The `deducorrect` vignette

Mark van der Loo, Edwin de Jonge and Sander Scholtus

March 11, 2011

Abstract

This vignette is unfinished. Version 1.0 of the package will contain a full vignette.

Contents

1 Introduction

Survey data is often plagued with internal inconsistencies wich have to be repaired before reliable statistical analysis can take place. Establishment surveys in particular are prone to errors, since they can consist of many numerical variables, interelated by numerous mathematical relationships. In particular, this package focuses on linear equalities and inequalities, which may be entered in any of the forms

$$\begin{aligned}
Ax &= b && (1)\\
Ax &< b && (2)\\
Ax &\leq b && (3)\\
Ax &> b && (4)\\
Ax &\geq b && (5)
\end{aligned}$$

or combinations thereof. Here, every A is a matrix, x a numerical data record and b a constant vector. To read, store and manipulate these relationships, the package relies on functionality of the `editrules` package.

The algorithms of this package are generalisations of the algorithms described in Scholtus (2008) and Scholtus (2009). This vignette is aimend to point out the generalisations and to provide the user with some coded examples, including the examples mentioned in the references.

Both papers are included in the `/doc` directory of this package.

2 The `deducorrect` object

When transforming raw data to a form suitable for statistical processing, it is important to keep track of the applied transformations so they can be taken into account when interpreting the results of the statistical analyses. To facilitate logging, every `correct-` function of the package returns does not only return the corrected data, but also information on the applied corrections, a timestamp and the user running R. See Table 1 for an overview.

3 Correcting for rounding errors

3.1 How it works

4 Correcting for tying errors

4.1 How it works

4.2 Some simple examples

5 Correcting for sign errors and value swaps

5.1 How it works

The function `correctSigns` tries to correct records violating equality contraints as in Eq. (1) while making sure that possible inequality constraints as in Eqs.

Table 1: Contents of the `deducorrect` object. All slots can be accessed or reassigned through the `$` operator.

`corrected`	The input data with records corrected where possible.
`corrections`	A `data.frame` describing the corrections. Every record contains a row number, labeling the row in the input data, a variable name of the input data, the old value and the new value.
`status`	A `data.frame` with at least one column giving treatment information of every record in the input data. Depending on the `correct` function, some extra columns may be added.
`timestamp`	The date and time when the `deducorrect` object was created.
`generatedby`	The name of the function that called `newdeducorrect` to create the object.
`user`	The name of the user running R, deduced from the environment variables of the system using R.

(2)-(5) are not violated in the process. To do so it tries to change the sign of (combinations of) variables and/or swap the order of variables. Sign flips and value swaps are closely related since

$$-(x-y) = y - x, \tag{6}$$

These simple linear relations frequently occur in profit-loss accounts for example. Basically, `correctSigns` first tries to correct a record by changing one sign, if that doesn't yield any solution, it tries changing two, and so on. If the user allows value swaps as well, it starts by trying to correct the record with a single sign flip or variable swap, if no solution is found, a combination is tried, and so on. The algorithm only treats the variables which have nonzero coefficients in one of the violated rows of Eq. (1). Since the number of combinations grows exponentially with the number of variables to treat, the user is given some control over the volume of the search space to cover when trying solotions in a number of ways. First of all, the variables which are allowed to flip signs or variable pairs which may be interchanged simultaneously can be determined by the user. Knowledge of the origin of the data and meaning of the questionaire will usually give a good idea on which variables are prone to sign errors. For example, in surveys on profit-loss accounts, respondents sometimes erroneously submit the cost as a negative number. Secondly, the user may limit the maximum number of simultaneous sign flips and or value swaps that may be tested. The third option limiting the search space is to cut of the algorithm when the number of combinations, given a number of actions to try becomes too large.

To account for sign errors and variable swap errors which are masked by rounding errors, the user can provide a nonegative tolerance ε, so the set of equality constraints are checked as

$$|Ax - b| < \varepsilon, \tag{7}$$

where $|\cdot|$ indicates the elementwise absolute value. The default value of ε is the machine accuracy (`.Machine$double.eps`). See Algorithm (1) on page 4 for a more detailed description in pseudocode.

Algorithm 1 Record correction for `correctSigns`

Input:
- A numeric record x
- A tolerance, ε
- A set of equality and inequality constraints of the form

$$\begin{aligned} Ax - b &= 0 \\ Bx - c &\geq 0, \end{aligned}$$

- A list `flip` of variables whos signs may be fliped.
- A list `swap` of variable pairs whos values may be interchanged
- An integer `maxActions`
- An integer `maxCombinations`

if $|Ax - b| < \varepsilon$ (elementwise) **then**
  Set `status` to `valid` and we are done.
else
  Create a list `actions`, of length n containing those elements of `flip` and `swap` that affect variables that occur in violated rows of A.
  Create an empty list S.
  $k \leftarrow 0$
  while $S = \varnothing$ **and** $k < \min(\texttt{maxActions}, n)$ **do**
    if not $\binom{n}{k} > \texttt{maxCombinations}$ **then**
      $k \leftarrow k + 1$
      Generate all $\binom{n}{k}$ combinations of k actions.
      Loop over those combinations, applying them to x. Add solutions obeying $|Ax - b| < \varepsilon$ and $Ax - c \geq 0$ to S.
    end if
  end while
  if not $S = \varnothing$ **then**
    Compute solution weights and choose solution with minimum weight. Choose the first solution in the case of degeneracy.
    Set `status` to `corrected`
  else
    Set `status` to invalid
  end if
end if

5.2 Some simple examples

In this section we walk through most of the options of the `correctSigns` function. Let's generate some data[1]:

```
> (dat <- data.frame(
+      x = c( 3, 14, 15,  1, 17, 12.3),
+      y = c(13, -4,  5,  2,  7, -2.1),
+      z = c(10, 10,-10, NA, 10, 10 )))

     x    y   z
1  3.0 13.0  10
2 14.0 -4.0  10
3 15.0  5.0 -10
4  1.0  2.0  NA
5 17.0  7.0  10
6 12.3 -2.1  10
```

We subject this data to the rule

$$z = x - y. \tag{8}$$

With the `editrules` package, this rule can be parsed to an `editmatrix`.

```
> require(editrules)
> (E <- editmatrix(c("z == x-y")))

Edit matrix:
    x y z CONSTANT
e1 -1 1 1        0

Edit rules:
e1 : z == x - y
```

Obviously, not all records in `dat` obey this rule. Let's check it with a function from the `editrules` package:

```
> cbind(dat, violatedEdits(E, dat))

     x    y   z    e1
1  3.0 13.0  10  TRUE
2 14.0 -4.0  10  TRUE
3 15.0  5.0 -10  TRUE
4  1.0  2.0  NA    NA
5 17.0  7.0  10 FALSE
6 12.3 -2.1  10  TRUE
```

Records 1, 2, 3 and 6 violate the editrule, record 5 is valid and for record 4 validity cannot be established since it has no value for z. If `correctSigns` is called without any options, all variables x, y and z can be sign-flipped:

```
> sol <- correctSigns(E, dat)
> cbind(sol$corrected, sol$status)
```

[1] brackets are included only to force R to print the result

```
     x    y   z    status weight degeneracy nflip nswap
1  3.0 13.0 -10 corrected      1          1     1     0
2 14.0  4.0  10 corrected      1          1     1     0
3 15.0  5.0  10 corrected      1          1     1     0
4  1.0  2.0  NA      <NA>      0          0     0     0
5 17.0  7.0  10     valid      0          0     0     0
6 12.3 -2.1  10   invalid      0          0     0     0
```

```
> sol$corrections
```

```
  row variable old new
1   1        z  10 -10
2   2        y  -4   4
3   3        z -10  10
```

So, the first three records have been correcte by flipping the sign of z, y and z respectively. Since no weight parameter was given, the weight is just the number of variables whose have been sign-flipped. Record 4 is not treated, since validity could not be established, record 5 was valid to begin with and record 6 could not be repaired with sign flips. However, record 6 seems to have a rounding error. We can try to accomodate for that by allowing a tolerance when checking equalities.

```
> sol <- correctSigns(E, dat, eps = 2)
> cbind(sol$corrected, sol$status)
```

```
     x    y   z    status weight degeneracy nflip nswap
1  3.0 13.0 -10 corrected      1          1     1     0
2 14.0  4.0  10 corrected      1          1     1     0
3 15.0  5.0  10 corrected      1          1     1     0
4  1.0  2.0  NA      <NA>      0          0     0     0
5 17.0  7.0  10     valid      0          0     0     0
6 12.3  2.1  10 corrected      1          1     1     0
```

```
> sol$corrections
```

```
  row variable   old   new
1   1        z  10.0 -10.0
2   2        y  -4.0   4.0
3   3        z -10.0  10.0
4   6        y  -2.1   2.1
```

Indeed, changing the sign of y in the last record brings the record within the allowed tolerance. Suppose that we have so much faith in the value of z, that we do not wish to change it's sign. This can be done with the `fixate` option:

```
> sol <- correctSigns(E, dat, eps = 2, fixate = "z")
> cbind(sol$corrected, sol$status)
```

```
      x     y   z    status weight degeneracy nflip nswap
1  -3.0 -13.0  10 corrected      2          1     2     0
2  14.0   4.0  10 corrected      1          1     1     0
```

```
3 -15.0  -5.0 -10 corrected      2          1     2     0
4   1.0   2.0  NA      <NA>      0          0     0     0
5  17.0   7.0  10     valid      0          0     0     0
6  12.3   2.1  10 corrected      1          1     1     0
```

```
> sol$corrections
```

```
  row variable  old   new
1   1        x  3.0  -3.0
2   1        y 13.0 -13.0
3   2        y -4.0   4.0
4   3        x 15.0 -15.0
5   3        y  5.0  -5.0
6   6        y -2.1   2.1
```

Indeed, we now find solutions whitout changing z, but at the price of more sign flips. By the way, the same result could have been obtained by

```
> correctSigns(E, dat, flip = c("x", "y"))
```

The sign flips in record 1 and three have the same effect of a variable swap. Allowing for swaps can be done as follows.

```
> sol <- correctSigns(E, dat, swap=list(c("x","y")),
+     eps=2, fixate="z")
> cbind(sol$corrected, sol$status)
```

```
     x    y   z    status weight degeneracy nflip nswap
1 13.0  3.0  10 corrected      1          1     0     1
2 14.0  4.0  10 corrected      1          1     1     0
3  5.0 15.0 -10 corrected      1          1     0     1
4  1.0  2.0  NA      <NA>      0          0     0     0
5 17.0  7.0  10     valid      0          0     0     0
6 12.3  2.1  10 corrected      1          1     1     0
```

```
> sol$corrections
```

```
  row variable  old  new
1   1        x  3.0 13.0
2   1        y 13.0  3.0
3   2        y -4.0  4.0
4   3        x 15.0  5.0
5   3        y  5.0 15.0
6   6        y -2.1  2.1
```

Notice that apart from swapping, the algorithm still tries to correct records by flipping signs. What happens is that the algorithm first tries to flip the sign of x, then of y, and then it tries to swap x and y. Each is counted as a single action. If no solution is found, it starts trying combinations. In this relatively simple example the result turned out well. In cases with more elaborate systems of equalities and inequalities, the result of the algorithm becomes harder to predict for users. It is therefore in general advisable to

- Use as much knowledge about the data as possible to decide which variables to flip sign and which variable pairs to swap. The problem treated in section 5.3 is a good example of this.
- Keep `flip` and `swap` disjunct. It is better to run the data a few times times through `correctSigns` with different settings.

Not allowing any sign flips can be done with the option `flip=c()`.

```
> sol <- correctSigns(E, dat, flip = c(), swap = list(c("x", "y")))
> cbind(sol$corrected, sol$status)

     x    y   z    status weight degeneracy nflip nswap
1 13.0  3.0  10 corrected      1          1     0     1
2 14.0 -4.0  10   invalid      0          0     0     0
3  5.0 15.0 -10 corrected      1          1     0     1
4  1.0  2.0  NA      <NA>      0          0     0     0
5 17.0  7.0  10     valid      0          0     0     0
6 12.3 -2.1  10   invalid      0          0     0     0

> sol$corrections

  row variable old new
1   1        x   3  13
2   1        y  13   3
3   3        x  15   5
4   3        y   5  15
```

This yields less corrected records. However running the data through

```
> correctSigns(E, sol$corrected, eps = 2)$status

     status weight degeneracy nflip nswap
1     valid      0          0     0     0
2 corrected      1          1     1     0
3     valid      0          0     0     0
4      <NA>      0          0     0     0
5     valid      0          0     0     0
6 corrected      1          1     1     0
```

will fix the remaining edit violations, and yields code which is a lot easyer to interpret.

5.3 Sign errors in a profit-loss account

Here, we will work through the example of chapter 3 of Scholtus (2009). This example considers 4 records, labeled case a, b, c, and d, which can be defined in `R` as

```
> dat <- data.frame(
+    case = c("a","b","c","d"),
+    x0r = c(2100,5100,3250,5726),
+    x0c = c(1950,4650,3550,5449),
```

```
+     x0  = c( 150, 450, 300, 276),
+     x1r = c(   0,   0, 110,  17),
+     x1c = c(  10, 130,  10,  26),
+     x1  = c(  10, 130, 100,  10),
+     x2r = c(  20,  20,  50,   0),
+     x2c = c(   5,   0,  90,  46),
+     x2  = c(  15,  20,  40,  46),
+     x3r = c(  50,  15,  30,   0),
+     x3c = c(  10,  25,  10,   0),
+     x3  = c(  40,  10,  20,   0),
+     x4  = c( 195, 610,-140, 221))
```

A record consists of 3 balance accounts wose results have to add up to a total. Each $x_{i,r}$ denotes some kind of return, x_{ic} some kind of cost and x_i the difference $x_{i,r} - x_{i,c}$. There are operating, financial, provisions and exeptional incomes and expenditures. The differences x_0, x_1, x_2 and x_3 have to add up to a given total x_4. These linear restrictions can be defined with the use of the `editrules` package.

```
> require(editrules)
> E <-editmatrix(c(
+      "x0 == x0r - x0c",
+      "x1 == x1r - x1c",
+      "x2 == x2r - x2c",
+      "x3 == x3r - x3c",
+      "x4 == x0 + x1 + x2 + x3"))
> E

Edit matrix:
   x0 x0c x0r x1 x1c x1r x2 x2c x2r x3 x3c x3r x4 CONSTANT
e1  1   1  -1  0   0   0  0   0   0  0   0   0  0        0
e2  0   0   0  1   1  -1  0   0   0  0   0   0  0        0
e3  0   0   0  0   0   0  1   1  -1  0   0   0  0        0
e4  0   0   0  0   0   0  0   0   0  1   1  -1  0        0
e5 -1   0   0 -1   0   0 -1   0   0 -1   0   0  1        0

Edit rules:
e1 : x0 == x0r - x0c
e2 : x1 == x1r - x1c
e3 : x2 == x2r - x2c
e4 : x3 == x3r - x3c
e5 : x4 == x0 + x1 + x2 + x3
```

Checking which records violate what edit rules can be done with the `violatedEdits` function of `editrules`.

```
> violatedEdits(E, dat)

        e1    e2    e3    e4    e5
[1,] FALSE  TRUE FALSE FALSE  TRUE
[2,] FALSE  TRUE FALSE  TRUE FALSE
[3,]  TRUE FALSE  TRUE FALSE  TRUE
[4,]  TRUE  TRUE  TRUE FALSE  TRUE
```

So record 1 (case a) for example, violates the restrictions e_1: $x_1 = x_{1,r} - x_{1,c}$ and e_5, $x_1 + x_2 + sx_3 = x_4$. We can try to solve the inconsistencies by allowing the following flips and swaps:

```
> swap <- list(
+      c("x1r","x1c"),
+      c("x2r","x2c"),
+      c("x3r","x3c"))
> flip <- c("x0","x1","x2","x3","x4")
```

Trying to correct the records by just flipping and swapping variables indicated above corresponds to trying to solve the system of equations

$$\begin{cases} x_0 s_0 &= x_{0,r} - x_{0,c} \\ x_1 s_1 &= (x_{1,r} - x_{1,c})t_1 \\ x_2 s_2 &= (x_{2,r} - x_{2,c})t_2 \\ x_3 s_3 &= (x_{3,r} - x_{3,c})t_3 \\ x_4 s_4 &= x_0 s_0 + x_1 s_1 + x_2 s_2 + x_3 s_3 \\ (s_0, s_1, s_2, s_3, s_4, t_1, t_2, t_3) \in \{-1, 1\}^8 \;, \end{cases} \tag{9}$$

where every s_i corresponds to a sign flip and t_j corresponds to a value swap, see also Eqn. (3.4) in Scholtus (2009). Using the `correctSigns` function, we get the following.

```
> cor <- correctSigns(E, dat, flip = flip, swap = swap)
> cor$status

     status weight degeneracy nflip nswap
1 corrected      1          1     1     0
2 corrected      2          1     0     2
3 corrected      2          1     1     1
4   invalid      0          0     0     0
```

As expected from the example in the reference, the last record could not be corrected because the solution is masked by a rounding errors. This can be solved by allowing a tolerance of two measurements units (in this case 1).

```
> cor <- correctSigns(E, dat, flip = flip, swap = swap, eps = 2)
> cor$status

     status weight degeneracy nflip nswap
1 corrected      1          1     1     0
2 corrected      2          1     0     2
3 corrected      2          1     1     1
4 corrected      2          1     2     0

> cor$corrected

  case  x0r  x0c   x0 x1r x1c  x1 x2r x2c  x2 x3r x3c x3   x4
1    a 2100 1950  150   0  10 -10  20   5  15  50  10 40  195
2    b 5100 4650  450 130   0 130  20   0  20  25  15 10  610
3    c 3250 3550 -300 110  10 100  90  50  40  30  10 20 -140
4    d 5726 5449  276  17  26 -10   0  46 -46   0   0  0  221
```

The latter table corresponds exactly to Table 2 in the reference.

References

Scholtus, S. (2008). Algorithms for correcting obvious inconsistencies and rounding errors in business data. Technical Report 08015, Statistics Netherlands, Den Haag. *Accepted for publication in the Journal of Official Statistics.*

Scholtus, S. (2009). Automatic correction of simple typing error in numerical data with balance edits. Technical Report 09046, Statistics Netherlands, Den Haag. *Accepted for publication in the Journal of Official Statistics.*