Package: BradleyTerryScalable (via r-universe)

November 22, 2024

Type Package

Title Fits the Bradley-Terry Model to Potentially Large and Sparse Networks of Comparison Data

Version 0.1.0.9300

Date 2017

Author Ella Kaye [aut, cre], David Firth [aut]

Maintainer Ella Kaye <E.Kaye.1@warwick.ac.uk>

URL https://github.com/EllaKaye/BradleyTerryScalable

BugReports https://github.com/EllaKaye/BradleyTerryScalable/issues

Description Facilities are provided for fitting the simple, unstructured Bradley-Terry model to networks of binary comparisons. The implemented methods are designed to scale well to large, potentially sparse, networks. A fairly high degree of scalability is achieved through the use of EM and MM algorithms, which are relatively undemanding in terms of memory usage (relative to some other commonly used methods such as iterative weighted least squares, for example). Both maximum likelihood and Bayesian MAP estimation methods are implemented. The package provides various standard methods for a newly defined 'btfit' model class, such as the extraction and summarisation of model parameters and the simulation of new datasets from a fitted model. Tools are also provided for reshaping data into the newly defined ``btdata'' class, and for analysing the comparison network, prior to fitting the Bradley-Terry model. This package complements, rather than replaces, the existing 'BradleyTerry2' package. (BradleyTerry2 has rather different aims, which are mainly the specification and fitting of ``structured'' Bradley-Terry models in which the strength parameters depend on covariates.)

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 2.10)

Imports igraph, Matrix, Rcpp (>= 1.0.3), methods, purrr, dplyr

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.0.2

Suggests Matrix.utils, stringr, knitr, rmarkdown, ggplot2

VignetteBuilder knitr

Config/pak/sysreqs libglpk-dev libxml2-dev

Repository https://ellakaye.r-universe.dev

RemoteUrl https://github.com/EllaKaye/BradleyTerryScalable

RemoteRef HEAD

RemoteSha 908f5f4887bb48c14c2ea3872b14ad881e452c0a

Contents

`BradleyTerryScalable` *A package for fitting the Bradley-Terry model to (potentially) large and sparse data sets.*

Description

A package for fitting the Bradley-Terry model to (potentially) large and sparse data sets.

`btdata` *Create a btdata object*

Description

Creates a btdata object, primarily for use in the btfit function.

Usage

```
btdata(x, return_graph = FALSE)

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

Arguments

`x`	The data, which is either a three- or four-column data frame, a directed igraph object, a square matrix or a square contingency table. See Details.
`return_graph`	Logical. If TRUE, an igraph object representing the comparison graph will be returned.
`object`	An object of class "btdata", typically the result `ob` of `ob <- btdata(..)`.
`...`	Other arguments

Details

The `x` argument to `btdata` can be one of four types:

- A matrix (either a base `matrix`) or a class from the `Matrix` package), dimension K by K, where K is the number of items. The *i,j*-th element is w_{ij}, the number of times item i has beaten item j. Ties can be accounted for by assigning half a win (i.e. 0.5) to each item.
- A contingency table of class `table`, similar to the matrix described in the above point.
- An `igraph`, representing the *comparison graph*, with the K items as nodes. For the edges:
  - If the graph is unweighted, a directed edge from node i to node j for every time item i has beaten item j
  - If the graph is weighted, then one edge from node i to node j if item i has beaten item j at least once, with the weight attribute of that edge set to the number of times i has beaten j.
- If `x` is a data frame, it must have three or four columns:
  - 3-column data frameThe first column contains the name of the winning item, the second column contains the name of the losing item and the third columns contains the number of times that the winner has beaten the loser. Multiple entries for the same pair of items are handled correctly. If `x` is a three-column dataframe, but the third column gives a code for who won, rather than a count, see `codes_to_counts`.

- 4-column data frameThe first column contains the name of item 1, the second column contains the name of item 2, the third column contains the number of times that item 1 has beaten item 2 and the fourth column contains the number of times item 2 has beaten item 1. Multiple entries for the same pair of items are handled correctly. This kind of data frame is also the output of `codes_to_counts`.
- In either of these cases, the data can be aggregated, or there can be one row per comparison.
- Ties can be accounted for by assigning half a win (i.e. 0.5) to each item.

`summary.btdata` shows the number of items, the density of the `wins` matrix and whether the underlying comparison graph is fully connected. If it is not fully connected, `summary.btdata` will additional show the number of fully-connected components and a table giving the frequency of components of different sizes. For more details on the comparison graph, and how its structure affects how the Bradley-Terry model is fitted, see `btfit` and the vignette: https://ellakaye.github.io/BradleyTerryScalable/articles/BradleyTerryScalable.html.

Value

An object of class "btdata", which is a list containing:

`wins`	A K by K square matrix, where K is the total number of players. The i, j-th element is w_{ij}, the number of times item i has beaten item j. If the items in `x` are unnamed, the wins matrix will be assigned row and column names 1:K.
`components`	A list of the fully-connected components.
`graph`	The comparison graph of the data (if return_graph = TRUE). See Details.

Author(s)

Ella Kaye

See Also

`codes_to_counts` `select_components`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
summary(citations_btdata)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
summary(toy_btdata)
```

`btfit` *Fits the Bradley-Terry model*

Description

`btfit` fits the Bradley-Terry model on (potentially) large and sparse datasets.

Usage

```
btfit(
  btdata,
  a,
  MAP_by_component = FALSE,
  subset = NULL,
  maxit = 10000,
  epsilon = 0.001
)
```

Arguments

`btdata`	An object of class "btdata", typically the result ob of ob <- btdata(..). See `btdata`.
`a`	Must be >= 1. When a = 1, the function returns the maximum likelihood estimate (MLE) of π (by component, if necessary). When a > 1, a is the shape parameter for the Gamma prior. See Details.
`MAP_by_component`	Logical. Only considered if a > 1. Then, if FALSE, the MAP estimate will be found on the full dataset. If TRUE, the MAP estimate will be found separately for each fully-connected component.
`subset`	A condition for selecting a subset of the components. This can either be a character vector of names of the components, a single predicate function (that takes a component as its argument), or a logical vector of the same length as the number of components).
`maxit`	The maximum number of iterations for the algorithm. If returning π by component, this will be the maximum number of iterations for each component.
`epsilon`	Determines when the algorithm is deemed to have converged. (See Details.)

Details

Let there be K items, let π_k be the Bradley-Terry strength parameter of item k, for $k = 1, \ldots, K$ and let π be the vector of all the π_k. Let w_{ij} be the number of times item i wins against item j, let $n_{ij} = w_{ij} + w_{ji}$ be the number of times they play, with $w_{ii} = 0$ by convention and let $W_i = \sum_{j=1}^{K} w_{ij}$. Then the Bradley-Terry model states that the probability of item i beating item j, p_{ij}, is:

$$p_{ij} = \frac{\pi_i}{\pi_i + \pi_j}.$$

The comparison graph, G_W, has the K players as the nodes and a directed edge from node i to node j whenever item i has beaten item j at least once. The MLE of the Bradley-Terry model exists and is finite if and only if the comparison graph is fully-connected (i.e. if there is a directed path from node i to node j for all items i and j).

Assuming that the comparison graph of the data is fully-connected, the MLE of the Bradley-Terry model can be found using the MM-algorithm (Hunter, 2004).

If the comparison graph of the data is not fully-connected, there are two principled options for fitting the Bradley-Terry model. One is to find the MLE within each fully-connected component. The other is to find the Bayesian MAP estimate, as suggested by Caron & Doucet (2012), where a $Gamma(a, b)$ gamma prior is placed on each π_i, and the product of these is taken as a prior on π. The MAP estimate can then be found with an EM-algorithm. When $a = 1$ and $b = 0$, the EM and MM-algorithms are equivalent and the MAP estimate and MLE are identical. The rate parameter of the Gamma prior, b, is not likelihood identifiable. When $a > 1$, b is set to $aK - 1$, where K is the number of items in the component; this choice of b minimises the number of iterations needed for the algorithm to converge.

The likelihood equations give

$$a - 1 + W_i = b\pi_i + \sum_{j \neq i} \frac{n_{ij}\pi_i}{\pi_i + \pi_j},$$

for $i = 1, \ldots, K$. For the algorithm to have converged, we want π to be such that the LHS and RHS of this equation are close for all i. Therefore, we set the convergence criteria as

$$\left| \frac{a - 1 + W_i}{b\pi_i + \sum_{j \neq i} \frac{n_{ij}\pi_i}{\pi_i + \pi_j}} - 1 \right| < \epsilon,$$

for all i.

Since the equations do not typeset well within the R help window, we recommend reading this section online: https://ellakaye.github.io/BradleyTerryScalable/reference/btfit.html.

Value

`btfit` returns an S3 object of class "btfit". It is a list containing the following components:

`call`	The matched call
`pi`	A list of length M, where M is the number of fully-connected components of the comparison graph G_W (or the requested subset) of two or more items. The m-th list item is a named vector π, the strength parameter, for the items in the m-th fully connected component, $m = 1, \ldots, M$. These are sorted in descending order.
`iters`	A vector of length M. The m-th entry is the number of iterations it took for the algorithm to converge for the m-th component, for $m = 1, \ldots, M$. Note that if the algorithm has not converged in any component, a warning will be produced.

`converged` A logical vector of length M, indicating whether the algorithm has converged for the m-th component in `maxit` iterations.

`N` A list of length M. The m-th list item is a matrix where each element n_{ij} is the number of times item i played against item j, for the items in the m-th component. The rows and columns are arranged in the same order as the ordered pi vector(s).

`diagonal` A list of length M. The m-th item is a vector of the diagonal elements of the `btdata$wins` matrix, for the items in the m-th fully-connected component. These values are used as the fitted values for the diagonal of the matrix output in `fitted.btfit`.

`names_dimnames` The names of the dimnames of the original `btdata$wins` matrix.

Author(s)

Ella Kaye, David Firth

References

Caron, F. and Doucet, A. (2012) Efficient Bayesian Inference for Generalized Bradley-Terry Models. *Journal of Computational and Graphical Statistics*, **21**(1), 174-196.

Hunter, D. R. (2004) MM Algorithms for Generalized Bradley-Terry Models. *The Annals of Statistics*, **32**(1), 384-406.

See Also

`btdata`, `summary.btfit`, `coef.btfit`, `fitted.btfit`, `btprob`, `vcov.btfit`, `simulate.btfit`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
summary(fit1)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
summary(fit2a)
fit2b <- btfit(toy_btdata, 1.1)
summary(fit2b)
fit2c <- btfit(toy_btdata, 1, subset = function(x) length(x) > 3)
summary(fit2c)
```

`btprob` *Calculates Bradley-Terry probabilities*

Description

Calculates the Bradley-Terry probabilities of each item in a fully-connected component of the comparison graph, G_W, winning against every other item in that component (see Details).

Usage

```
btprob(object, subset = NULL, as_df = FALSE)
```

Arguments

`object`	An object of class "btfit", typically the result `ob` of `ob <- btfit(..)`. See `btfit`.
`subset`	A condition for selecting one or more subsets of the components. This can either be a character vector of names of the components (i.e. a subset of `names(object$pi)`), a single predicate function (that takes a vector of `object$pi` as its argument), or a logical vector of the same length as the number of components, (i.e. `length(object$pi)`).
`as_df`	Logical scalar, determining class of output. If `TRUE`, the function returns a data frame. If `FALSE` (the default), the function returns a matrix (or list of matrices). Note that setting `as_df = TRUE` can have a significant computational cost when any of the components have a large number of items.

Details

Consider a set of K items. Let the items be nodes in a graph and let there be a directed edge (i, j) when i has won against j at least once. We call this the comparison graph of the data, and denote it by G_W. Assuming that G_W is fully connected, the Bradley-Terry model states that the probability that item i beats item j is

$$p_{ij} = \frac{\pi_i}{\pi_i + \pi_j},$$

where π_i and π_j are positive-valued parameters representing the skills of items i and j, for $1 \leq i, j, \leq K$. The function `btfit` can be used to find the strength parameter π. It produces a `"btfit"` object that can then be passed to `btprob` to obtain the Bradley-Terry probabilities p_{ij}.

If G_W is not fully connected, then a penalised strength parameter can be obtained using the method of Caron and Doucet (2012) (see `btfit`, with `a > 1`), which allows for a Bradley-Terry probability of any of the K items beating any of the others. Alternatively, the MLE can be found for each fully connected component of G_W (see `btfit`, with `a = 1`), and the probability of each item in each component beating any other item in that component can be found.

Value

If `as_df = FALSE`, returns a matrix where the i, j-th element is the Bradley-Terry probability p_{ij}, or, if the comparison graph, G_W, is not fully connected and `btfit` has been run with `a = 1`, a list of such matrices for each fully-connected component of G_W. If `as_df = TRUE`, returns a five-column

data frame, where the first column is the component that the two items are in, the second column is `item1`, the third column is `item2`, the fourth column is the Bradley-Terry probability that item 1 beats item 2 and the fifth column is the Bradley-Terry probability that item 2 beats item 1. If the original `btdata$wins` matrix has named dimnames, these will be the `colnames` for columns one and two. See Details.

Author(s)

Ella Kaye

References

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: 1. The method of paired comparisons. *Biometrika*, **39**(3/4), 324-345.

Caron, F. and Doucet, A. (2012). Efficient Bayesian Inference for Generalized Bradley-Terry Models. *Journal of Computational and Graphical Statistics*, **21**(1), 174-196.

See Also

`btfit`, `btdata`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
btprob(fit1)
btprob(fit1, as_df = TRUE)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
btprob(fit2a)
btprob(fit2a, as_df = TRUE)
btprob(fit2a, subset = function(x) "Amy" %in% names(x))
fit2b <- btfit(toy_btdata, 1.1)
btprob(fit2b, as_df = TRUE)
```

BT_EM *Fit the Bradley-Terry model using the EM or MM algorithm*

Description

Fit the Bradley-Terry model using the EM or MM algorithm

Usage

```
BT_EM(W, a, b, maxit = 5000L, epsilon = 0.001)
```

Arguments

W	a K*K square matrix of class "dgCMatrix"
a	the shape parameter of the gamma prior
b	the rate parameter of the gamma prior
maxit	the maximum number of iterations
epsilon	controls the convergence criteria

Value

A list containing a K*1 matrix with the pi estimate, the N matrix, the number of iterations, and whether the algorithm converged.

---

citations *Statistics Journal Citation Data from Stigler (1994)*

Description

Extracted from a larger table in Stigler (1994). Inter-journal citation counts for four journals, "Biometrika", "Comm Statist.", "JASA" and "JRSS-B", as used on p448 of Agresti (2002)

Usage

```
citations
```

Format

A four by four matrix, where the i, j-th element is the number of times journal i has been cited by journal j.

Details

In the context of paired comparisons, the 'winner' is the cited journal and the 'loser' is the one doing the citing.

This dataset also appears in the `BradleyTerry2` package.

Source

Agresti, A. (2002) *Categorical Data Analysis* (2nd ed.). New York: Wiley

References

Stigler, S. (1994) Citation patterns in the journals of statistics and probability. *Statistical Science*, **9**, 384-406.

codes_to_counts *Converts data frame with a code for wins to counts of wins*

Description

Convert a three-column data frame in which the third column is a code representing whether the item in column 1 won, lost or (if applicable) drew over/with the item in column 2, to a dataframe with counts (suitable for use in btdata)

Usage

```
codes_to_counts(df, codes)
```

Arguments

df	A three-column data frame. Each row represents a comparison between two items. The first and second columns are the names of the first and second items respectively. The third column gives a code for which won. See Details and Examples.
codes	A numeric vector or character vector, of length two or three (depending on whether there are ties.) The first and second element gives the codes used if the first or second item won respectively. If there are ties, the third element gives the code used in that case. See Details and Examples.

Details

This function is needed in the BradleyTerryScalable workflow when the user data is stored in a three-column data frame where each row is a comparison between two items, and where the third column is NOT a count of the number of times the item in the first column beat the item in the second column. Rather, it could be that the third column is a code for which of the two items won (including the possibility of a tie), for example "W1", "W2", "D". Or else, it could be that the third column gives the score only in relation to the first item, e.g. 1 for a win, 0 for a loss or 0.5 for a draw without there anywhere in the table being the corresponding record for the second item (i.e. respectively 0 for a loss, 1 for a win and 0.5 for a draw.)

Value

A four-column data frame where the first two columns are the name of the first and second item. The third and fourth column gives the wins count for the first and second item respectively: 1 for a win, 0 for a loss, and 0.5 each for a draw. This data frame is in the correct format to be passed to btdata

Author(s)

Ella Kaye

See Also

btdata

Examples

```
first <- c("A", "A", "B", "A")
second <- c("B", "B", "C", "C")
df1 <- data.frame(player1 = first, player2 = second, code = c("W1", "W2", "D", "D"))
codes_to_counts(df1, c("W1", "W2", "D"))
df2 <- data.frame(item1 = first, item2 = second, result = c(0, 1, 1, .5))
codes_to_counts(df2, c(1, 0, .5))
df3 <- data.frame(player1 = first, player2 = second, which_won = c(1,2,2,1))
codes_to_counts(df3, c(1,2))
codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
```

---

coef.btfit *Extract coefficients of a 'btfit' object*

Description

coef method for class "btfit"

Usage

```
## S3 method for class 'btfit'
coef(object, subset = NULL, ref = NULL, as_df = FALSE, ...)
```

Arguments

object	An object of class "btfit", typically the result ob of ob <- btfit(..). See btfit.
subset	A condition for selecting one or more subsets of the components. This can either be a character vector of names of the components (i.e. a subset of names(object$pi)), a single predicate function (that takes a vector of object$pi as its argument), or a logical vector of the same length as the number of components, (i.e. length(object$pi)).
ref	A reference item. Either a string with the item name, or the number 1, or NULL. If NULL, then the coefficients are constrained such that their mean is zero. If an item name is given, the coefficient estimates are shifted so that the coefficient for the ref item is zero. If there is more than one component, the components that do not include the ref item will be treated as if ref = NULL. If ref = 1, then the first item of each component is made the reference item.
as_df	Logical scalar, determining class of output. If TRUE, the function returns a data frame. If FALSE (the default), the function returns a named vector (or list of such vectors).
...	other arguments

Details

Note that the values given in the `estimate` column of the `item_summary` element are NOT the same as the values in `object$pi`. Rather, they are the λ_i, where $\lambda_i = \log \pi_i$. By default, these are normalised so that mean(λ_i) = 0. However, if `ref` is not equal to `NULL`, then the λ_i in the component in which `ref` appears are shifted to $\lambda_i - \lambda_{ref}$, for $i = 1, \ldots, K_c$, where K_c is the number of items in the component in which `ref` appears, and λ_{ref} is the estimate for the reference item.

Value

If as_df = TRUE, a data frame a numeric vector of estimated coefficients, where the first column is the component the item is in, the second column in the item and the third column in the coefficient. If as_df = FALSE, then a numeric vector is returned if the model is fitted on the full dataset, or else a list of numeric vectors is returned, one for each fully connected component. Within each component, the items are arranged by estimate, in descending order.

Author(s)

Ella Kaye

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
coef(fit1)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
coef(fit2a)
coef(fit2a, subset = function(x) length(x) > 3, as_df = TRUE)
coef(fit2a, subset = function(x) "Amy" %in% names(x))
coef(fit2a, as_df = TRUE)
fit2b <- btfit(toy_btdata, 1.1)
coef(fit2b)
coef(fit2b, ref = "Cyd")
```

`fitted.btfit` *Fitted Method for "btfit"*

Description

`fitted.btfit` returns the fitted values from a fitted btfit model object.

Usage

```
## S3 method for class 'btfit'
fitted(object, subset = NULL, as_df = FALSE, ...)
```

Arguments

`object`	An object of class "btfit", typically the result `ob` of `ob <- btfit(..)`. See `btfit`.
`subset`	A condition for selecting one or more subsets of the components. This can either be a character vector of names of the components (i.e. a subset of `names(object$pi)`), a single predicate function (that takes a vector of `object$pi` as its argument), or a logical vector of the same length as the number of components, (i.e. `length(object$pi)`).
`as_df`	Logical scalar, determining class of output. If `TRUE`, the function returns a data frame. If `FALSE` (the default), the function returns a matrix (or list of matrices). Note that setting `as_df = TRUE` can have a significant computational cost when any of the components have a large number of items.
`...`	Other arguments

Details

Consider a set of K items. Let the items be nodes in a graph and let there be a directed edge (i, j) when i has won against j at least once. We call this the comparison graph of the data, and denote it by G_W. Assuming that G_W is fully connected, the Bradley-Terry model states that the probability that item i beats item j is

$$p_{ij} = \frac{\pi_i}{\pi_i + \pi_j},$$

where π_i and π_j are positive-valued parameters representing the skills of items i and j, for $1 \leq i, j, \leq K$.

The expected, or fitted, values under the Bradley-Terry model are therefore:

$$m_{ij} = n_{ij} p_{ij},$$

where n_{ij} is the number of comparisons between item i and item j.

If there are values on the diagonal in the original `btdata$wins` matrix, then these appear as the values on the diagonal of the fitted matrix. These values do not appear in the data frame if the `as_df` argument is set to `TRUE`.

The function `btfit` is used to fit the Bradley-Terry model. It produces a `"btfit"` object that can then be passed to `fitted.btfit` to obtain the fitted values m_{ij}. Note that the Bradley-Terry probabilities p_{ij} can be calculated using `btprob`.

If G_W is not fully connected, then a penalised strength parameter can be obtained using the method of Caron and Doucet (2012) (see `btfit`, with `a > 1`), which allows for a Bradley-Terry probability of any of the K items beating any of the others. Alternatively, the MLE can be found for each fully-connected component of G_W (see `btfit`, with `a = 1`), and the probability of each item in each component beating any other item in that component can be found.

Value

If `as_df = FALSE` and the model has been fit on the full dataset, returns a matrix where the i, j-th element is the Bradley-Terry expected value m_{ij} (See Details). Otherwise, a list of such matrices is returned, one for each fully-connected component. If `as_df = TRUE`, returns a five-column data frame, where the first column is the component that the two items are in, the second column is `item1`, the third column is `item2`, the fourth column, `fit1`, is the expected number of times that

item 1 beats item 2 and the fifth column, `fit2`, is the expected number of times that item 2 beats item 1. If `btdata$wins` has named dimnames, these will be the `colnames` for columns one and two. Otherwise these colnames will be `item1` and `item2`. See Details.

Author(s)

Ella Kaye

References

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: 1. The method of paired comparisons. *Biometrika*, **39**(3/4), 324-345.

Caron, F. and Doucet, A. (2012). Efficient Bayesian Inference for Generalized Bradley-Terry Models. *Journal of Computational and Graphical Statistics*, **21**(1), 174-196.

See Also

`btfit`, `btprob`, `btdata`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
fitted(fit1)
fitted(fit1, as_df = TRUE)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
fitted(fit2a)
fitted(fit2a, as_df = TRUE)
fitted(fit2a, subset = function(x) "Amy" %in% names(x))
fit2b <- btfit(toy_btdata, 1.1)
fitted(fit2b, as_df = TRUE)
```

`select_components` *Subset a btdata object*

Description

Subset a btdata object by selecting components from it.

Usage

```
select_components(btdata, subset, return_graph = FALSE)
```

Arguments

btdata	An object of class "btdata", typically the result ob of ob <- btdata(..). See btdata.
subset	A condition for selecting a subset of the components. This can either be a character vector of names of the components, a single predicate function (that takes a component as its argument), or a logical vector of the same length as the number of components).
return_graph	Logical. If TRUE, an igraph object representing the comparison graph of the selected components will be returned.

Value

A btdata object, which is a list containing:

wins	A square matrix, where the i, j-th element is the number of times item i has beaten item j.
components	A list of the fully-connected components. The names of the list preserve the names of the original btdata object.
graph	The comparison graph of the selected components (if return_graph = TRUE).

Author(s)

Ella Kaye

See Also

btdata, btfit

Examples

```
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
## The following all return the same component
select_components(toy_btdata, "3", return_graph = TRUE)
select_components(toy_btdata, function(x) length(x) == 4)
select_components(toy_btdata, function(x) "Cyd" %in% x)
select_components(toy_btdata, c(FALSE, FALSE, TRUE))
```

---

simulate_BT

This function simulates one or more pseudo-random datasets from a specified Bradley-Terry model. Counts are simulated from independent binomial distributions, with the binomial probabilities and totals specified through the function arguments.

Description

This function simulates one or more pseudo-random datasets from a specified Bradley-Terry model. Counts are simulated from independent binomial distributions, with the binomial probabilities and totals specified through the function arguments.

Usage

```
simulate_BT(
  pi,
  N,
  nsim = 1,
  seed = NULL,
  result_class = c("sparseMatrix", "btdata")
)

## S3 method for class 'btfit'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  result_class = c("sparseMatrix", "btdata"),
  ...
)
```

Arguments

`pi`	a numeric vector, with all values finite and positive. The vector of item strengths in the Bradley-Terry model.
`N`	a symmetric, numeric matrix with dimensions the same as `length(pi)`. The elements should be numeric representations of non-negative integers. These are the binomial totals to use for the simulated data.
`nsim`	a scalar integer, the number of datasets to be generated.
`seed`	an object specifying if and how the random number generator should be initialized ('seeded'). For details see `simulate`.
`result_class`	a character vector specifying whether the generated datasets should be of class "sparseMatrix" or of class "btdata". If not specified, the first match among those alternatives is used.
`object`	An object of class "btfit", typically the result of `ob` of `ob <- btfit(..)`. This object must only have one component, i.e. `length(object$pi) == 1`.
`...`	Other arguments

Value

a list of length `nsim` of simulated datasets. If `result_class = "sparseMatrix"`, the datasets are sparse matrices with the same dimensions as `N`. If `result_class = "btdata"` then the datasets are "btdata" objects. See `btdata`

Author(s)

David Firth

See Also

btfit, btdata

Examples

```
set.seed(1)
n <- 6
N <- matrix(rpois(n ^ 2, lambda = 1), n, n)
N <- N + t(N) ; diag(N) <- 0
p <- exp(rnorm(n)/4)
names(p) <- rownames(N) <- colnames(N) <- letters[1:6]
simulate_BT(p, N, seed = 6)
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
simulate(fit1, nsim = 2, seed = 1)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2 <- btfit(toy_btdata, 1, subset = function(x) "Amy" %in% x)
fit2_sim <- simulate(fit2, nsim = 3, result_class = "btdata")
fit2_sim$sim_1
purrr::map(fit2_sim, "wins")
```

---

summary.btfit *Summarizing Bradley-Terry Fits*

Description

summary method for class "btfit"

Usage

```
## S3 method for class 'btfit'
summary(object, subset = NULL, ref = NULL, SE = FALSE, ...)
```

Arguments

object	An object of class "btfit", typically the result ob of ob <- btfit(..). See btfit.
subset	A condition for selecting one or more subsets of the components. This can either be a character vector of names of the components (i.e. a subset of names(object$pi)), a single predicate function (that takes a vector of object$pi as its argument), or a logical vector of the same length as the number of components, (i.e. length(object$pi)).

`ref` A reference item. Either a string with the item name, or the number 1, or NULL. If NULL, then the coefficients are constrained such that their mean is zero. If an item name is given, the coefficient estimates are shifted so that the coefficient for the ref item is zero. If there is more than one component, the components that do not include the ref item will be treated as if ref = NULL. If ref = 1, then the first item of each component is made the reference item.

`SE` Logical. Whether to include the standard error of the estimate in the `item_summary` table. Default is `FALSE`. **N.B. calculating the standard error can be slow when the number of items is large**. See `vcov.btfit`.

`...` other arguments

Details

Note that the values given in the `estimate` column of the `item_summary` element are NOT the same as the values in `object$pi`. Rather, they are the λ_i, where $\lambda_i = \log \pi_i$ (i.e. the coefficients as found by They are the coefficients, as found by `coef.btfit`.). By default, these are normalised so that mean(λ_i) = 0. However, if `ref` is not equal to `NULL`, then the λ_i in the component in which `ref` appears are shifted to $\lambda_i - \lambda_{ref}$, for $i = 1, \ldots, K_c$, where K_c is the number of items in the component in which `ref` appears, and λ_{ref} is the estimate for the reference item.

Value

An S3 object of class `"summary.btfit"`. It is a list containing the following components:

`item_summary` A `tibble` with columns for the item name, its coefficient, the standard error and the component it is in. Within each component, the items are arranged by estimate, in descending order. Note that the `estimate` is NOT the same as the values in `summary$pi`. See Details.

`component_summary`
A `tibble` with a row for each component in the `btfit` object (named according to the original `btdata$components`, with the number of items in the component, the number of iterations the fitting algorithm ran for, and whether it converged.

Author(s)

Ella Kaye

See Also

`btfit`, `coef.btfit`, `vcov.btfit`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
fit1 <- btfit(citations_btdata, 1)
summary(fit1)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
summary(fit2a)
```

```
fit2b <- btfit(toy_btdata, 1.1)
summary(fit2b, SE = TRUE)
fit2c <- btfit(toy_btdata, 1)
summary(fit2c, subset = function(x) "Amy" %in% names(x))
summary(fit2c, subset = function(x) length(x) > 3, ref = "Amy")
```

`toy_data` *A toy data set for the* `BradleyTerryScalable` *package*

Description

A toy data set, where the underlying comparison graph of the players is not fully connected. Each row represents one game.

Usage

```
toy_data
```

Format

A data frame with 13 rows and 3 variables:

player1 The name of player1

player2 The name of player2

outcome Outcome of the game: `"W1"` if player1 beats player2, `"W2"` if player2 beats player2 and `"D"`if it was a draw.

`vcov.btfit` *Calculate variance-covariance matrix for a btfit object*

Description

`vcov` method for class "btfit"

Usage

```
## S3 method for class 'btfit'
vcov(object, subset = NULL, ref = NULL, ...)
```

Arguments

`object`	An object of class "btfit", typically the result ob of `ob <- btfit(..)`. See `btfit`.
`subset`	A condition for selecting one or more subsets of the components. This can either be a character vector of names of the components (i.e. a subset of `names(object$pi)`), a single predicate function (that takes a vector of `object$pi` as its argument), or a logical vector of the same length as the number of components, (i.e. `length(object$pi)`).
`ref`	A reference item. Either a string with the item name, or the number 1, or NULL. If NULL, then the coefficients are constrained such that their mean is zero. If an item name is given, the coefficient estimates are shifted so that the coefficient for the ref item is zero. If there is more than one component, the components that do not include the ref item will be treated as if ref = NULL. If ref = 1, then the first item of each component is made the reference item.
`...`	other arguments

Details

N.B. this can be slow when there are a large number of items in any component.

Value

A square numeric matrix, which is a non-full-rank variance-covariance matrix for the estimates in `coef(object, subset = subset, ref = ref)`; or a list of such matrices if `object` has more than one component. The rows and columns of the matrix (or matrices) are arranged in the same order as the `object$pi` vector(s). -#' @author David Firth, Ella Kaye

See Also

`btfit`, `coef.btfit`, `summary.btfit`

Examples

```
citations_btdata <- btdata(BradleyTerryScalable::citations)
#' fit1 <- btfit(citations_btdata, 1)
#' vcov(fit1)
toy_df_4col <- codes_to_counts(BradleyTerryScalable::toy_data, c("W1", "W2", "D"))
toy_btdata <- btdata(toy_df_4col)
fit2a <- btfit(toy_btdata, 1)
vcov(fit2a)
vcov(fit2a, subset = function(x) length(x) > 3)
vcov(fit2a, subset = function(x) "Cyd" %in% names(x))
fit2b <- btfit(toy_btdata, 1.1)
vcov(fit2b, ref = "Cyd")
```

Index