Package 'rspa'

July 2, 2014

Maintainer Mark van der Loo <mark.vanderloo@gmail.com>

License GPL-3

Title Adapt numerical records to fit (in)equality restrictions with the Successive Projection Algorithm

LazyData no

Type Package

LazyLoad yes

Author Mark van der Loo

Description Based on (optionally sparse) quadratic optimization with the main algorithms implemented in C. Includes features for easy processing of many (smaller) records. The algorithm has been tested on fairly large optimization problems with up to a few million variables and several hundred thousand restrictions.

Version 0.1-5

Depends R (>= 2.13.0), editrules

URL https://github.com/markvanderloo/rspa

Date 2013-10-21

Suggests knitr, testthat

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-05-16 11:30:12

R topics documented:

rspa-package 2
adjust 2
adjusted 4
adjustedRecords 5
adjustRecords 6
sparseConstraints 7

Index **9**

`rspa-package` *A package for minimal vector adjustment.*

Description

A package for minimal vector adjustment.

Overview

Given a vector $\boldsymbol{x}^0$, and a set linear restrictions of the form $\boldsymbol{a}_i \cdot \boldsymbol{x}_i = b_i$ and/or $\boldsymbol{a}_i \cdot \boldsymbol{x}_i \leq b_i$ with $i = 1, 2, \ldots, m$. This package finds the nearest vector to $\boldsymbol{x}^0$ (in the (weighted) euclidean sense) that obeys all restrictions.

The package can handle large (sparse) problemes and has been tested by us on vectors with half a million variables and sixty-thousand constraints. All underlying algorithms have been implemented as a `C` library.

`adjust` *Adjust a data to meet linear (in)equality constraints*

Description

Adjust a vector $\boldsymbol{x}$ to meet constraints $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$.

Usage

```
adjust(object, ...)

## S3 method for class 'editmatrix'
adjust(object, x, w = rep(1, length(x)),
  method = c("dense", "sparse"), ...)

## S3 method for class 'sparseConstraints'
adjust(object, x, w = rep(1, length(x)),
  tol = 0.01, maxiter = 1000L, ...)
```

```
## S3 method for class 'matrix'
adjust(object, b, x, neq = length(b), w = rep(1,
  length(x)), tol = 0.01, maxiter = 1000L, ...)
```

Arguments

object	an R object describing constraints (see details)
...	Arguments to be passed to other methods
method	use dense or sparse matrix method.
b	Constant vector of the constraint system $Ax \leq b$
x	The vector to be adjusted
neq	the first neq linear relations are equalities.
w	A positive weight vector
tol	The maximum allowed deviation from the constraints (see details).
maxiter	maximum number of iterations

Value

Object of class adjusted.

Details

adjust is a generic function allowing several definitions of the constraints in object.

editmatrix If object is an editmatrix, the function will try to match the names of x to the variable names in object before further processing. In that case the length of x is unimportant, as long as all variables in object are also in x. Depending on the choice of method, object is converted to matrix or sparseConstraints format before solving the adjustment problem.

matrix If object is a matrix, you also need to provide the constant vector b and the number of equations neq to define the problem. It is assumed that the first neq rows of object and the first new elements of b correspond to equalities. No names are matched, so x must be in the correct order and must be of the right dimension. See sparseConstraints on how to translate a matrix problem to the sparse version.

sparseConstraints If object is of class sparseConstraints, the sparse method is used to adjust x. Some basic checks on x and w are performed, but no attempt is made to match names of x to those of object.

The tolerance tol is defined as the maximum absolute value of the difference vector $\boldsymbol{Ax} - \boldsymbol{b}$ for equalities. For inequalities, the difference vector is set to zero when it's value is lesser than zero (i.e. when the restriction is obeyed). The function keeps iterating until either the tolerance is met, the number of allowed iterations is exceeded or divergence is detected.

Note

adjust does not perform any consistency checks. When the system of constraints is contradictory (*e.g.* $x > 1$ and $x < 0$) this will result in either divergence or in exceeding the number of iterations.

Examples

```
# a very simple adjustment example
E <- editmatrix(expression(
x + y == 10,
x > 0,
y > 0
))

# x and y will be adjusted by the same amount
adjust(E, c(x=4,y=5))

# One of the inequalies violated
adjust(E, c(x=-1,y=5))

# Weighted distances: 'heavy' variables change less
adjust(E,c(x=4,y=5), w=c(100,1))

# if w=1/x0, the ratio between coefficients of x0 stay the same (to first order)
x0 <- c(x=4,y=5)
x1 <- adjust(E,x0,w=1/x0)

x0[1]/x0[2]
x1$x[1] / x1$x[2]
```

---

adjusted *Adjusted object*

Description

Adjusted object

Usage

```
## S3 method for class 'adjusted'
print(x, maxprint = 10, ...)
```

Arguments

x	an object of class adjusted
maxprint	max number of output values to print
...	parameters to pass to other methds

Details

A `adjusted` object contains the adjusted vector as well as some information on how the adjustment was achieved. In particular, it contains the following slots (to be accessed with the dollar operator):

- `$x`: the adjusted vector.
- `$accuracy`: Maximum deviance of `$x` from the constraints (see `adjust` for details).
- `$objective` : Square root of objective function $\sum_i (x_i - x_i^0)^2 w_i$.
- `$duration`: `proc_time` object showing time it took to run the adjustment. (See `proc.time`).
- `$niter`: Number of iterations.
- `$status`: A `character` string stating whether the adjustment was successful, aborted, or if the maximum number of iterations was reached before convergence.
- `$method`: `'sparse'` or `'dense'`.

See Also

`adjust`

`adjustedRecords` *Adjusted records*

Description

Adjusted records

Usage

```
## S3 method for class 'adjustedRecords'
print(x, ...)

## S3 method for class 'adjustedRecords'
summary(object, ...)

## S3 method for class 'adjustedRecords'
plot(x, ...)
```

Arguments

`x`	object of class `adjustedRecords`
`...`	additional parameters to pass to other methods
`object`	object of class `adjustedRecords`

Details

The adjustedRecords object contains adjusted data as well as some information on the adjusting process. In particular:

- $adjusted: data.frame similar to dat with adjusted values.
- $status: data.frame with the same number of rows as dat. Each row stores the information of one adjusted object.

When printed, only the first 10 rows of cbind(adjusted, status) are shown. Use summary for a quick overview. The plot method shows kernel density estimates of the accuracy and objective functions. To avoid densities at values below 0, the accuracy densities are evaluated under a sqrt-transform and transformed back before plotting. For the objective function values a log-transform is used.

See Also

adjustRecords

---

adjustRecords *Adjust records in a* data.frame

Description

A convenient wrapper around adjust that loops over all records in a data.frame

Usage

```
adjustRecords(E, dat, adjust = array(TRUE, dim = dim(dat)), w = rep(1,
  ncol(dat)), verbose = FALSE, ...)
```

Arguments

E	a editmatrix
dat	a data.frame
adjust	a nrow(dat) x ncol(dat) boolean array, indicating which fields must be adjusted.
w	a vector of length ncol(dat) or array of size adjust with adjustment weights.
verbose	print progress to console
...	extra options, passed through to adjust

Value

An object of class adjustedRecords

Details

This function is not written to be especiallty speedy or memory-efficient, but to offer a convenient interface to adjusting a `data.frame` of records.

See Also

`adjust`

`sparseConstraints` *Generate sparse set of constraints.*

Description

Generate sparse set of constraints.

Usage

```
sparseConstraints(x, ...)

## S3 method for class 'editmatrix'
sparseConstraints(x, tol = 1e-08, ...)

## S3 method for class 'matrix'
sparseConstraints(x, b, neq = length(b), tol = 1e-08, ...)

## S3 method for class 'data.frame'
sparseConstraints(x, b, neq = length(b), base = min(x[,
  2]), sorted = FALSE, ...)

## S3 method for class 'sparseConstraints'
print(x, range = 1L:10L, ...)
```

Arguments

`x`	R object to be translated to sparseConstraints format.
`...`	options to be passed to other methods
`tol`	Tolerance for testing where coefficients are zero
`b`	Constant vector
`neq`	The first `new` equations are interpreted as equality constraints, the rest as '<='
`base`	are the indices in `x[,1:2]` base 0 or base 1?
`sorted`	is `x` sorted by the first column?
`range`	integer vector stating which constraints to print

Value

Object of class `sparseConstraints` (see details).

Details

The `sparseConstraints` objects holds the system $\boldsymbol{Ax} \leq \boldsymbol{b}$ in column sparse format, outside of `R`'s memory. In `R`, it is a *reference object*. In particular, it is meaningless to

- Copy the object. You only will only generate a pointer to physically the same object.
- Save the object. The physical object is destroyed when `R` closes, or when `R`'s garbage collector cleans up a removed `sparseConstraints` object.

Examples

```
# define constraints from editmatrix object:
E <- editmatrix(expression(
     x1 + x8 ==  950,
     x3 + x4 ==  950 ,
     x6 + x7 == x8,
     x4 > 0
))

# generate sparseConstraints object
sparseConstraints(E)

# same constraints, from data.frame
rc <- data.frame(
   row = c( 1, 1, 2, 2, 3, 3, 3, 4),
   col = c( 1, 2, 3, 4, 2, 5, 6, 4),
  coef = c(-1,-1,-1,-1, 1,-1,-1,-1)
)
b <- c(-950, -950, 0,0)

sparseConstraints(rc, b, neq=3)


# same constraints, from dense matrix

A <- getA(E)
b <- getb(E)
sparseConstraints(A, b, neq=3)
```

Index

adjust, 2, *5–7*
adjusted, *3*, 4, *6*
adjustedRecords, 5
adjustRecords, *6*, 6

editmatrix, *6*

plot.adjustedRecords *(*adjustedRecords*)*, 5
print.adjusted *(*adjusted*)*, 4
print.adjustedRecords *(*adjustedRecords*)*, 5
print.sparseConstraints *(*sparseConstraints*)*, 7

rspa-package, 2

sparseConstraints, *3*, 7
summary.adjustedRecords *(*adjustedRecords*)*, 5