Package: WIPF (via r-universe)

January 7, 2025

Type Package

Title Weighted Iterative Proportional Fitting

Version 0.1.0-1

Description Implementation of the weighted iterative proportional fitting (WIPF) procedure for updating/adjusting a N-dimensional array (currently N<=3) given a weight structure and some target marginals. Acknowledgements: The author wish to thank 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] (<https://orcid.org/0000-0002-0129-726X>)

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

Repository CRAN

Date/Publication 2025-01-07 16:30:05 UTC

Contents

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 seedand 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 if 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.
`error.margins`	An object similar to `margin` with the absolute differences between the values in `margin` and the weighted sum(s) 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 the margins compatible with the weights. In those cases, margins are updated (are made compatible) in increasing order of sub-indices (i.e., `margin2` is adjusted to be 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	Logical (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.

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.
margin1	A R-length vector of positive values with the actual margin1 object used to reach the solution. This coincides with margin1 when all the margins are 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.
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	Logical (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.

`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`.

`margin1` A R-length vector of positive values with the actual margin1 object used to reach the solution. This coincides with `margin1` when all the margins are 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 `margin2` 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 `margin2` 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 `margin2` 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 `margin2` when all the margins are compatible given the weights.

`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