

# Package ‘rspa’

July 23, 2025

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

**License** GPL-3

**Title** Adapt Numerical Records to Fit (in)Equality Restrictions

**Type** Package

**LazyLoad** yes

**Author** Mark van der Loo

**Description** Minimally adjust the values of numerical records in a data.frame, such that each record satisfies a predefined set of equality and/or inequality constraints. The constraints can be defined using the 'validate' package. The core algorithms have recently been moved to the 'lintools' package, refer to 'lintools' for a more basic interface and access to a version of the algorithm that works with sparse matrices.

**Version** 0.2.8

**Depends** R (>= 2.13.0)

**Imports** graphics, stats, validate, lintools

**Suggests** editrules, tinytest

**URL** <https://github.com/markvanderloo/rspa>

**BugReports** <https://github.com/markvanderloo/rspa/issues>

**RoxygenNote** 7.2.1

**Encoding** UTF-8

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2022-12-22 10:00:02 UTC

## Contents

rspa-package	2
match_restrictions	2
remove_tag	4
tagged_values	4
tag_missing	5

## Index

6

---

rspa-package	<i>A package for minimal vector adjustment.</i>
--------------	-------------------------------------------------

---

## Description

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

## Details

Much of this package's functionality, including algorithms for working with large, sparse problems has been moved to the `lintools` package. This package will serve as a front-end for application of the successive projection algorithm for data stored in `data.frame` like objects.

---

match_restrictions	<i>Alter numeric data records to match linear (in)equality constraints.</i>
--------------------	-----------------------------------------------------------------------------

---

## Description

Apply the successive projection algorithm to adjust each record in `dat` to satisfy a set of linear (in)equality constraints.

## Usage

```
match_restrictions(
  dat,
  restrictions,
  adjust = NULL,
  weight = rep(1, ncol(dat)),
  remove_tag = TRUE,
  ...
)
```

## Arguments

<code>dat</code>	A <code>data.frame</code>
<code>restrictions</code>	An object of class <code>validator</code>
<code>adjust</code>	(optional) A logical matrix of dimensions <code>dim(dat)</code> where TRUE indicates that a value may be adjusted. When missing, the <code>tagged_values</code> are used. If no tagging was applied, <code>adjust</code> will default to an all TRUE matrix with dimensions equal to <code>dim(dat)</code> .
<code>weight</code>	A weight vector of length <code>ncol(dat)</code> or a matrix of dimensions <code>dim(dat)</code> .
<code>remove_tag</code>	if a value position indicator is present, remove it?
<code>...</code>	arguments passed to <code>project</code> .

**Value**

dat, with values adapted.

**Note on inequality restrictions**

All inequality restrictions of the form  $a.x < b$  are treated as  $a.x \leq b$ . The idea is to project the original record  $x$  onto the boundary defined by the (in)equations. Projection on a boundary defined by a strict inequation is illdefined since the value  $b$  in the restriction  $a.x < b$  is strictly outside the valid region.

**See Also**

[tag\\_missing](#)

**Examples**

```
# a very simple adjustment example

v <- validate::validator(
  x + y == 10,
  x > 0,
  y > 0
)

# x and y will be adjusted by the same amount
match_restrictions(data.frame(x=4,y=5), v)

# One of the inequalities violated
match_restrictions(data.frame(x=-1,y=5), v)

# Weighted distances: 'heavy' variables change less
match_restrictions(data.frame(x=4,y=5), v, weight=c(100,1))

# if w=1/x0, the ratio between coefficients of x0 stay the same (to first order)
x0 <- data.frame(x=4,y=5)
x1 <- match_restrictions(x0, v, weight=1/as.matrix(x0))

x0[,1]/x0[,2]
x1[,1] / x1[,2]

# example of tag usage
v <- validate::validator(x + y == 1, x>0,y>0)
d <- data.frame(x=NA,y=0.5)
d <- tag_missing(d)
# impute
d[1,1] <- 1

# only the tagged values will be altered. The tag is
# removed afterwards.
match_restrictions(d,v)
```

---

remove_tag	<i>Remove cell position tags</i>
------------	----------------------------------

---

**Description**

Remove cell position tags

**Usage**

```
remove_tag(dat, ...)
```

**Arguments**

dat	[data.frame]
...	Currently not used

**Value**

dat with tag removed

**See Also**

Other tagging: [tag\\_missing\(\)](#), [tagged\\_values\(\)](#)

---

tagged_values	<i>Retrieve tagged cell positions</i>
---------------	---------------------------------------

---

**Description**

Retrieve tagged cell positions

**Usage**

```
tagged_values(dat, ...)
```

**Arguments**

dat	[data.frame]
...	Currently not used

**Value**

A logical matrix, or NULL

**See Also**

Other tagging: [remove\\_tag\(\)](#), [tag\\_missing\(\)](#)

---

tag_missing	<i>Tag currently missing elements of a data.frame</i>
-------------	-------------------------------------------------------

---

**Description**

Attach an attribute that marks which cells are empty (NA).

**Usage**

```
tag_missing(dat, ...)
```

**Arguments**

dat	[data.frame] to be tagged
...	Currently not used.

**Value**

dat, tagged for missing values.

**See Also**

Other tagging: [remove\\_tag\(\)](#), [tagged\\_values\(\)](#)

# Index

## \* tagging

- remove\_tag, 4
- tag\_missing, 5
- tagged\_values, 4

match\_restrictions, 2

project, 2

remove\_tag, 4, 4, 5

rspa-package, 2

tag\_missing, 3, 4, 5

tagged\_values, 2, 4, 4, 5

validator, 2