Package ‘cuadramelo’

October 10, 2024

Type Package

Title Matrix Balancing and Rounding

Version 1.0.0

Description Balancing and rounding matrices subject to restrictions. Adjustment of matrices so that columns and rows add up to given vectors, rounding of a matrix while keeping the column and/or row totals, performing these by blocks...

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports CVXR, dplyr, methods, utils

URL `https://mserrano-ine.github.io/cuadramelo/`

Suggests knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 2.10)

NeedsCompilation no

Author Miguel Serrano [aut, cre]

Maintainer Miguel Serrano <`miguel.serrano.martin@ine.es`>

Repository CRAN

Date/Publication 2024-10-10 16:20:11 UTC

Contents

`apply_by_block` *Modify matrix by blocks*

Description

Applies a function to a matrix by horizontal or vertical blocks.

Usage

```
apply_by_block(Y, layout, L, FUN, ...)
```

Arguments

`Y`	Matrix
`layout`	Blocks are distributed: 1 horizontally, 2 vertically.
`L`	Number of lines of the block.
`FUN`	Funtion to apply to the block.
`...`	Arguments to be passed to FUN.

Value

A matrix.

`balance_by_blocks` *Balance matrix by blocks*

Description

Applies `balance_matrix()` to equally-sized blocks that partition the matrix either vertically or horizontally.

Usage

```
balance_by_blocks(Y, col_totals = NULL, row_totals = NULL, layout, L)
```

Arguments

`Y`	Matrix to be balanced.
`col_totals`	Desired colSums for each block. See details.
`row_totals`	Desired rowSums for each block. See details.
`layout`	The blocks are distributed: 1 horizontally, 2 vertically.
`L`	Number of lines that a block encompasses.

Details

When Y is composed of **vertically** stacked blocks, col_totals must be a matrix whose rows are the colSums for each block, and row_totals just a (vertical) vector.

When Y is composed of blocks arraged **horizontally**, col_totals is a (horizontal) vector, and row_totals is a matrix whose columns are the rowSums for each block.

Value

The balanced matrix.

Examples

```
set.seed(10)
Y <- (rnorm(32)*10) |> matrix(ncol = 2) |> round(3)
v <- aggregate(Y, by = list(rep(1:4, times = rep(4,4))), FUN = sum)[, -1] |>
  round() |> as.matrix()
X <- balance_by_blocks(Y, v, layout = 2, L = 4)
U <- Y[5:8,] |> balance_matrix(v[2,])
X[5:8,] - U
```

`balance_matrix` *Balance matrix*

Description

Balances a matrix so that the columns and/or rows add up to a certain vector.

Usage

```
balance_matrix(Y, col_totals = NULL, row_totals = NULL, allow_negative = TRUE)
```

Arguments

`Y`	Matrix to be balanced.
`col_totals`	(optional) Desired sum of columns.
`row_totals`	(optional) Desired sum of rows.
`allow_negative`	Are negative entries in the balanced matrix allowed?

Details

Balancing is done according to the criteria of minimum sum of squares.

If neither `col_totals` nor `row_totals` is given, the same matrix will be returned. If only one of them is given, only that axis will be balanced.

Value

The balanced matrix.

Examples

```
set.seed(2)
Y <- rnorm(3*5) |> matrix(3,5) |> round(3)
v <- c( 0.876, -1.078, 3.452, 0.261, 1.349)
h <- c(-1.851, 0.243, 6.468)
X1 <- balance_matrix(Y, v, h)
Y
X1
h
rowSums(X1)
v
colSums(X1)
X3 <- balance_matrix(Y, col_totals = v)
v
colSums(X3)
X4 <- balance_matrix(Y, row_totals = h)
h
rowSums(X4)
```

`make_non_negative` *Make non-negative*

Description

Modifies as little as possible the entries of a matrix in order to make them non-negative, keeping row and column totals unchanged.

Usage

```
make_non_negative(Y, allowSlack = FALSE)
```

Arguments

`Y`	Matrix to be positivized.
`allowSlack`	Can colSums and rowSums be modified?

Value

A non-negative matrix, except if it is impossible to balance the matrix.

Examples

```
Y <- c(1,2,-1,1,
       2,2,3,1,
       1,1,-2,3) |>
       matrix(nrow = 3)
X <- make_non_negative(Y)
Y
X |> round(2)
```

```
rowSums(Y)
rowSums(X)
colSums(Y)
colSums(X)
set.seed(2)
Y <- rnorm(3*5) |> matrix(3,5) |> round(3)
Y
tryCatch(make_non_negative(Y), error = function(e) {
  print(e)
})
make_non_negative(Y, allowSlack = TRUE) |> round()
```

---

round_by_blocks *Round matrix by blocks*

Description

Applies `round_matrix()` to equally-sized blocks that partition the matrix either vertically or horizontally.

Usage

```
round_by_blocks(Y, layout, L, digits = 0, MARGIN_BLOCK = 0)
```

Arguments

`Y`	Matrix.
`layout`	The blocks are distributed: 1 horizontally, 2 vertically.
`L`	Number of lines that a block encompasses.
`digits`	Number of decimal places to be rounded to.
`MARGIN_BLOCK`	For each block • 0 Preserves the rounded colSums and rowSums. • 1 Preserves the rounded rowSums independently of each other. • 2 Preserves the rounded colSums independently of each other.

Value

The rounded matrix.

Examples

```
set.seed(10)
Y <- (rnorm(32)*10) |> matrix(ncol = 2) |> round(3)
X <- round_by_blocks(Y, 2, 4)
U <- Y[5:8,] |> round_matrix()
X[5:8,] - U
```

`round_matrix` *Round a matrix*

Description

Returns an integer matrix that preserves the rounded colSums and rowSums.

Usage

```
round_matrix(Y, digits = 0, MARGIN = 0)
```

Arguments

`Y`	A matrix.
`digits`	Decimal places to round to.
`MARGIN`	One of • 0 Preserves the rounded colSums and rowSums. • 1 Preserves the rounded rowSums independently of each other. • 2 Preserves the rounded colSums independently of each other.

Details

The function will throw a *warning* if the problem is infeasable. To be able to round the matrix in this fashion, the following things must be equal:

- the sum of the differences between the row totals and the rounded row totals
- the sum of the differences between the column totals and the rounded row totals

Value

The rounded matrix.

Examples

```
set.seed(6)
Y <- rnorm(3*5)*10 |> matrix(3,5) |> round(3)
X <- round_matrix(Y)
Y
X
colSums(Y) |> round()
colSums(X)
rowSums(Y) |> round()
rowSums(X)
```

`round_vector` *Round univariate*

Description

Rounds a vector preserving the rounded sum.

Usage

```
round_vector(x, digits = 0)
```

Arguments

`x`	A vector.
`digits`	Number of decimal places to be rounded to.

Value

description

Examples

```
set.seed(4)
x <- (rnorm(5)*10) |> abs()
y <- round_vector(x)
cbind(x, y)
round(sum(x)) - sum(y)
```

Index