Package 'WIPF'

May 8, 2026

Type Package

Title Weighted Iterative Proportional Fitting

Version 0.1.0-4

Description Implementation of the weighted iterative proportional fitting (WIPF) procedure for updating/adjusting a N-dimensional array given a weight structure and some target marginals.
Acknowledgements:
The author wish to thank Conselleria de Educación, Cultura, Universidades y Empleo (grant CIAICO/2023/031), Ministerio de Ciencia, Innovación y Universidades (grant PID2021-128228NB-I00) and Fundación Mapfre (grant 'Modelización espacial e intra-anual de la mortalidad en España. Una herramienta automática para el cálculo de productos de vida') for supporting this research.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Jose M. Pavía [aut, cre] (ORCID:
<https://orcid.org/0000-0002-0129-726X>)

Maintainer Jose M. Pavía <jose.m.pavia@uv.es>

Repository CRAN

Date/Publication 2026-03-06 20:40:02 UTC

Contents

array2df *Array to long-format data frame*

Description

Organizes the information in an array with N dimensions into a long-format data frame with N + 1 columns, where the last column contains the values of the array.

Usage

```
array2df(arr, value_name = "value")
```

Arguments

arr	An array of N-dimensiones
value_name	Name for the column containing the values of the array.

Author(s)

Jose M. Pavia, <pavia@uv.es>

df2array *Long-format data frame to array*

Description

Organizes the information in a long-format data frame with N factor columns and one value column into a K-dimensional array ($K \leq N$), where the size of each dimension corresponds to the number of levels of the respective factor.

Usage

```
df2array(
  df,
  margins = 1:(ncol(df) - 1),
  values = ncol(df),
  NA2zeros = TRUE,
  names = TRUE,
  ...
)
```

Arguments

df	A long-format data frame.
margins	A vector of integers indicating which columns of df should be mapped to the dimensions of the array, and in what order. By default, 1:(ncol(df)-1), meaning all columns of df except the last are mapped to dimensions in the order they appear in the data frame.
values	An integer specifying which column of df is mapped to the cell values of the array. By default, ncol(df), meaning that the values of the last column of df are used as the cell values.
NA2zeros	A TRUE/FALSE argument indicating whether intersections of levels not present in df should be imputed with zero. Default, TRUE.
names	A TRUE/FALSE argument indicating whether the level labels of the factors should be used as dimension names (dimnames) for the array. Default is TRUE.
...	Other arguments to be passed to the function. Not currently used.

Value

An array with length(margins) dimensions.

Author(s)

Jose M. Pavia, <pavia@uv.es>

Examples

```
x <- structure(list(LF1 = c(1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L,
                            1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L,
                            2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L,
                            2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L,
                            3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L,
                            3L, 3L, 3L, 3L, 3L, 3L, 3L, 4L, 4L, 4L, 4L, 4L, 4L,
                            4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L, 4L,
                            4L, 4L, 4L, 4L, 4L),
                       LF2 = c(1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 2L, 3L,
                            3L, 3L, 3L, 3L, 4L, 4L, 4L, 4L, 4L, 4L, 1L, 1L,
                            1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L,
                            3L, 3L, 3L, 4L, 4L, 4L, 4L, 4L, 4L, 1L, 1L, 1L, 1L,
                            1L, 1L, 2L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L,
                            3L, 4L, 4L, 4L, 4L, 4L, 4L, 1L, 1L, 1L, 1L, 1L, 1L,
                            2L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L, 3L, 4L,
                            4L, 4L, 4L, 4L, 4L),
                       LF3 = c(1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L,
                            2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L,
                            3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L,
                            4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L,
                            5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L,
                            6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L,
                            1L, 2L, 3L, 4L, 5L, 6L, 1L, 2L, 3L, 4L, 5L, 6L, 1L,
                            2L, 3L, 4L, 5L, 6L),
```

```
                    ix = c(0.8812, 0.8887, 1.0035, 0.8782, 1.1580, 1.3894, 0.7986,
                           1.0170, 1.0875, 0.9499, 0.9524, 1.1707, 0.4907, 1.4251,
                           0.8045, 0.7830, 0.7144, 0.9673, 0.5705, 6.8399, 0.6700,
                           0.6110, 2.1088, 0.7673, 0.8206, 1.0989, 1.0824, 0.7626,
                           1.1863, 1.6287, 0.8107, 0.8689, 1.0907, 0.9404, 0.9957,
                           1.2035, 0.5604, 0.9439, 0.8367, 0.7845, 0.8614, 1.0996,
                           0.3270, 1.1892, 0.6776, 0.5313, 0.7801, 0.9651, 1.2576,
                           1.1939, 1.2554, 1.1225, 1.5741, 1.5718, 0.8092, 0.8460,
                           1.0899, 1.0742, 1.0668, 1.0680, 0.8204, 0.8988, 1.0015,
                           1.0354, 0.9541, 1.0639, 0.5223, 0.6963, 0.6749, 0.7230,
                           0.6616, 0.9579, 1.1752, 1.3359, 1.2824, 1.6836, 1.5313,
                           2.5715, 1.0579, 0.8304, 1.0632, 1.0016, 0.9370, 1.1711,
                           0.7874, 1.4360, 1.0949, 0.8646, 0.8430, 1.4736, 0.7795,
                           0.9362, 0.8489, 0.8246, 0.8449, 0.6331)),
                           row.names = c(NA, -96L), class = "data.frame")

example <- df2array(df = x, margins = c(2, 1, 3))
```

`WIPF` *Weighted Iterative Proportional Fitting (WIPF) in N (greater than 1) dimensions*

Description

Implements the Weighted Iterative Proportional Fitting (WIPF) algorithm to adjust an N-dimensional array subject to weighted marginal constraints.

Usage

```
WIPF(
  seed,
  weights,
  margins,
  indices,
  normalize = TRUE,
  tol = 10^-6,
  maxit = 1000,
  full = FALSE,
  ...
)
```

Arguments

`seed`	An N-dimensional array of non-negative values with the initial values.
`weights`	An N-dimensional array of non-negative values with the weights associated to each entry of the `seed` array.
`margins`	A list of arrays of non-negative values of order less than N with the target (weighted) marginal sums over the specified dimensions.

indices	A list of vectors with the same length as margins, indicating the indices corresponding to each array in margins.
normalize	TRUE/FALSE argument indicating whether the weights should be normalized across all dimensions before constructing the weighted sums for comparison with the margin values. Default is TRUE. Normalization is essential when adjusting a set of indices for which the margins represent theoretical convex combinations of the inner indices. This characterizes a typical context in which WIPF may be of value.
tol	Stopping criterion. The algorithm stops when the maximum difference between the weighted sums of the values to be fitted and the margins is lower than the value specified by tol. Default, 0.000001.
maxit	Stopping criterion. A positive integer number indicating the maximum number of iterations allowed. Default, 1000. The algorithm will stop if the values to be fitted still has not converged after these many iterations.
full	TRUE/FALSE argument indicating if either only the solution should be saved or a more complete output. Default FALSE.
...	Other arguments to be passed to the function. Not currently used.

Details

The function updates an initial N-dimensional array (referred to as the seed) using an N-dimensional array of weights to align with a collection of lower-dimensional margins of different orders, some of which may be missing. When all provided margins are compatible given the weights, the updated array ensures that the corresponding weighted sums over all specified index subsets coincide with the supplied margins. If the provided margins are incompatible given the weights, the functions WIPF1 and WIPF are applied to the provided margins to guarantee their compatibility with the weights. In these cases, the margins are updated in increasing order of sub-indices, with higher-order indices running faster.

Value

When full = FALSE an object similar to seed with the solution reached when the algorithm stops. When full = TRUE a list with the following components:

sol	An object similar to seed with the solution reached at convergence (or when the maximum number of iterations is reached).
iter	Number of iterations when the algorithm stops.
margins	A list of arrays similar margins with the actual margins used to reach the solution. Each array whose margins are compatible given the weights coincides with the original array.
dev.margins	A list of arrays similar to the margins output containing the absolute maximum deviations between the values in margins and the corresponding weighted sums of the values in sol.
dev.congruence	A list of arrays, similar to the margins output, containing the differences between the values in margins and those in inputs$margins. In other words, it shows the difference between the final margins actually used (after adjusting them, if necessary, to be compatible with the weights) and the original margins provided by the user.

`inputs` A list containing all the objects with the values used as arguments by the function.

Note

Weighted Iterative Proportional Fitting is an extension of IPF. WIPF produces the same solutions than IPF with all weights being ones and when they are not normalized. IPF is also known as RAS in economics, raking in survey research or matrix scaling in computer science.

Author(s)

Jose M. Pavia, <pavia@uv.es>

Examples

```
s <- structure(c(0.9297, 0.9446, 0.8763, 0.92, 0.8655, 0.8583, 0.8132,
                 0.8679, 0.7968, 0.7834, 0.721, 0.7859, 0.7747, 0.7851, 0.8632,
                 1.041, 1.5617, 1.5642, 1.4847, 1.5176, 1.4157, 1.3851, 1.3456,
                 1.4012, 1.3017, 1.2626, 1.1904, 1.2668, 1.3203, 1.3181, 1.1965,
                 1.1654, 1.2219, 1.3863, 1.306, 1.1963, 1.1376, 1.35, 1.2595,
                 1.1289, 1.0456, 1.2863, 1.1274, 1.0208, 1.0542, 1.1272, 1.1594,
                 1.1668, 1.1931, 1.1328, 1.1221, 1.1011, 1.1298, 1.0454, 1.0573,
                 1.0557, 1.0599, 0.973, 0.9545, 0.9721, 1.0489, 0.9934, 0.9382,
                 0.876, 1.339, 1.1939, 1.0229, 1.0378, 1.0402, 0.9554, 0.9794,
                 1.0089, 0.9422, 0.8584, 0.8563, 0.9013, 0.9252, 0.8706, 0.8354,
                 0.8071, 0.9737, 1.0008, 0.9593, 0.9257, 0.9556, 0.9534, 0.9313,
                 0.9151, 0.883, 0.8731, 0.8285, 0.8309, 0.9131, 0.9258, 0.8467,
                 0.7785), .Dim = c(4L, 4L, 6L))
w <- structure(c(18520.3, 11776.3, 19479.5, 22497.6, 18968.7, 17263.7,
                 36494.7, 21707, 13406.3, 13570.4, 37746.1, 20593.2, 6595.6, 25444.6,
                 59868.2, 81777.2, 3380.4, 20610.7, 22247.3, 6800.9, 5236.3, 14877.8,
                 7205, 5028.4, 1130.7, 6603.2, 4007.4, 2620.5, 374.8, 1624.3,
                 4963.7, 9551.3, 31806, 93615.9, 121986.6, 44640.3, 32110.6, 95814.4,
                 72827.9, 30922.5, 43197.3, 72050.8, 66673.4, 40370.1, 31488.2,
                 55014.9, 69457.2, 80021.2, 17701.7, 8765.2, 11790.9, 3872.8,
                 30544.5, 12141.2, 12415.2, 9471.9, 36138.6, 19198.1, 23120.1,
                 15597.9, 12140.2, 8058.3, 20948.3, 19380.2, 78543.9, 86503.6,
                 28727.8, 29208.7, 26300.6, 42363, 20786.6, 14380.3, 9493.5, 17816.2,
                 19844.1, 10898.2, 1419, 4211.5, 20615, 22748.2, 3365.8, 2639.8,
                 2433.3, 930.5, 22119.6, 31022.7, 12748.5, 10161.4, 15450.2, 32747.1,
                 22596.4, 13228.1, 17289.2, 30189.2, 31476.6, 15338.7),
                .Dim = c(4L, 4L, 6L))
m1 <- c(1.025527, 1.018229, 0.969744, 0.994998)
m2 <- c(1.111023, 1.030213, 0.935041, 0.906709)
m3 <- c(0.810568, 1.375203, 1.07096, 1.044461, 0.949441, 0.915284)
m12 <- structure(c(1.061059, 1.120345, 1.097519, 1.188501, 1.017091,
                   0.967245, 1.03447, 1.18867, 0.9797, 0.900885, 0.85575, 1.070772,
                   1.041953, 1.074653, 0.887316, 0.791906), .Dim = c(4L, 4L))
m13 <- structure(c(0.779029, 0.865343, 0.757887, 0.852708, 1.351367,
                   1.409585, 1.350907, 1.361528, 1.091867, 1.107661, 0.99364, 1.127478,
                   1.13439, 0.948428, 1.075919, 0.916096, 1.031958, 0.835103, 1.006321,
                   0.982888, 0.86109, 0.976673, 0.961731, 0.764211), .Dim = c(4L, 6L))
```

```
m23 <- structure(c(0.962955, 0.880973, 0.798545, 0.714783, 1.547556,
                   1.277098, 1.149491, 1.210108, 1.186342, 1.084436, 0.976822, 1.003611,
                   1.092564, 1.066306, 1.038601, 0.996779, 0.971751, 1.016173, 0.867197,
                   0.803929, 0.831913, 0.933863, 0.857392, 0.960169), .Dim = c(4L, 6L))

m <- list(m1, m3, m12, m23)
ind <- list(1, 3, c(1, 2), c(2, 3))


example <- WIPF(seed = s, weights = w, margins = m, indices = ind)
```

WIPF1 *Weighted Iterative Proportional Fitting (WIPF) in one dimension*

Description

Implements WIPF in one dimension. This function updates, using a set of weights, an initial 1-dimensional array, a vector (referred as the seed), to match a given value (referred as the margin), in such as way that the weighted sum of the updated values coincide with the margin.

Usage

```
WIPF1(
  seed,
  weights,
  margin = 1,
  normalize = TRUE,
  tol = 10^-6,
  maxit = 1000,
  full = FALSE,
  ...
)
```

Arguments

seed	A vector of non-negative values with the initial values.
weights	A vector of non-negative values with the weights associated to each component of seed and with the same length as seed.
margin	A non-negative scalar with the (weighted) marginal total to be fitted. Default, 1.
normalize	TRUE/FALSE argument indicating whether weights should be normalized to sum 1 before building the weighted sum to be compared with the margin value. Default, TRUE. Normalization is necessary when we want to adjust a set of indexes for which the margin correspond to other index that is a theoretical convex combination of the inner indexes. This characterizes a typical context where WIPF could be of value.

tol	Stopping criterion. The algorithm stops when the maximum absolute difference between the solutions obtained in two consecutive iteration is lower than the value specified by tol. Default, 0.000001.
maxit	Stopping criterion. A positive integer number indicating the maximum number of iterations allowed. Default, 1000. The algorithm will stop if the values to be fitted still has not converged after this many iterations.
full	TRUE/FALSE argument indicating if either only the solution should be shown or a more complete output.
...	Other arguments to be passed to the function. Not currently used.

Value

When full = FALSE an object similar to seed with the solution reached when the algorithm stops. When full = TRUE a list with the following components:

sol	An object similar to seed with the solution reached at convergence (or when the maximum number of iterations is reached).
iter	Number of iterations when the algorithm stops.
dev.margins	An object similar to margin with the absolute differences between the values in margin and the weighted sum of the values in sol.
inputs	A list containing all the objects with the values used as arguments by the function.

Note

Weighted Iterative proportional fitting is an extension of IPF. WIPF produces the same solutions than IPF with all weights being ones and when they are not normalized. IPF is also known as RAS in economics, raking in survey research or matrix scaling in computer science.

Author(s)

Jose M. Pavia, <pavia@uv.es>

Examples

```
s <- c(1.0595723, 0.9754876, 0.8589494, 0.8589123)
w <- c(651301.9, 581185.1, 555610.8, 602595.6)
example <- WIPF1(seed = s, weights = w)
```

WIPF2 *Weighted Iterative Proportional Fitting (WIPF) in two dimensions*

Description

Implements WIPF in two dimensions. This function updates an initial 2-dimensional array (a matrix, referred to as the seed) using a matrix of weights to align with a set of two vectors (referred to as the margins), where one of them can be missing. When `margin1` and `margin2` are compatible given the weights, the updated values ensure that the weighted sum across columns matches `margin1` and the weighted sum across rows matches `margin2`. If the margins are incompatible given the weights, the function `WIPF1` is applied to the initial margins to make them compatible. In those cases, margins are updated (are made compatible) in increasing order of sub-indices (i.e., `margin2` is adjusted to make it compatible with `margin1`).

Usage

```
WIPF2(
  seed,
  weights,
  margin1,
  margin2,
  normalize = TRUE,
  tol = 10^-6,
  maxit = 1000,
  full = FALSE,
  ...
)
```

Arguments

`seed`	A (RxC) matrix of non-negative values with the initial values.
`weights`	A (RxC) matrix of non-negative values with the weights associated to each entry of the matrix.
`margin1`	A R-length vector of positive values with the target (weighted) marginal sums across columns to be fitted.
`margin2`	A C-length vector of positive values with the target (weighted) marginal sums across rows to be fitted.
`normalize`	`TRUE`/`FALSE` argument indicating whether the weights should be normalized (across all dimensions, for either row or column weights to sum 1) before constructing the weighted sums for comparison with the margin values. Default, `TRUE`. Normalization is essential when adjusting a set of indexes where the margins represent theoretical convex combinations of the inner indexes. This characterizes a typical context where WIPF could be of value.
`tol`	Stopping criterion. The algorithm stops when the maximum difference between the weighted sum(s) of the values to be fitted and the margin(s) is lower than the value specified by `tol`. Default, `0.000001`.

maxit	Stopping criterion. A positive integer number indicating the maximum number of iterations allowed. Default, 1000. The algorithm will stop if the values to be fitted still has not converged after this many iterations.
full	TRUE/FALSE argument indicating if either only the solution should be shown or a more complete output.
...	Other arguments to be passed to the function. Not currently used.

Value

When full = FALSE an object similar to seed with the solution reached when the algorithm stops. When full = TRUE a list with the following components:

sol	An object similar to seed with the solution reached at convergence (or when the maximum number of iterations is reached).
iter	Number of iterations when the algorithm stops.
margin1	A R-length vector of positive values with the actual margin1 object used to reach the solution. This coincides with margin1 even when all the margins are not compatible given the weights.
margin2	A C-length vector of positive values with the actual margin2 object used to reach the solution. This coincides with margin2 when all the margins are compatible given the weights.
dev.margins	A list with a set of objects similar to the margins with absolute maximum deviations between the values in margins and the corresponding weighted sums of the values in sol.
dev.congruence	A list with a set of objects similar to the margins containing the differences between the final margins actually used (after adjusting them, if necessary, to be compatible with the weights) and the original margins provided by the user.
inputs	A list containing all the objects with the values used as arguments by the function.

Note

Weighted Iterative proportional fitting is an extension of IPF. WIPF produces the same solutions than IPF with all weights being ones and when they are not normalized. IPF is also known as RAS in economics, raking in survey research or matrix scaling in computer science.

Author(s)

Jose M. Pavia, <pavia@uv.es>

Examples

```
s <- structure(c(1.1279, 1.1304, 1.0304, 0.8554, 1.5606, 1.4171, 1.2862,
                 1.2472, 1.0746, 1.0796, 0.9806, 0.928, 1.1607, 1.2436, 1.2191,
                 1.0786, 1.0194, 1.1716, 0.9937, 0.8611, 1.0172, 1.2511, 1.1606,
                 1.1959), .Dim = c(4L, 6L))
w <- structure(c(72161.97, 93725.94, 84408.83, 172774.13, 52875.08,
                 31936.92, 14191.44, 12595.46, 291698.94, 231408.32,
```

```
                221763.43, 235217.74, 42028.56, 64458.09, 93443.13,
                60348.74, 222482.04, 103695.94, 57066.82, 48657.48,
                9572.75, 75745.02, 83912.38, 94019.92), .Dim = c(4L, 6L))
m1 <- c(1.110737, 1.029947, 0.934799, 0.906475)
m2 <- c(0.810992, 1.375921, 1.071519, 1.045006, 0.949938, 0.915762)
example <- WIPF2(seed = s, weights = w, margin1 = m1, margin2 = m2, full = TRUE)
```

WIPF3 *Weighted Iterative Proportional Fitting (WIPF) in three dimensions*

Description

Implements WIPF in three dimensions. This function updates an initial 3D-array (referred to as the seed) using a 3D-array of weights to align with a set of three vectors (referred to as 1D-margins) and three matrices (referred to as 2D-margins), where some of them can be missing. When all provided margins are compatible given the weights, the updated values ensure that the weighted sums across rows, columns, layers, and combinations of (row, column), (row, layer), and (column, layer) coincide with the provided margins. If the provided margins are incompatible given the weights, the functions `WIPF1` and `WIPF2` are applied to the initial margins to make the margins compatible with the weights. In those cases, the margins are updated (are made compatible) in increasing order of sub-indices and with the second sub-indices running faster.

Usage

```
WIPF3(
  seed,
  weights,
  margin1,
  margin2,
  margin3,
  margin12,
  margin13,
  margin23,
  normalize = TRUE,
  tol = 10^-6,
  maxit = 1000,
  full = FALSE,
  ...
)
```

Arguments

`seed`	A (RxCxL) array of non-negative values with the initial values.
`weights`	A (RxCxL) array of non-negative values with the weights associated to each entry of the `seed` array.

margin1	A R-length vector of positive values with the target (weighted) marginal sum across both layers and columns to be fitted.
margin2	A C-length vector of positive values with the target (weighted) marginal sum across both rows and layers to be fitted.
margin3	A L-length vector of positive values with the target (weighted) marginal sum across both rows and columns to be fitted.
margin12	A RxC matrix of positive values with the target (weighted) marginal sum across layers to be fitted.
margin13	A RxL matrix of positive values with the target (weighted) marginal sum across columns to be fitted.
margin23	A CxL matrix of positive values with the target (weighted) marginal sum across both rows to be fitted.
normalize	TRUE/FALSE argument indicating whether the weights should be normalized across all dimensions (for either row, column, layer, row-column, row-layer or column-layer weights to sum 1) before constructing the weighted sums for comparison with the margin values. Default, TRUE. Normalization is essential when adjusting a set of indexes where the margins represent theoretical convex combinations of the inner indexes. This characterizes a typical context where WIPF could be of value.
tol	Stopping criterion. The algorithm stops when the maximum difference between the weighted sum(s) of the values to be fitted and the margin(s) is lower than the value specified by tol. Default, 0.000001.
maxit	Stopping criterion. A positive integer number indicating the maximum number of iterations allowed. Default, 1000. The algorithm will stop if the values to be fitted still has not converged after this many iterations.
full	TRUE/FALSE argument indicating if either only the solution should be shown or a more complete output.
...	Other arguments to be passed to the function. Not currently used.

Value

When full = FALSE an object similar to seed with the solution reached when the algorithm stops. When full = TRUE a list with the following components:

sol	An object similar to seed with the solution reached at convergence (or when the maximum number of iterations is reached).
iter	Number of iterations when the algorithm stops.
margin1	A R-length vector of positive values with the actual margin1 object used to reach the solution. This coincides with margin1 even when all the margins are not compatible given the weights.
margin2	A C-length vector of positive values with the actual margin2 object used to reach the solution. This coincides with margin2 when all the margins are compatible given the weights.
margin3	A L-length vector of positive values with the actual margin3 object used to reach the solution. This coincides with margin3 when all the margins are compatible given the weights.

margin12	A RxC matrix of positive values with the actual margin12 object used to reach the solution. This coincides with margin12 when all the margins are compatible given the weights.
margin13	A RxL matrix of positive values with the actual margin13 object used to reach the solution. This coincides with margin13 when all the margins are compatible given the weights.
margin23	A CxL matrix of positive values with the actual margin23 object used to reach the solution. This coincides with margin23 when all the margins are compatible given the weights.
dev.margins	A list with a set of objects similar to the margins with absolute maximum deviations between the values in margins and the corresponding weighted sums of the values in sol.
dev.congruence	A list with a set of objects similar to the margins containing the differences between the final margins actually used (after adjusting them, if necessary, to be compatible with the weights) and the original margins provided by the user.
inputs	A list containing all the objects with the values used as arguments by the function.

Note

Weighted Iterative proportional fitting is an extension of IPF. WIPF produces the same solutions than IPF with all weights being ones and when they are not normalized. IPF is also known as RAS in economics, raking in survey research or matrix scaling in computer science.

Author(s)

Jose M. Pavia, <pavia@uv.es>

Examples

```
s <- structure(c(0.9297, 0.9446, 0.8763, 0.92, 0.8655, 0.8583, 0.8132,
                 0.8679, 0.7968, 0.7834, 0.721, 0.7859, 0.7747, 0.7851, 0.8632,
                 1.041, 1.5617, 1.5642, 1.4847, 1.5176, 1.4157, 1.3851, 1.3456,
                 1.4012, 1.3017, 1.2626, 1.1904, 1.2668, 1.3203, 1.3181, 1.1965,
                 1.1654, 1.2219, 1.3863, 1.306, 1.1963, 1.1376, 1.35, 1.2595,
                 1.1289, 1.0456, 1.2863, 1.1274, 1.0208, 1.0542, 1.1272, 1.1594,
                 1.1668, 1.1931, 1.1328, 1.1221, 1.1011, 1.1298, 1.0454, 1.0573,
                 1.0557, 1.0599, 0.973, 0.9545, 0.9721, 1.0489, 0.9934, 0.9382,
                 0.876, 1.339, 1.1939, 1.0229, 1.0378, 1.0402, 0.9554, 0.9794,
                 1.0089, 0.9422, 0.8584, 0.8563, 0.9013, 0.9252, 0.8706, 0.8354,
                 0.8071, 0.9737, 1.0008, 0.9593, 0.9257, 0.9556, 0.9534, 0.9313,
                 0.9151, 0.883, 0.8731, 0.8285, 0.8309, 0.9131, 0.9258, 0.8467,
                 0.7785), .Dim = c(4L, 4L, 6L))
w <- structure(c(18520.3, 11776.3, 19479.5, 22497.6, 18968.7, 17263.7,
                 36494.7, 21707, 13406.3, 13570.4, 37746.1, 20593.2, 6595.6, 25444.6,
                 59868.2, 81777.2, 3380.4, 20610.7, 22247.3, 6800.9, 5236.3, 14877.8,
                 7205, 5028.4, 1130.7, 6603.2, 4007.4, 2620.5, 374.8, 1624.3,
                 4963.7, 9551.3, 31806, 93615.9, 121986.6, 44640.3, 32110.6, 95814.4,
                 72827.9, 30922.5, 43197.3, 72050.8, 66673.4, 40370.1, 31488.2,
```

```
                  55014.9, 69457.2, 80021.2, 17701.7, 8765.2, 11790.9, 3872.8,
                  30544.5, 12141.2, 12415.2, 9471.9, 36138.6, 19198.1, 23120.1,
                  15597.9, 12140.2, 8058.3, 20948.3, 19380.2, 78543.9, 86503.6,
                  28727.8, 29208.7, 26300.6, 42363, 20786.6, 14380.3, 9493.5, 17816.2,
                  19844.1, 10898.2, 1419, 4211.5, 20615, 22748.2, 3365.8, 2639.8,
                  2433.3, 930.5, 22119.6, 31022.7, 12748.5, 10161.4, 15450.2, 32747.1,
                  22596.4, 13228.1, 17289.2, 30189.2, 31476.6, 15338.7),
                 .Dim = c(4L, 4L, 6L))
m1 <- c(1.025527, 1.018229, 0.969744, 0.994998)
m2 <- c(1.111023, 1.030213, 0.935041, 0.906709)
m3 <- c(0.810568, 1.375203, 1.07096, 1.044461, 0.949441, 0.915284)
m12 <- structure(c(1.061059, 1.120345, 1.097519, 1.188501, 1.017091,
                   0.967245, 1.03447, 1.18867, 0.9797, 0.900885, 0.85575, 1.070772,
                   1.041953, 1.074653, 0.887316, 0.791906), .Dim = c(4L, 4L))
m13 <- structure(c(0.779029, 0.865343, 0.757887, 0.852708, 1.351367,
                   1.409585, 1.350907, 1.361528, 1.091867, 1.107661, 0.99364, 1.127478,
                   1.13439, 0.948428, 1.075919, 0.916096, 1.031958, 0.835103, 1.006321,
                   0.982888, 0.86109, 0.976673, 0.961731, 0.764211), .Dim = c(4L, 6L))
m23 <- structure(c(0.962955, 0.880973, 0.798545, 0.714783, 1.547556,
                   1.277098, 1.149491, 1.210108, 1.186342, 1.084436, 0.976822, 1.003611,
                   1.092564, 1.066306, 1.038601, 0.996779, 0.971751, 1.016173, 0.867197,
                   0.803929, 0.831913, 0.933863, 0.857392, 0.960169), .Dim = c(4L, 6L))

example <- WIPF3(seed = s, weights = w, margin3 = m3, margin12 = m12, margin13 = m13)
```

Index