Package 'eiCircles'

April 11, 2025

Type Package

Title Ecological Inference of RxC Tables by Overdispersed-Multinomial Models

Version 0.0.1-12

Description Estimates RxC (R by C) vote transfer matrices (ecological contingency tables) from aggregate data using the model described in Forcina et al. (2012), as extension of the model proposed in Brown and Payne (1986). Allows incorporation of covariates.
References:
Brown, P. and Payne, C. (1986). "Aggregate data, ecological regression and voting transitions". Journal of the American Statistical Association, 81, 453–460. <DOI:10.1080/01621459.1986.10478290>.
Forcina, A., Gnaldi, M. and Bracalente, B. (2012). "A revised Brown and Payne model of voting behaviour applied to the 2009 elections in Italy". Statistical Methods & Applications, 21, 109–119. <DOI:10.1007/s10260-011-0184-x>.

License GPL (>= 2)

Encoding UTF-8

Imports stats, NlcOptim (>= 0.6)

Suggests ggplot2, scales

RoxygenNote 7.3.2

NeedsCompilation no

Author Antonio Forcina [aut] (<https://orcid.org/0000-0001-5239-5495>), 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 2025-04-11 11:20:01 UTC

Contents

BPF *Ecological Inference of RxC Tables by Overdispersed-Multinomial Models*

Description

Implements the model proposed in Forcina et al. (2012), as extension of Brown and Payne (1986), to estimate RxC vote transfer matrices (ecological contingency tables). Allows incorporation of covariates.

Usage

```
BPF(
  X,
  Y,
  local = "IPF",
  covariates = NULL,
  census.changes = "adjust1",
  stable.units = TRUE,
  stability.par = 0.12,
  confidence = 0.95,
  cs = 50,
  null.cells = NULL,
  row.cells.relationships = NULL,
  row.cells.relationships.C = NULL,
  pair.cells.relationships = NULL,
  cells.fixed.logit = NULL,
  dispersion.rows = data.frame(row1 = rep(1L, ncol(X) - 1L), row2 = 2:ncol(X)),
  start.values = NULL,
  seed = NULL,
  max.iter = 100,
  max.iter.hyper = 1000,
  tol = 1e-04,
  verbose = FALSE,
  save.beta = FALSE,
  ...
)
```

Arguments

X	matrix (or data.frame) of order KxR with either the electoral results recorded in election 1 or the sum across columns (the margins of row options) of the K ecological tables.
Y	matrix (or data.frame) of order KxC with either the electoral results recorded in election 2 or the sum across rows (the margins of column options) of the K ecological tables.
local	A character string indicating the algorithm to be used for adjusting the estimates of the transition probabilities obtained for the whole area (electoral space) with the actual observations available in each local unit. Only "IPF" (iterative proportional fitting, also known as raking), "lik" (an algorithm based on the assumed likelihood), "hyper" (an algorithm based on assuming a multihypergeometric distribution for the inner values of the unit table given the observed row and column margins, which should be integers; even after census adjustments, if this is necessary) and "none" are allowed. When local = "none", no local estimates are obtained. Default, "IPF"
covariates	A list with two components, covar and meta. covar is a matrix (or data.frame), of order KxNC (where K is the number of (polling) units and NC the number of covariates), with the values of the covariate(s) in each unit. meta is a matrix (or data.frame) with three columns. The data in these columns inform about the cell(s) (row and column) and covariate(s) that should be employed for modelling probabilities in each cell. Cell(s) and covariate(s) could be identified by position or names. For instance, (2, 3, "income") means that the covariate identified as "income" in the object covar should be used as covariate to model the probability corresponding to cell (2, 3) of the transfer (transition probability) matrix. Equally, ("party1", "party2", 4) means that the covariate located in the fourth column of meta should be used to model the transfer probability from "party1" to "party2", where "party1" (in X) and "party2" (in Y) are names used to identified columns in the election data objects. Default, NULL: no covariates are used.
census.changes	A string character indicating how census changes between elections must be handled. At the moment, it only admits two values "adjust1" and "adjust2", where the distributions of votes in election 1 or 2 are, respectively, adjusted to match the outcomes of the other election: "adjust1" adjusts the census of the first election to match that of the second one; "adjust2" adjusts the census of the second election to match that of the first one. Default, "adjust1".
stable.units	A TRUE/FALSE character indicating whether only stable units (those whose number of total number of voters have experienced a small change) are selected. Default, TRUE.
stability.par	A non-negative number that controls the maximum proportion of relative change in the total census for a unit to be considered stable. Default, 0.12. The relative change is measured as the absolute value of the difference of the logarithms of the sizes (censuses) in the two elections. Measuring the relative change this way avoids dependence on which election is used as reference.
confidence	A number between 0 and 1 to be used as level of confidence for the confidence intervals of the transition probabilities (TP estimates). Default, 0.95.

`cs` A positive number indicating the average number of cluster size. Default, 50.

`null.cells` A matrix (or data.frame) with two columns (row, column) informing about the cells whose probabilities should be constrained to be zero. Cells could be identified by position or names. For instance, (2, 3) means that the probability corresponding to cell (2, 3) of the transfer matrix should be constrained to be zero. Equally, ("party1", "party2") means that the transfer probability from "party1" (in `X`) to "party2" (in `Y`) will be zero, where "party1" and "party2" are names used to identified columns in the election data objects. Because the model takes the last option of `Y` as reference, constraints of this kind cannot be defined involving a cell of the reference category. See `Note` and `Details` for more information about constraints and how properly define them. Default, `NULL`: no null constraints.

`row.cells.relationships`
A matrix (or data.frame) with four columns (row, column1, column2, constant) may be used to assign a pre-specified value to the ratio between the transition probabilities of two cells within the same row. Because the model takes the value in column2 as reference to define this constraint, column1 and column2 must be different from the last column which has already been used to define the logits. Rows and columns could be identified by position or names. For instance, (2, 3, 5, 0.5) means that the probability corresponding to cell (2, 3) of the transfer matrix is constrained to be equal to 0.5 times the probability corresponding to cell (2, 5) of the transfer matrix. Because each cell defined by (row, column2) is used as reference relative to the corresponding cell (row, column1), it is removed and thus that cell cannot be reference within two different constraints. So, constraints involving the same cell should be defined with care. To be specific, the cells defined by (row, columns2) should not appear in other constraints. For instance, if in the i-th row you want constrain (cell 3) = (cell 1) x 0.6 and (cell 3) = (cell 2) x 0.3 you need to specify it as (cell 3) = (cell 1) x 0.6 and as (cell 2) = (cell 1) x 2. See `Note` and `Details` for more information about constraints and how properly define them.. Default, `NULL`: no row-cell constraints.

`row.cells.relationships.C`
A matrix (or data.frame) with three columns (row, column, constant) informing about the analog to the constraints described in `row.cells.relationships` when 'column2' refers to the reference category (C-th column in `Y`). This is needed because logits are already computed with reference to column C, constraining these ratios is equivalent to assign a specified value to the logit in the corresponding cell. Rows and columns could be identified by position or names. For instance, (2, 3, 0.5) means that the probability corresponding to cell (2, 3) of the transfer matrix is constrained to be equal to 0.5 times the probability corresponding to cell (2, `ncol(Y)`) of the transfer matrix. See `Note` and `Details` for more information about constraints and how properly define them. Default, `NULL`: no row-proportional constraints.

`pair.cells.relationships`
This is a kind of less stringent version of the argument `row.cells.relationships`. Both may be used to increase or decrease a transition which is expected to be too different from informed expectations. This argument is declared via a matrix (or data.frame) with seven columns (row1, column1.1, column1.2, row2,

column2.1, column2.2, constant) which imposes proportional relationships between ratios of probabilities corresponding to row1 and and row2. Let r1 be the ratio between the probabilities in columns 1.1 and 1.2 in row 1, 'r1 = cell(row1, column1.1)/cell(row1, column1.2)', and r2 the equivalent ration between probabilities in columns 2.1 and 2.2 in row2, 'r2 = cell(row2, column2.1)/cell(row2, column2.2)', then this argument is used to assign the specified value 'constant' to 'r2/r1'. Rows and columns could be identified by position or names. For instance, (2, 3, 5, 3, 4, 2, 0.5) means that the ratio of probabilities corresponding to cells (2, 3) and (2, 5) of the transfer matrix is constrained to be equal to 0.5 times the ratio of probabilities corresponding to cells (3, 4) and (3, 2) of the transfer matrix. See `Note` and `Details` for more information about constraints and how properly define them. Default, `NULL`: no ratio-proportional constraints.

`cells.fixed.logit`

A matrix (or data.frame) with three columns (row, column, number) informing about the cells with fixed values for the logit of the probability corresponding to the cell; this does not set the actual transition but its ratio with respect to the reference category. For instance, (2, 3, -5) means that the logit of the probability corresponding to cell (2, 3) of the transfer matrix is constrained to be -5. See `Note` and `Details` for more information about constraints and how properly define them. Default, `NULL`: no logit constraints.

`dispersion.rows`

A matrix (or data.frame) with two columns (row1, row2) indicating what pair of two rows should have equal overdispersions. Default, over-dispersions are assumed to be the same in all rows: `data.frame("row1" = rep(1L, ncol(X) - 1L), "row2" = 2:ncol(X))`. See `Note` and `Details` for more information about constraints and how properly define them. Use `dispersion.rows = NULL` to specify that overdispersion is unconstrained, i.e., that each row has a different parameter.

`start.values` A vector of length `ncol(X)*ncol(Y) + nrow(meta) - NR`, where `nrow(meta)` accounts for the number of regression coefficients and `NR` is the number of restrictions imposed to either cell probabilities of the transition matrix or overdispersions through the arguments `cells.fixed.logit`, `row.cells.relationships`, `null.cells`, `row.cells.relationships.C`, `pair.cells.relationships` and `dispersion.rows`, with the initial estimates for (i) the logits of the transition matrix probabilities, taking the last column of `Y` as reference, (ii) the overdispersions (in the logit scale) and (iii) the coefficients in the regression models defined via `covariates`. Typically, this is a beta vector obtained from a previous run of `BPF` with the same specified model, but which abruptly stopped because of a break in the converging process (see the `save.beta` argument). Default, `NULL`. When `start = NULL` random initial values for the transition probabilities are generated assuming independence between origin and destination options (i.e., implying that transition probabilities are constant across rows), sound values for the over-dispersion parameters are generated and zero coefficients are assumed for the predictors of the regression models.

`seed` A number indicating the random seed to be used. Default, `NULL`: no seed is used.

`max.iter` Integer positive number. Maximum number of iterations to be performed for the Fisher scoring algorithm during the MLE estimation. Default, 100.

max.iter.hyper	Integer positive number. Maximum number of iterations without change to be performed for search of the MLE estimate in each unit table when local = "hyper". Default, 1000.
tol	Maximum value allowed for the numerical estimates of the partial derivatives of the likelihood in the point of convergence. Default, 0.0001.
verbose	A TRUE/FALSE character indicating whether intermediate results should be printed in the screen during the convergence process. Default FALSE.
save.beta	A TRUE/FALSE character indicating whether, while convergence is performed, the vector of temporary logits, over-dispersion (in logit scale) parameters and (if required) regression coefficients should be saved in the working directory in the file "beta.Rdata" file. This data could be used to restart the process in case of a premature failure of convergence process. Default FALSE.
...	Other arguments to be passed to the function. Not currently used.

Details

Description about how **defining constraints** in more detail.

To define constraints properly is a little tricky. Clearly, in the first place, it is the responsibility of the user to define constraints that are mutually compatible among themselves. The function does not check them to be jointly congruent. It is important to be aware that each linear constraint, when implemented, requires an element of the vector of internal parameters to be set to a known value and the corresponding element of the (underlying) design matrix to be removed. In addition, certain constraints are implemented by replacing one or more columns of the design matrix by suitable linear combinations of the columns that correspond to the cells involved in the constraint. A warning will be issued when two or more constraints require to remove the same column of the design matrix. To avoid conflicting constraints, a safe rule is that each constraint should be acting on disjoint sets of cells.

For each type of constraint, below we specify which column of the design matrix is removed and when a linear combination is needed how it is defined. Note that, in the unconstrained model, the design matrix has a column for each cell of the transition probabilities listed by row except for the last column which is used as reference:

- null.cells: The column of the design matrix corresponding to the cell defined by 'row' and column' declared when defining the constraint is removed.
- row.cells.relationships: The column of the design matrix corresponding to the cell (row, column2) is removed while the one corresponding to the cell (row, column2) is adjusted.
- row.cells.relationships.C: The column of the design matrix corresponding to the cell determined by each pair 'row', 'column' is removed.
- pair.cells.relationships: This constraint is defined by 4 pairs of "row, column"; the column of the design matrix corresponding to the last pair (row2, column2.2) will be removed and the others adjusted.

Value

A list with the following components

`TM` The estimated RxC table (matrix) of transition probabilities/rates. This coincides with `TP` when `local = "none"` and is equal to `TR` when `local = "IPF"`, `local = "hyper"` or `local = "lik"`.

`TM.votes` The estimated RxC table (matrix) of votes corresponding to `TM`.

`TP` The estimated RxC table (matrix) of underlying transition probabilities obtained after applying the approach in Forcina et al. (2012) with the specified model.

`TP.units.cov` With covariates an array of order RxCxK with the estimates tables/matrices of transition probabilities corresponding to each unit taking into account the values of the covariates in the unit. Without covariates this object is `NULL`.

`TR` When `local = "IPF"`, `local = "hyper"` or `local = "lik"`, the estimated RxC table/matrix of transition rates obtained as composition of the estimated unit tables/matrices attained after adjusting `TP` in each polling unit to the unit margins using the iterative proportional fitting algorithm. When `local = "none"`, this object is `NULL`.

`TR.units` When `local = "IPF"`, `local = "hyper"` or `local = "lik"`, an array of order RxCxK with the tables/matrices of transition rates attained in each unit attained after adjusting `TP` using the iterative proportional fitting algorithm to the unit margins. When `local = "none"`, this object is `NULL`.

`TR.votes.units` When `local = "IPF"`, `local = "hyper"` or `local = "lik"`, the array of order RxCxK with the tables/matrices of votes linked to the `TR.units` array. When `local = "none"`, this object is `NULL`.

`TP.lower` A matrix of order RxC with the estimated lower limits of the confidence intervals, based on a normal approximation, of the underlying transition probabilities (`TP`) of the row-standardized vote transitions from election 1 to election 2.

`TP.upper` A matrix of order RxC with the estimated upper limits of the confidence intervals, based on a normal approximation, of the underlying transition probabilities (`TP`) of the row-standardized vote transitions from election 1 to election 2.

`beta` The estimated vector of internal parameters (logits) at convergence. The first `R(C-1) - NR` elements (where `NR` is the number of restrictions imposed in cell probabilities) are logits of transitions and the last `nrow(meta)` elements are the regression coefficients in case covariates are present. The over dispersion(s) parameter(s) is (are) in between. Default, just one over-dispersion parameter. In case of non-convergence, if the function is used with `save.beta = TRUE`, the components of beta from the file "beta.Rdata" may be used to restart the algorithm from where it stopped by introducing them via the `start.values` argument.

`overdispersion` The estimated vector at convergence of internal overdispersion parameters in the scale from 0 to 1.

`sd.TP` Estimated standard deviations of the estimated transition probabilities.

`sd.beta` The estimated standard errors of the elements of beta.

`cov.beta` The estimated covariance matrix of beta. It may be used to compute approximate variances of transformations of the beta parameters, such as transition probabilities.

madis	A vector of length K with discrepancies of individual local units based on the Mahalanobis measure. It is essentially the quadratic discrepancy between observed and estimated votes weighted by the inverse of the estimated variance.
lk	The value of the log-likelihood at convergence.
selected.units	A vector with the indexes corresponding to the units finally selected to estimate the vote transition probability matrix.
iter	An integer number indicating the number of iterations performed before converging or when stopped.
X	Matrix of order KxR with the adjusted electoral results recorded in election 1.
Y	Matrix of order KxC with the adjusted electoral results recorded in election 2.
inputs	A list containing all the objects with the values used as arguments by the function.

Note

Constraints may be used to force estimates to take values different from those obtained by unconstrained estimation. As such, these tools should be used sparingly and, essentially, to assess whether estimates are substantially (significantly) different from what we would expect or unexpected estimates are only due to random variation. To first order approximation, twice the difference between the unconstrained and the constrained log-likelihood should be distributed as a chi-square with 1 degree of freedom. This allows to test which constraints are in substantial conflict with the data.

Author(s)

Antonio Forcina, <forcinarosara@gmail.com>

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

References

Brown, P. and Payne, C. (1986). Aggregate data, ecological regression and voting transitions. *Journal of the American Statistical Association*, 81, 453–460. doi:10.1080/01621459.1986.10478290

Forcina, A., Gnaldi, M. and Bracalente, B. (2012). A revised Brown and Payne model of voting behaviour applied to the 2009 elections in Italy. *Statistical Methods & Applications*, 21, 109–119. doi:10.1007/s102600110184x

Examples

```
votes1 <- structure(list(P1 = c(16L, 4L, 13L, 6L, 1L, 16L, 6L, 17L, 48L, 14L),
                         P2 = c(8L, 3L, 0L, 5L, 1L, 4L, 7L, 6L, 28L, 8L),
                         P3 = c(38L, 11L, 11L, 3L, 13L, 39L, 14L, 34L, 280L, 84L),
                         P4 = c(66L, 5L, 18L, 39L, 30L, 57L, 35L, 65L, 180L, 78L),
                         P5 = c(14L, 0L, 5L, 2L, 4L, 21L, 6L, 11L, 54L, 9L),
                         P6 = c(8L, 2L, 5L, 3L, 0L, 7L, 7L, 11L, 45L, 17L),
                         P7 = c(7L, 3L, 5L, 2L, 3L, 17L, 7L, 13L, 40L, 8L)),
                         row.names = c(NA, 10L), class = "data.frame")
votes2 <- structure(list(C1 = c(2L, 1L, 2L, 2L, 0L, 4L, 0L, 4L, 19L, 14L),
                         C2 = c(7L, 3L, 1L, 7L, 2L, 5L, 3L, 10L, 21L, 6L),
                         C3 = c(78L, 7L, 28L, 42L, 28L, 84L, 49L, 85L, 260L, 100L),
```

```
                          C4 = c(56L, 14L, 20L, 7L, 19L, 54L, 22L, 50L, 330L, 91L),
                          C5 = c(14L, 3L, 6L, 2L, 3L, 14L, 8L, 8L, 45L, 7L)),
                          row.names = c(NA, 10L), class = "data.frame")
example <- BPF(votes1, votes2, local = "IPF")$TM
```

---

plot.BPF *Graphical representation of a RxC ecological inference (vote transfer) matrix*

Description

Plot method for objects obtained with BPF.

Usage

```
## S3 method for class 'BPF'
plot(
  x,
  margins = TRUE,
  digits = 2,
  row.names = NULL,
  col.names = NULL,
  size.numbers = 6,
  size.labels = 4,
  size.margins = 6,
  colour.cells = "darkolivegreen3",
  colour.grid = "floralwhite",
  alpha = 0.5,
  which = NULL,
  ...,
  show.plot = TRUE
)
```

Arguments

x	An object output of the **BPF** function.
margins	A TRUE/FALSE argument informing whether the margins of the matrix should be displayed. Default, TRUE.
digits	Integer indicating the number of decimal places to be shown. Default, 2.
row.names	Names to be used for the rows of the matrix.
col.names	Names to be used for the columns of the matrix.
size.numbers	A reference number indicating the average font size to be used for the transfer numbers. Default, 6.
size.labels	A number indicating the font size to be used for labels. Default, 4.
size.margins	A number indicating the font size to be used for margin numbers. Default, 6.

colour.cells	Background base colour for cells.
colour.grid	Colour to be used for grid lines.
alpha	A [0,1] number of colour transparency.
which	A vector of integers informing the units for which the aggregate transfer matrix should be plotted. Default, NULL: the global matrix is shown.
...	Other arguments passed on to methods. Not currently used.
show.plot	A TRUE/FALSE value indicating whether the plot should be displayed as a side-effect. By default, TRUE.

Value

Invisibly returns the (ggplot) description of the plot, which is a list with components that contain the plot itself, the data, information about the scales, panels etc.

Note

ggplot2 is needed to be installed for this function to work.

Author(s)

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

Examples

```
votes1 <- structure(list(P1 = c(16L, 4L, 13L, 6L, 1L, 16L, 6L, 17L, 48L, 14L),
                         P2 = c(8L, 3L, 0L, 5L, 1L, 4L, 7L, 6L, 28L, 8L),
                         P3 = c(38L, 11L, 11L, 3L, 13L, 39L, 14L, 34L, 280L, 84L),
                         P4 = c(66L, 5L, 18L, 39L, 30L, 57L, 35L, 65L, 180L, 78L),
                         P5 = c(14L, 0L, 5L, 2L, 4L, 21L, 6L, 11L, 54L, 9L),
                         P6 = c(8L, 2L, 5L, 3L, 0L, 7L, 7L, 11L, 45L, 17L),
                         P7 = c(7L, 3L, 5L, 2L, 3L, 17L, 7L, 13L, 40L, 8L)),
                         row.names = c(NA, 10L), class = "data.frame")
votes2 <- structure(list(C1 = c(2L, 1L, 2L, 2L, 0L, 4L, 0L, 4L, 19L, 14L),
                         C2 = c(7L, 3L, 1L, 7L, 2L, 5L, 3L, 10L, 21L, 6L),
                         C3 = c(78L, 7L, 28L, 42L, 28L, 84L, 49L, 85L, 260L, 100L),
                         C4 = c(56L, 14L, 20L, 7L, 19L, 54L, 22L, 50L, 330L, 91L),
                         C5 = c(14L, 3L, 6L, 2L, 3L, 14L, 8L, 8L, 45L, 7L)),
                         row.names = c(NA, 10L), class = "data.frame")
example <- BPF(votes1, votes2)
p <- plot(example, show.plot = FALSE)
p
```

print.BPF *Print a summary of an output of the BPF function*

Description

Print method for objects obtained with the BPF function.

Usage

```
## S3 method for class 'BPF'
print(x, ..., margins = TRUE, digits = 2)
```

Arguments

x	An object output of the **BPF** function.
...	Other arguments passed on to methods. Not currently used.
margins	A TRUE/FALSE argument informing if the margins of the transition matrix should be displayed. Default, TRUE.
digits	Integer indicating the number of decimal places to be shown. Default, 2.

Value

No return value, called for side effects.

Author(s)

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

Examples

```
votes1 <- structure(list(P1 = c(16L, 4L, 13L, 6L, 1L, 16L, 6L, 17L, 48L, 14L),
                         P2 = c(8L, 3L, 0L, 5L, 1L, 4L, 7L, 6L, 28L, 8L),
                         P3 = c(38L, 11L, 11L, 3L, 13L, 39L, 14L, 34L, 280L, 84L),
                         P4 = c(66L, 5L, 18L, 39L, 30L, 57L, 35L, 65L, 180L, 78L),
                         P5 = c(14L, 0L, 5L, 2L, 4L, 21L, 6L, 11L, 54L, 9L),
                         P6 = c(8L, 2L, 5L, 3L, 0L, 7L, 7L, 11L, 45L, 17L),
                         P7 = c(7L, 3L, 5L, 2L, 3L, 17L, 7L, 13L, 40L, 8L)),
                         row.names = c(NA, 10L), class = "data.frame")
votes2 <- structure(list(C1 = c(2L, 1L, 2L, 2L, 0L, 4L, 0L, 4L, 19L, 14L),
                         C2 = c(7L, 3L, 1L, 7L, 2L, 5L, 3L, 10L, 21L, 6L),
                         C3 = c(78L, 7L, 28L, 42L, 28L, 84L, 49L, 85L, 260L, 100L),
                         C4 = c(56L, 14L, 20L, 7L, 19L, 54L, 22L, 50L, 330L, 91L),
                         C5 = c(14L, 3L, 6L, 2L, 3L, 14L, 8L, 8L, 45L, 7L)),
                         row.names = c(NA, 10L), class = "data.frame")
example <- BPF(votes1, votes2, local = "none")
print(example, digits = 1, margins = TRUE)
```

print.summary.BPF *Print a summary of a summary.BPF object*

Description

Print method for summary.BPFC objects

Usage

```
## S3 method for class 'summary.BPF'
print(x, ..., margins = TRUE, digits = 2)
```

Arguments

x	An summary.BPF class object.
...	Other arguments passed on to methods. Not currently used.
margins	A TRUE/FALSE argument informing if the margins of the transition matrix should be displayed. Default, TRUE.
digits	Integer indicating the number of decimal places to be shown. Default, 2.

Value

No return value, called for side effects.

simula_BPF *Simulate RxC Tables from Overdispersed-Multinomial Models*

Description

Generates at random a set of RxC tables with the joint distribution of voters in two elections according to the model proposed in Forcina et al. (2012), as extension of Brown and Payne (1986), under the assumption that local units are homogeneous (no covariates). Results in the first election may be provided by the user or generated at random according to the overdispersed multinomial model.

Usage

```
simula_BPF(
  n.units,
  TM,
  prop1,
  polling.sizes,
  theta1 = 0.1,
  theta2 = 0.1,
  cs = 50,
  noise = 0,
```

```
  simplify = FALSE,
  ...
)
```

Arguments

n.units	Either a positive integer number, K, indicating the number of polling units to be simulated, or a KxR data.frame of a matrix with the number of votes gained in election 1 for each of the R options in each of the K units. If n.units is a matrix (data.frame) of counts (votes) the values of arguments prop1 and theta1 are ommitted.
TM	A row-standardized RxC matrix with the underlying global transition probabilities of the simulated elections. If the matrix is not row-standardized, it is internally row-standardized by the function.
prop1	A vector of length R with the initial assumed probabilities of voting (to be simulated) for each of the R competing options in the first election. If the provided vector is not a set of probabilities (i.e., a vector of positive numbers adding to 1), it is internally standardized by the function.
polling.sizes	Either a vector of two components with two positive integer numbers indicating the minimum and maximum number of voters for each unit or a vector of length n.units of positive integer numbers informing about the number of voters in each unit. When polling.sizes is a vector of length two, a number of voters is randomly assigned for each unit using a uniform distribution with parameters the minimum and maximum values included in polling.sizes.
theta1	A number between 0 and 1 used as the overdispersion parameter. This parameter is employed by the underlying Dirichlet distribution, in conjunction with prop1, to randomly generate vectors of probabilities for each unit. These vectors are then used to simulate the results of the first election. The smaller the value of this parameter, the closer the unit-level marginal distributions for the first election are to prop1. Default, 0.1.
theta2	Either a single number between 0 and 1 or a vector of length nrow(TM) containing numbers between 0 and 1. The values in theta2 serve as overdispersion parameters and are used alongside the row-probability vectors in TM within the underlying Dirichlet distributions. These distributions are employed to generate probability vectors for each combination of unit, cluster, and row, which are then used to simulate vote transfers from the first to the second election. If theta2 is a vector, each row is assigned a distinct overdispersion parameter based on its corresponding value. Default, 0.1.
cs	A positive number indicating the average number of cluster size. Default, 50.
noise	Either a single number between 0 and 1 or a vector of length nrow(TM) containing numbers between 0 and 1. These numbers account for the proportion of causal voters of each origin party (row). These numbers are used to introduce more variability, compared to the BPF model, into the simulations. If noise > 0, a 100*noise percentage of votes of each row of each unit are randomly assigned among the column parties. Default, 0.

simplify	A TRUE/FALSE argument indicating whether the simulated RxCxK array of counts by polling unit should be rearranged as a matrix of order Kx(RC). Default, FALSE.
...	Other arguments to be passed to the function. Not currently used.

Value

A list with the following components

votes1	A matrix of order KxR with the results simulated in each polling unit for the first election.
votes2	A matrix of order KxC with the results simulated in each polling unit for the second election..
TM.global	A matrix of order RxC with the actual simulated global transfer matrix of counts.
TM.units	An array of order RxCxK with the simulated transfer matrices of votes by polling unit. If simplify = TRUE the simulated transfer matrices of votes are returned organized in a Kx(RC) matrix.
inputs	A list containing all the objects with the values used as arguments by the function.

Author(s)

Antonio Forcina, <forcinarosara@gmail.com>

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

References

Brown, P. and Payne, C. (1986). Aggregate data, ecological regression and voting transitions. *Journal of the American Statistical Association*, 81, 453–460. doi:10.1080/01621459.1986.10478290

Forcina, A., Gnaldi, M. and Bracalente, B. (2012). A revised Brown and Payne model of voting behaviour applied to the 2009 elections in Italy. *Statistical Methods & Applications*, 21, 109–119. doi:10.1007/s102600110184x

See Also

Other simulators for ecological inference overdispersed-multinomial models: simula_BPF_with_deviations()

Examples

```
TMg <- matrix(c(0.6, 0.1, 0.3, 0.1, 0.7, 0.2, 0.1, 0.1, 0.8),
              byrow = TRUE, nrow = 3)
example <- simula_BPF(n.units = 100, TM = TMg, prop1 = c(0.3, 0.3, 0.4),
                      polling.sizes = c(750, 850))
```

simula_BPF_with_deviations *Simulate RxC Square Tables with Ecological Fallacy Effects Based on Overdispersed-Multinomial Models*

Description

Generates a set of RxC square (RxR) tables at random, representing the joint distribution of voters in two elections, according to the model proposed by Forcina et al. (2012) as an extension of Brown and Payne (1986), under the assumption that transition probabilities are non-homogeneous across local units. For each unit, a unique transition table is constructed to simulate voter behavior within that unit. Each table is created using a mixture model that considers four latent types of voters: one group following the underlying global transition probabilities of the BPF model, another composed mainly of loyal voters, a third characterized by strategic voting, and a final group whose probability of loyalty to the party they supported in the first election depends on that party's strength in the unit during the first election

Usage

```
simula_BPF_with_deviations(
  n.units,
  TM,
  prop1,
  polling.sizes,
  theta1 = 0.1,
  theta2 = 0.1,
  cs = 50,
  prop.dev = c(0.4, 0.6),
 prop.loyal = matrix(0.34, nrow = ifelse(is.null(dim(n.units)), n.units, nrow(n.units)),
    ncol = nrow(TM)),
  prop.strategic = matrix(0.33, nrow = ifelse(is.null(dim(n.units)), n.units,
    nrow(n.units)), ncol = nrow(TM)),
  prop.context = matrix(0.33, nrow = ifelse(is.null(dim(n.units)), n.units,
    nrow(n.units)), ncol = nrow(TM)),
  par.loyal = 0.95,
  par.strategic = 0.5,
  par.context = 0.5,
  simplify = FALSE,
  ...
)
```

Arguments

n.units	Either a positive integer number, K, indicating the number of polling units to be simulated, or a KxR data.frame of a matrix with the number of votes gained in election 1 for each of the R options in each of the K units. If n.units is a matrix

(data.frame) of counts (votes) the values of arguments prop1 and theta1 are ommitted.

TM — A row-standardized RxC matrix with the underlying global transition probabilities for the Overdispersed-Multinomial Model. If the matrix is not row-standardized, it is internally row-standardized by the function.

prop1 — A vector of length R with the initial assumed probabilities of voting (to be simulated) for each of the R competing options in the first election. If the provided vector is not a set of probabilities (i.e., a vector of positive numbers adding to 1), it is internally standardized by the function.

polling.sizes — Either a vector of two components with two positive integer numbers indicating the minimum and maximum number of voters for each unit or a vector of length n.units of positive integer numbers informing about the number of voters in each unit. When polling.sizes is a vector of length two, a number of voters is randomly assigned for each unit using a uniform distribution with parameters the minimum and maximum values included in polling.sizes.

theta1 — A number between 0 and 1 used as the overdispersion parameter. This parameter is employed by the underlying Dirichlet distribution, in conjunction with prop1, to randomly generate vectors of probabilities for each unit. These vectors are then used to simulate the results of the first election. The smaller the value of this parameter, the closer the unit-level marginal distributions for the first election are to prop1. Default, 0.1.

theta2 — Either a single number between 0 and 1 or a vector of length nrow(TM) containing numbers between 0 and 1. The values in theta2 serve as overdispersion parameters and are used alongside the row-probability vectors in TM within the underlying Dirichlet distributions. These distributions are employed to generate probability vectors for each combination of unit, cluster, and row, which are then used to simulate vote transfers from the first to the second election. If theta2 is a vector, each row is assigned a distinct overdispersion parameter based on its corresponding value. Default, 0.1.

cs — A positive number indicating the average number of cluster size. Default, 50.

prop.dev — Either a two-component vector with positive values between 0 and 1, indicating the minimum and maximum proportion of voters (to be simulated) that deviate from the base Overdispersed-Multinomial Model in each unit or a vector of length n.units specifying the proportion of voters deviating from the basic model in each unit. If prop.dev is a two-component vector, the proportion of deviating voters in each unit is randomly assigned using a uniform distribution with the specified minimum and maximum values. Default, c(0.4, 0.6).

prop.loyal — A KxR matrix where each cell (k, r) represents the proportion of voters from party r in unit k who are strongly loyal. These voters are highly likely to vote for the same party with near certainty (see the parameter par.loyal). In contrast, the remaining prop.dev percent of the voters from the party follow the transition probabilities specified in TM. The sum of the matrices prop.loyal, prop.strategic, and prop.contextual must equal one for each cell. If this condition is not met, the function internally standardizes the provided matrices. Default, matrix(0.34, nrow = ifelse(is.null(dim(n.units)), n.units, nrow(n.units)), ncol = nrow(TM)).

`prop.strategic` A KxR matrix where each cell `(k, r)` represents the proportion of voters from party `r` in unit `k` who are strategic voters. These voters are a `par.strategic` percent more likely to support parties that improve their results in the second election compared to their performance in their first election (see the parameter `par.strategic`). In contrast, the remaining `prop.dev` percent of the voters from the party follow the transition probabilities specified in `TM`. The sum of the matrices `prop.loyal`, `prop.strategic`, and `prop.contextual` must equal one for each cell. If this condition is not met, the function internally standardizes the provided matrices. Default, `matrix(0.33, nrow = ifelse(is.null(dim(n.units)), n.units, nrow(n.units)), ncol = nrow(TM))`.

`prop.context` A KxR matrix where each cell `(k, r)` represents the proportion of voters from party `r` in unit `k` who are influenced by the relative strength in their neighborhood of the party they voted for in the first election. These voters are a `par.context` multiplied by the party's strength in the unit percent more likely to support the same party in the second election (see the parameter `par.context`). In contrast, the remaining `prop.dev` percent of the voters from the party follow the transition probabilities specified in `TM`. The sum of the matrices `prop.loyal`, `prop.strategic`, and `prop.contextual` must equal one for each cell. If this condition is not met, the function internally standardizes the provided matrices. Default, `matrix(0.33, nrow = ifelse(is.null(dim(n.units)), n.units, nrow(n.units)), ncol = nrow(TM))`.

`par.loyal` A number between 0.9 and 1 indicating the minimum probability with which loyal voters will support the same party in the second election as they did in the first. For each unit, the probability is randomly chosen between `par.loyal` and 1. Default, `0.95`.

`par.strategic` A positive number indicating the proportion of increase that the initial transfer probabilities in `TM` should be increased for those parties improving their support in the second election compared to their performance in their first election. Default, `0.5`.

`par.context` A positive number indicating the factor by which the proportion of support for a party in each unit should be multiplied to increase the initial transfer probabilities in `TM` corresponding to that party. Default, `0.5`.

`simplify` A TRUE/FALSE argument indicating whether the simulated RxCxK array of counts by polling unit should be rearranged as a matrix of order Kx(RC). Default, FALSE.

`...` Other arguments to be passed to the function. Not currently used.

Value

A list with the following components

`votes1` A matrix of order KxR with the results simulated in each polling unit for the first election.

`votes2` A matrix of order KxC with the results simulated in each polling unit for the second election..

`TM.global` A matrix of order RxC with the actual simulated global transfer matrix of counts.

TM.units	An array of order RxCxK with the simulated transfer matrices of votes by polling unit. If simplify = TRUE the simulated transfer matrices of votes are returned organized in a Kx(RC) matrix.
inputs	A list containing all the objects with the values used as arguments by the function.

Author(s)

Antonio Forcina, <forcinarosara@gmail.com>

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

References

Brown, P. and Payne, C. (1986). Aggregate data, ecological regression and voting transitions. *Journal of the American Statistical Association*, 81, 453–460. doi:10.1080/01621459.1986.10478290

Forcina, A., Gnaldi, M. and Bracalente, B. (2012). A revised Brown and Payne model of voting behaviour applied to the 2009 elections in Italy. *Statistical Methods & Applications*, 21, 109–119. doi:10.1007/s102600110184x

See Also

Other simulators for ecological inference overdispersed-multinomial models: simula_BPF()

Examples

```
TMg <- matrix(c(0.6, 0.1, 0.3, 0.1, 0.7, 0.2, 0.1, 0.1, 0.8),
              byrow = TRUE, nrow = 3)
example <- simula_BPF_with_deviations(n.units = 100, TM = TMg, prop1 = c(0.3, 0.3, 0.4),
                                      polling.sizes = c(750, 850))
```

---

summary.BPF *Summarize a BPF output object*

Description

Summary method for objects obtained with the BPF function

Usage

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

Arguments

object	An object output of the **BPF** function.
...	Other arguments passed on to methods. Not currently used.

Value

An object of class "summary.BPF". A list with four components:

prop.matrix	A matrix of order RxC with the estimated underlying proportions/rates of the vote transitions from election 1 to election 2.
counts.matrix	A matrix of order RxC with the estimated vote transfers from election 1 to election 2.
row.margins	A vector of length R with the aggregate observed/adjusted distribution of proportions of votes in election 1.
col.margins	A vector of length C with the aggregate observed/adjusted distribution of proportions of votes in election 2.

Author(s)

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

Examples

```
votes1 <- structure(list(P1 = c(16L, 4L, 13L, 6L, 1L, 16L, 6L, 17L, 48L, 14L),
                         P2 = c(8L, 3L, 0L, 5L, 1L, 4L, 7L, 6L, 28L, 8L),
                         P3 = c(38L, 11L, 11L, 3L, 13L, 39L, 14L, 34L, 280L, 84L),
                         P4 = c(66L, 5L, 18L, 39L, 30L, 57L, 35L, 65L, 180L, 78L),
                         P5 = c(14L, 0L, 5L, 2L, 4L, 21L, 6L, 11L, 54L, 9L),
                         P6 = c(8L, 2L, 5L, 3L, 0L, 7L, 7L, 11L, 45L, 17L),
                         P7 = c(7L, 3L, 5L, 2L, 3L, 17L, 7L, 13L, 40L, 8L)),
                         row.names = c(NA, 10L), class = "data.frame")
votes2 <- structure(list(C1 = c(2L, 1L, 2L, 2L, 0L, 4L, 0L, 4L, 19L, 14L),
                         C2 = c(7L, 3L, 1L, 7L, 2L, 5L, 3L, 10L, 21L, 6L),
                         C3 = c(78L, 7L, 28L, 42L, 28L, 84L, 49L, 85L, 260L, 100L),
                         C4 = c(56L, 14L, 20L, 7L, 19L, 54L, 22L, 50L, 330L, 91L),
                         C5 = c(14L, 3L, 6L, 2L, 3L, 14L, 8L, 8L, 45L, 7L)),
                         row.names = c(NA, 10L), class = "data.frame")
example <- BPF(votes1, votes2, local = "none")
summary(example)
```

Index