

If an `editmatrix` is created with `normalize = TRUE`, all edits will be transformed into `==`,`<` and/or $\leq$ form. This option facilitates the mixed specification of edits.

`getVars` retrieves the variables that are part of the linear (in) equalities.

```
> getVars(E)

[1] "cost"     "profit"   "turnover"
```

An important step in data correction and imputation is error location. Determining which variables are used in violated edits is a necessary part. This can be easily done using `getVars` and `violatedEdits`.