Package 'brittnu'

April 28, 2023

Title Britt's Nu and Euclidean Krippendorff's Alpha

Version 0.2.0

Description This package computes two reliability coefficients. Britt's nu is used to assess reliability for data sets composed of multiple Dirichlet-distributed samples. Euclidean Krippendorff's alpha is used to assess reliability for Euclidean and compositional data that do not necessarily adhere to a Dirichlet distribution. These measures are especially useful for the cross-validation of topic models.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports utils,
stats,
methods,
plyr,
topicmodels,
gtools

R topics documented:

`brittnu` *Britt's Nu*

Description

This function computes the Britt's nu reliability coefficient for data sets composed of multiple Dirichlet-distributed deviates, as described by Britt (under review).

Usage

```
brittnu(
  x,
  type = NA,
  alpha = NA,
  symmetric_alpha = FALSE,
  pairwise = TRUE,
  estimate_pairwise_alpha_from_joint = TRUE,
  force_bootstrapping = FALSE,
  shuffle = FALSE,
  shuffle_method = "rotational",
  shuffle_dimension = NA,
  clustering_method = "average",
  slow_clustering = FALSE,
  samples = 1000,
  sampling_method = "parametric",
  lower_bound = FALSE,
  convcrit = 1e-05,
  maxit = 1000,
  progress = FALSE,
  verbose = FALSE,
  different_documents = FALSE,
  zero = (10^(-16)),
  tol = 1e-04
)
```

Arguments

`x`	A list such that each element is either the output from a `LDA` function call or a 2D double vector with observations as the rows and each category within each observation as the columns (such that each row in the vector sums to 1)

type
: A string indicating the type of reliability being assessed, with permitted values of "wt" (word-topic reliability), "td" (topic-document reliability), and NA; if x contains the output from multiple LDA function calls, this must be set to either "wt" (word-topic reliability) or "td" (topic-document reliability)

alpha
: A list whose length is equal to the number of observations, with each list element representing a vector comprising the concentration parameters for the categories in the corresponding observation; if this is NA and sampling_method=="parametric", the concentration parameters will be estimated from x

symmetric_alpha
: A boolean value indicating whether concentration parameters should be assumed to be unchanged between observations, which is common in many topic modeling procedures; this parameter only has an effect when alpha=NA

pairwise
: A boolean value indicating whether pairwise reliability between individual raters should be computed; if FALSE, pairwise reliability will be ignored in order to reduce the time and memory complexity of the computation

estimate_pairwise_alpha_from_joint
: A boolean value indicating, when estimating reliability for pairs of raters, whether each pair of raters should use the concentration parameters estimated across all raters (TRUE) or whether separate sets of concentration parameters should be estimated for each pair of raters (FALSE); this argument only has an effect when alpha==NA, and it is strongly suggested that the default value of TRUE be used for this argument when possible

force_bootstrapping
: A boolean value indicating whether bootstrapping should be used to estimate expected differences rather than exactly computing them even if shuffle==FALSE

shuffle
: A boolean value indicating whether or not the topics may have been shuffled into different sequences by each rater; if this is TRUE and shuffle_dimension="rows", then the observations (rows) corresponding to each rater will be reordered to achieve a (local) optimal fit, whereas if this is TRUE and shuffle_dimension="columns", the categories (columns) will be reordered instead

shuffle_method
: A string indicating what method ("rotational", "agglomerative", or "divisive") will be used to reorder topics

shuffle_dimension
: A string indicating whether "rows" or "columns" should be reordered; this argument only has an effect if shuffle==TRUE and type==NA

clustering_method
: A string indicating the criterion that will be used to merge or divide clusters if shuffle==TRUE and either shuffle_method=="agglomerative" or shuffle_method=="divisive" permitted values are "average", "minimum", "maximum", "median", "centroid", and "wald"

slow_clustering
: A boolean value indicating whether the distances between clusters should be recalculated after each individual observation is added to the new cluster (TRUE), which can improve the cohesiveness of the resulting cluster, or whether all observations should be added to the new cluster without recomputing the distances between clusters after every addition (FALSE); this argument only has an effect if shuffle==TRUE and shuffle_method=="divisive"

samples
: An integer value indicating how many samples to use to estimate expected differences when shuffle==TRUE

`sampling_method`
: A string indicating whether bootstrapped sampling should be performed using a `"parametric"` or `"nonparametric"` approach when `shuffle==TRUE`

`lower_bound`
: A boolean value indicating whether the lower bound of the expected differences (based on i.i.d. beta distributions) should be used rather than shuffling Dirichlet distributions to estimate the exact difference in every sample when `shuffle==TRUE`

`convcrit`
: A numeric value indicating the threshold for convergence used to estimate concentration parameters when `alpha==NA`, `sampling_method=="parametric"`, and concentration parameters could not be directly obtained from the output of a `LDA` object provided as `x`

`maxit`
: A numeric value indicating the maximum number of iterations used to estimate concentration parameters when `alpha==NA`, `sampling_method=="parametric"`, and concentration parameters could not be directly obtained from the output of a `LDA` object provided as `x`

`progress`
: A boolean value indicating whether progress updates should be provided when estimating concentration parameters, if doing so is necessary

`verbose`
: A boolean value indicating whether brittnu and its helper functions should provide progress updates beyond the estimation of concentration parameters

`different_documents`
: A boolean value indicating, if each list element in `x` represents the output from an `LDA` function call, whether some raters used different sets of documents than others

`zero`
: A numeric value; if `type=="wt"` and `different_documents==TRUE`, then whenever a word appears in the portion of the corpus evaluated by some raters but not others, this value is assigned as its allocation to all topics for any rater that did not evaluate that word

`tol`
: A numeric value representing the tolerance for observations whose sums are greater than or less than `1`; a warning message appears if the sum of the values for any observation is further from `1` than this value

Details

One of the most important uses for Britt's nu is to assess the reliability of the allocations of words to topics or of topics to documents via topic modeling techniques such as latent Dirichlet allocation (LDA). When the results of multiple LDA cross-validation iterations obtained via `LDA` are provided as `x`, the `type` parameter should be set to either `"wt"` or `"td"` to indicate whether word-topic or topic-document reliability, respectively, should be assessed.

Crucially, different topic modeling cross-validation iterations (whether LDA or otherwise), which effectively represent distinct raters of the same data set, may result in the same topics appearing in a different sequence. As such, those topics, which may represent either the rows or the columns of the data set, often need to be reordered in order to yield an optimal fit.

In practice, it is generally infeasible to assess every possible combination of topics across all raters. As such, brittnu provides three methods of converging toward an optimal topic order for each rater. The default, `shuffle_method="rotational"`, takes the provided sequence and swaps pairs of categories until no further swaps would further improve the fit. `shuffle_method="agglomerative"` and `shuffle_method="divisive"` instead perform hierarchical cluster analyses of all topics, with the added restriction that two topics constructed by the same rater may not appear in the same cluster. Notably, the `"agglomerative"` option can take excessive time in some cases, while the

"divisive" option forms clusters based on the macro-level dynamics of the data and may therefore yield weakly matched sets of individual topics compared to the other available options.

Regardless of whether the rows or columns of the data set must be reordered, when estimating Britt's nu, it is common to compute four separate coefficients in this family: Britt's single-observation nu for all raters, Britt's single-observation nu for pairs of raters, Britt's omnibus nu for all raters, and Britt's single-observation nu for pairs of raters. For example, when assessing the reliability of multiple latent Dirichlet allocation cross-validation folds that each used the same data set and number of topics, single-observation reliability can be used to assess the reliability of the allocations of words to each individual topic, while multiple-observation reliability indicates the reliability of the allocations of words to the set of all topics. Likewise, the coefficients that take all raters into account generally more valuable for the summative assessment of reliability, but the pairwise coefficients are sometimes useful for diagnostic purposes.

By default, brittnu provides all four reliability coefficients. To reduce the time and memory required, the pairwise versions may optionally be omitted from the computation by specifying pairwise=FALSE.

Also by default, the expected difference component of Britt's nu is estimated using parametric bootstrapping based on the distribution of the original data. This process can sometimes become computationally intensive. In such cases, it may be advisable to set sampling_method="nonparametric" in order to use nonparametric bootstrapping rather than parametric bootstrapping. You may also consider setting lower_bound=TRUE to use i.i.d. beta distributions to estimate the lower bound of the expected differences between observations, ultimately yielding an estimated lower bound for Britt's nu itself. This method may be faster than some methods of reordering rows or columns in some cases, although this is not always true.

Additional usage notes:

1. When x is a list of 2D double vectors (rather than a list of objects outputted from LDA), each row of each list element should be a single observation from a Dirichlet distribution (e.g., an LDA topic), and each column should be a category within that distribution (e.g., a word).

2. It is generally recommended that estimate_pairwise_alpha_from_joint be set to TRUE, which is the default setting. If concentration parameters must be estimated from x, and if all elements were generated via LDA from the same corpus or are otherwise assumed to have emerged from the same underlying distribution, then there is little reason to expect different sets of concentration parameters to be necessary across iterations. You should only consider setting estimate_pairwise_alpha_from_jo if different data sets that may not have come from the same underlying distribution were used to generate different elements of x. Doing so, however, may substantially slow the procedure, so it is generally not recommended unless essential.

3. If you are assessing the reliability of the allocations of words to topics different set of documents. In such cases, different_documents must be set to TRUE. For instance, if mydata1, mydata2, mydata3, mydata4, and mydata5 each contain 80 documents from a single data set, then you could run

```
cv1 <- LDA(mydata1, k=15)
cv2 <- LDA(mydata2, k=15)
cv3 <- LDA(mydata3, k=15)
cv4 <- LDA(mydata4, k=15)
cv5 <- LDA(mydata5, k=15)
cv <- list(cv1, cv2, cv3, cv4, cv5)
rel <- brittnu(cv, type="wt", shuffle=TRUE, different_documents=TRUE)
summary(rel)
```

to assess the word-topic allocation reliability for a 15-topic model. Any words that appear in some cross-validation iterations but not others will automatically be set to an allocation near 0 for any

iterations in which they are absent from the data set. If you are instead assessing the reliability of the allocations of topics to documents, then if `different_documents==TRUE`, any iteration in which a given document does not appear will simply be excluded from the reliability computation for that document. Thus, fewer cross-validation iterations will be used to assess reliability for each individual document. This will weaken the stability of the reliability computation. Therefore, unless you are specifically assessing whether your topic model is stable regardless of what subset of documents was used to generate it, it is advised that all documents be included in all cross-validation iterations.

4. When `shuffle==TRUE`, each element in `x` comprises a large number of topics, and/or `samples` is large, setting `lower_bound=TRUE` may sometimes be useful in order to eliminate the need to reorder a large number of topics in numerous bootstrapped samples. Additionally, for Dirichlet distributions with many categories, precise estimates of the concentration parameters may require excessive time, so whenever possible, a priori known concentration parameters should be provided via the `alpha` parameter rather than estimating them from the data. This is also useful to ensure the validity of Britt's nu. When this is not possible, consider setting `sampling_method="nonparametric"` in order to avoid the use of concentration parameters altogether. Alternatively, the procedure may be expedited by changing `convcrit` from its default value (0.00001) to a larger number. You may also wish to set `progress=TRUE` and `verbose=TRUE` to receive periodic progress updates and ensure that the computation is proceeding as expected.

5. When `shuffle==TRUE`, the topics in `x` are reordered. This is described in the matrix of reordered topics, which is provided as one of the elements of the list returned by brittnu and can be viewed using `summary()` on that object. Each row of this matrix indicates the manner in which the topics were reordered. For instance, if the first row is c(3,1,2), that means that for the first rater, the third topic was moved to the first position, the first topic was moved to the second position, and the second topic was moved to the third position.

Value

A list of class `brittnu_rel` containing seven elements: Britt's single-observation nu for all raters (as a numeric vector indicating the reliability for each observation), Britt's single-observation nu for pairs of raters (as a list containing lists containing numeric vectors indicating the reliability of each observation for each pair of raters, e.g., the second element of the first list contains the reliability for raters 1 and 2), Britt's multiple-observation nu for all raters (as a numeric value), Britt's multiple-observation nu for pairs of raters (as a list containing lists containing numeric values, e.g., the second element of the first list contains the reliability for raters 1 and 2), the matrix of reordered topics for each rater (if `shuffle==TRUE`), any warnings raised, and the value of the `type` argument

References

Britt, B. C. (under review). Interrater reliability for compositional, Euclidean, and Dirichlet Data.

Examples

```
#Example 1: LDA results
require(topicmodels)
data("AssociatedPress")
ap1 <-  LDA(AssociatedPress, k = 10)
ap2 <-  LDA(AssociatedPress, k = 10)
ap3 <-  LDA(AssociatedPress, k = 10)
ap_all <- list(ap1,ap2,ap3)
reliability_ap_td <- brittnu(ap_all, type="td", symmetric_alpha=TRUE,
                             shuffle=TRUE, samples=1000, verbose=TRUE)
reliability_ap_wt <- brittnu(ap_all, type="wt", symmetric_alpha=TRUE,
```

```
                              shuffle=TRUE, samples=1000, verbose=TRUE)
summary(reliability_ap_td)
summary(reliability_ap_td, element="all")
summary(reliability_ap_wt)
summary(reliability_ap_wt, element="all")

#Example 2: Manually inputted data with known concentration parameters
require(gtools)
alpha1 <- c(1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0)
alpha2 <- c(2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0)
alpha3 <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)
data_with_known_alpha1 <- rbind(gtools::rdirichlet(1,alpha1),
                                gtools::rdirichlet(1,alpha2),
                                gtools::rdirichlet(1,alpha3))
data_with_known_alpha2 <- rbind(gtools::rdirichlet(1,alpha1),
                                gtools::rdirichlet(1,alpha2),
                                gtools::rdirichlet(1,alpha3))
data_with_known_alpha3 <- rbind(gtools::rdirichlet(1,alpha1),
                                gtools::rdirichlet(1,alpha2),
                                gtools::rdirichlet(1,alpha3))
data_with_known_alpha4 <- rbind(gtools::rdirichlet(1,alpha1),
                                gtools::rdirichlet(1,alpha2),
                                gtools::rdirichlet(1,alpha3))
data_with_known_alpha5 <- rbind(gtools::rdirichlet(1,alpha1),
                                gtools::rdirichlet(1,alpha2),
                                gtools::rdirichlet(1,alpha3))
data_with_known_alpha <- list(data_with_known_alpha1, data_with_known_alpha2,
                              data_with_known_alpha3, data_with_known_alpha4,
                              data_with_known_alpha5)
reliability_known <- brittnu(data_with_known_alpha,
                             alpha=list(alpha1, alpha2, alpha3))
summary(reliability_known, element="all")
```

`clustering_determine_difference` *Distance Between Clusters*

Description

This helper function computes the distance between two clusters based on a specified metric.

Usage

```
clustering_determine_difference(
  x_obs,
  y_obs,
  alldiff = NA,
  x_restructured = NA,
  clustering_method = NA
)
```

Arguments

x_obs	A vector of indices representing the observations in the first cluster
y_obs	A vector of indices representing the observations in the second cluster
alldiff	A 2D array indicating the distance between each pair of observations across all raters
x_restructured	A 2D array containing the original data set, restructured to facilitate more straightforward references in this function
clustering_method	A string indicating the criterion that will be used to evaluate the distance between clusters; permitted values are "average", "minimum", "maximum", "median", "centroid", and "wald"

Value

A numeric value indicating the distance between clusters

---

compute_expected_joint_differences_lower_bound

Compute Expected Joint Differences (Lower Bound)

Description

This helper function estimates the lower bound of the expected differences among all raters.

Usage

```
compute_expected_joint_differences_lower_bound(x, shuffle_dimension, verbose)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
shuffle_dimension	A string indicating whether "rows" or "columns" were shuffled in the original data set
verbose	A boolean value indicating whether to provide progress updates

Value

A list of lists containing numeric vectors indicating the lower bound of the expected differences between all raters

compute_expected_pairwise_differences_lower_bound *Compute Expected Pairwise Differences (Lower Bound)*

Description

This helper function estimates the lower bound of the expected differences between pairs of raters.

Usage

```
compute_expected_pairwise_differences_lower_bound(
  x,
  shuffle_dimension,
  verbose
)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
shuffle_dimension	A string indicating whether "rows" or "columns" were shuffled in the original data set
verbose	A boolean value indicating whether to provide progress updates

Value

A list of lists containing numeric vectors indicating the lower bound of the expected differences between pairs of raters

compute_observed_differences *Compute Observed Differences*

Description

This helper function computes the differences between all raters and between all pairs of raters in a given real or bootstrapped data set.

Usage

```
compute_observed_differences(x)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns

Value

A two-element list indicating the observed differences between observations for all raters and for each pair of raters, respectively

---

`compute_single_expectation` *Compute Single Expectation*

Description

This helper function computes the expected squared difference between deviates from the same underlying Dirichlet distribution when the categories in each distribution are not allowed to be reordered.

Usage

```
compute_single_expectation(
  x,
  joint_alpha,
  pairwise_alpha,
  pairwise,
  samples = 1000,
  sampling_method = "nonparametric",
  verbose = FALSE
)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`joint_alpha`	A vector of concentration parameters representing all raters
`pairwise_alpha`	A list of lists of vectors of concentration parameters representing all pairs of raters
`pairwise`	A boolean value indicating whether pairwise differences between individual raters should be computed; if `FALSE`, pairwise differences will be ignored in order to reduce the time and memory complexity of the computation
`samples`	An integer value indicating how many samples to use to estimate expected differences
`sampling_method`	A string indicating whether bootstrapped sampling should be performed using a `"parametric"` or `"nonparametric"` approach
`verbose`	A boolean value indicating whether to provide progress updates

Value

A two-element list indicating the expected differences between observations for all raters and for each pair of raters, respectively

`dirichlet.mle` *Estimate Concentration Parameters*

Description

This helper function estimates the concentration parameters of the Dirichlet distribution underlying multiple observations, with no restrictions placed on the values of those concentration parameters. This function is heavily based on `dirichlet.mle`, with modifications to avoid potential singularities in the estimation procedure.

Usage

```
dirichlet.mle(
  x,
  weights = NULL,
  eps = 10^(-5),
  convcrit = 1e-05,
  maxit = 1000,
  oldfac = 0.3,
  progress = FALSE
)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`weights`	A numeric vector used to calibrate the initial estimates of concentration parameters
`eps`	A numeric value used as a tolerance parameter to prevent logarithms of zero
`convcrit`	A numeric value indicating the threshold for convergence used to estimate concentration parameters
`maxit`	A numeric value indicating the maximum number of iterations used to estimate concentration parameters
`oldfac`	A numeric value between 0 and 1 used as the convergence acceleration factor
`progress`	A boolean value indicating whether progress updates should be provided

Value

A list of the estimated concentration parameters, the sum of those estimated concentration parameters, and the ratio between each estimated concentration parameter and the sum of those parameters

`estimate_alpha_from_data` *Estimate Alpha from Data*

Description

This helper function estimates the expected concentration parameters of the Dirichlet distribution underlying multiple observations.

Usage

```
estimate_alpha_from_data(
  x,
  pairwise,
  estimate_pairwise_alpha_from_joint,
  symmetric_alpha = FALSE,
  convcrit = 1e-05,
  maxit = 1000,
  progress = FALSE
)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`pairwise`	A boolean value indicating whether pairwise differences between individual raters should be computed; if `FALSE`, pairwise differences will be ignored in order to reduce the time and memory complexity of the computation
`estimate_pairwise_alpha_from_joint`	A boolean value indicating, when estimating reliability for pairs of raters, whether each pair of raters should use the concentration parameters estimated across all raters (`TRUE`) or whether separate sets of concentration parameters should be estimated for each pair of raters (`FALSE`); this argument only has an effect when `alpha==NA`, and it is strongly suggested that the default value of `TRUE` be used for this argument when possible
`symmetric_alpha`	A boolean value indicating whether concentration parameters should be assumed to be unchanged between observations, which is common in many topic modeling procedures
`convcrit`	A numeric value indicating the threshold for convergence used to estimate concentration parameters
`maxit`	A numeric value indicating the maximum number of iterations used to estimate concentration parameters
`progress`	A boolean value indicating whether to provide progress updates

Value

A two-element list indicating the estimated concentration parameters for all raters and for each pair of raters, respectively

`estimate_single_expectation` *Estimate Single Expectation*

Description

This helper function uses bootstrapping to estimate the expected squared difference between observations from the same underlying distribution.

Usage

```
estimate_single_expectation(
  x,
  joint_alpha,
  pairwise_alpha,
  pairwise,
  shuffle = TRUE,
  shuffle_method = "rotational",
  shuffle_dimension = "rows",
  clustering_method = "average",
  slow_clustering = FALSE,
  samples = 1000,
  sampling_method = "nonparametric",
  lower_bound = TRUE,
  verbose = FALSE
)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`joint_alpha`	A vector of concentration parameters representing all raters
`pairwise_alpha`	A list of lists of vectors of concentration parameters representing all pairs of raters
`pairwise`	A boolean value indicating whether pairwise differences between individual raters should be computed; if `FALSE`, pairwise differences will be ignored in order to reduce the time and memory complexity of the computation
`shuffle`	A boolean value indicating whether or not the topics may have been shuffled into different sequences by each rater
`shuffle_method`	A string indicating what method (`"rotational"`, `"agglomerative"`, or `"divisive"`) will be used to reorder topics
`shuffle_dimension`	A string indicating whether `"rows"` or `"columns"` should be reordered; this argument only has an effect if `shuffle==TRUE` and `type==NA`
`clustering_method`	A string indicating the criterion that will be used to merge or divide clusters if `shuffle==TRUE` and either `shuffle_method=="agglomerative"` or `shuffle_method=="divisive"` permitted values are `"average"`, `"minimum"`, `"maximum"`, `"median"`, `"centroid"`, and `"wald"`

`slow_clustering`
: A boolean value indicating whether the distances between clusters should be recalculated after each individual observation is added to the new cluster (`TRUE`), which can improve the cohesiveness of the resulting cluster, or whether all observations should be added to the new cluster without recomputing the distances between clusters after every addition (`FALSE`); this argument only has an effect if `shuffle==TRUE` and `shuffle_method=="divisive"`

`samples`
: An integer value indicating how many samples to use to estimate expected differences

`sampling_method`
: A string indicating whether bootstrapped sampling should be performed using a `"parametric"` or `"nonparametric"` approach

`lower_bound`
: A boolean value indicating whether the lower bound of the expected differences (based on i.i.d. beta distributions) should be used rather than reordering Dirichlet distributions to estimate the exact difference in every sample when `shuffle==TRUE`

`verbose`
: A boolean value indicating whether to provide progress updates

Value

A two-element list indicating the expected differences between observations for all raters and for each pair of raters, respectively

---

euclidkrip *Euclidean Krippendorff's Alpha*

Description

This function computes Euclidean Krippendorff's alpha for data sets composed of compositional or Euclidean data, as described by Britt (under review).

Usage

```
euclidkrip(
  x,
  pairwise = TRUE,
  force_bootstrapping = FALSE,
  shuffle = FALSE,
  shuffle_method = "rotational",
  shuffle_dimension = "rows",
  clustering_method = "average",
  slow_clustering = FALSE,
  samples = 1000,
  lower_bound = FALSE,
  verbose = FALSE
)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
pairwise	A boolean value indicating whether pairwise reliability between individual raters should be computed; if FALSE, pairwise reliability will be ignored in order to reduce the time and memory complexity of the computation
force_bootstrapping	A boolean value indicating whether bootstrapping should be used to estimate expected differences rather than exactly computing them even if shuffle==FALSE
shuffle	A boolean value indicating whether or not the topics may have been shuffled into different sequences by each rater; if this is TRUE and shuffle_dimension="rows", then the observations (rows) corresponding to each rater will be reordered to achieve a (local) optimal fit, whereas if this is TRUE and shuffle_dimension="columns", the categories (columns) will be reordered instead
shuffle_method	A string indicating what method ("rotational", "agglomerative", or "divisive") will be used to reorder topics
shuffle_dimension	A string indicating whether "rows" or "columns" should be reordered; this argument only has an effect if shuffle==TRUE and type==NA
clustering_method	A string indicating the criterion that will be used to merge or divide clusters if shuffle==TRUE and either shuffle_method=="agglomerative" or shuffle_method=="divisive" permitted values are "average", "minimum", "maximum", "median", "centroid", and "wald"
slow_clustering	A boolean value indicating whether the distances between clusters should be recalculated after each individual observation is added to the new cluster (TRUE), which can improve the cohesiveness of the resulting cluster, or whether all observations should be added to the new cluster without recomputing the distances between clusters after every addition (FALSE); this argument only has an effect if shuffle==TRUE and shuffle_method=="divisive"
samples	An integer value indicating how many samples to use to estimate expected differences when shuffle==TRUE
lower_bound	A boolean value indicating whether the lower bound of the expected differences (based on i.i.d. beta distributions) should be used rather than shuffling Dirichlet distributions to estimate the exact difference in every sample when shuffle==TRUE
verbose	A boolean value indicating whether euclidkrip and its helper functions should provide progress updates beyond the estimation of concentration parameters

Details

One of the most important uses for Euclidean Krippendorff's alpha is to assess the reliability of the allocations of words to topics or of topics to documents via topic modeling techniques that do not necessarily adhere to a Dirichlet distribution. This includes, for instance, Dirichlet multinomial mixture models, which are commonly used for short text topic modeling.

Crucially, different topic modeling cross-validation iterations, which effectively represent distinct raters of the same data set, may result in the same topics appearing in a different sequence. As such, those topics, which may represent either the rows or the columns of the data set, often need to be reordered in order to yield an optimal fit.

In practice, it is generally infeasible to assess every possible combination of topics across all raters. As such, euclidkrip provides three methods of converging toward an optimal topic order for each rater. The default, `shuffle_method="rotational"`, takes the provided sequence and swaps pairs of categories until no further swaps would further improve the fit. `shuffle_method="agglomerative"` and `shuffle_method="divisive"` instead perform hierarchical cluster analyses of all topics, with the added restriction that two topics constructed by the same rater may not appear in the same cluster. Notably, the `"agglomerative"` option can take excessive time in some cases, while the `"divisive"` option forms clusters based on the macro-level dynamics of the data and may therefore yield weakly matched sets of individual topics compared to the other available options.

Regardless of whether the rows or columns of the data set must be reordered, when computing Euclidean Krippendorff's alpha, it is common to compute two separate coefficients in this family: Euclidean Krippendorff's alpha for all raters and for pairs of raters. Unlike Britt's nu, both of these Euclidean Krippendorff's alpha measures yield a single reliability score for all observations, as the formula used by Euclidean Krippendorff's alpha to compute the expected differences between raters does not produce separate results for each individual observation. However, if the reliability of each observation is desired, then nonparametric bootstrapping may be used to estimate those coefficients. This is automatically done when `shuffle==TRUE`, and it is also implemented when `force_bootstrapping==TRUE`. In either of those cases, the single-observation reliability values will be reported alongside the multiple-observation reliability values that euclidkrip already provides.

By default, euclidkrip provides reliability coefficients for all raters and for pairs of raters. To reduce the time and memory required, the pairwise versions may optionally be omitted from the computation by specifying `pairwise=FALSE`.

Additional usage notes:

1. `x` must be a list of 2D double vectors, and each row of each list element should be a single observation from a Dirichlet distribution (e.g., an LDA topic) while each column should represent a category within that distribution (e.g., a word).

2. When `shuffle==TRUE`, each element in `x` comprises a large number of topics, and/or `samples` is large, setting `lower_bound=TRUE` may sometimes be useful in order to eliminate the need to reorder a large number of topics in numerous bootstrapped samples. You may also wish to set `verbose=TRUE` to receive periodic progress updates and ensure that the computation is proceeding as expected.

3. When `shuffle==TRUE`, the topics in `x` are reordered. This is described in the matrix of reordered topics, which is provided as one of the elements of the list returned by euclidkrip and can be viewed using `summary()` on that object. Each row of this matrix indicates the manner in which the topics were reordered. For instance, if the first row is c(3,1,2), that means that for the first rater, the third topic was moved to the first position, the first topic was moved to the second position, and the second topic was moved to the third position.

Value

A list of class `euclidkrip_rel` containing six elements: Euclidean Krippendorff's alpha for each observation and all raters (as a numeric vector indicating the reliability for each observation), Euclidean Krippendorff's alpha for each observation and pairs of raters (as a list containing lists containing numeric vectors indicating the reliability of each observation for each pair of raters, e.g., the second element of the first list contains the reliability for raters 1 and 2), Euclidean Krippendorff's alpha for all observations and all raters (as a numeric value), Euclidean Krippendorff's alpha for all observations and pairs of raters (as a list containing lists containing numeric values, e.g., the second element of the first list contains the reliability for raters 1 and 2), the matrix of reordered topics for each rater (if `shuffle==TRUE`), and any warnings raised

References

Britt, B. C. (under review). Interrater reliability for compositional, Euclidean, and Dirichlet Data.

Examples

```
rater1 <- rbind(c(0.80,0.05,0.15), c(0.10,0.85,0.05),
                c(0.05,0.05,0.90), c(0.85,0.10,0.05))
rater2 <- rbind(c(0.10,0.85,0.05), c(0.10,0.10,0.80),
                c(0.85,0.05,0.10), c(0.15,0.80,0.05))
data <- list(rater1, rater2)
rel <- euclidkrip(data, shuffle=TRUE, shuffle_method="agglomerative",
                  shuffle_dimension="columns", clustering_method="average",
                  samples=500, verbose=TRUE)
summary(rel)
summary(rel, element="all")
```

---

generate_artificial_data_nonparametric

Generate Artificial Data (Nonparametric)

Description

This helper function uses nonparametric bootstrapping to generate an artificial data set from a provided data set

Usage

```
generate_artificial_data_nonparametric(x, shuffle, shuffle_dimension)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`shuffle`	A boolean value indicating whether or not the topics may have been shuffled into different sequences by each rater
`shuffle_dimension`	A string indicating whether `"rows"` or `"columns"` were shuffled in the original data set; this argument only has an effect if `shuffle==TRUE`

Value

A list of nonparametrically bootstrapped data such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns

generate_artificial_data_parametric *Generate Artificial Data (Parametric)*

Description

This helper function uses parametric bootstrapping to generate an artificial data set from a provided data set

Usage

```
generate_artificial_data_parametric(x, pairwise_alpha)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
pairwise_alpha	A list of lists of vectors of concentration parameters representing all pairs of raters

Value

A list of parametrically bootstrapped data such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns

label_assignments *Label Assignments Matrix*

Description

This helper function adds labels to the assignments matrix based on whether the rows and columns represent documents and topics, topics and words, or something else, in order to facilitate more informative printing.

Usage

```
label_assignments(assignments, type = NA)
```

Arguments

assignments	A matrix indicating the relationship between each reordered topic and its original position in the data
type	A string indicating the type of reliability being assessed, with permitted values of "wt" (word-topic reliability), "td" (topic-document reliability), and NA

Value

A matrix indicating the relationship between each reordered topic and its original position in the data, with the addition of row and column labels

print.brittnu_rel *Print (brittnu)*

Description

This helper method calls `summary()` when the user attempts to print an object of class `brittnu_rel` outputted from brittnu.

Usage

```
## S3 method for class 'brittnu_rel'
print(x, element = NA, ...)
```

Arguments

`x`	An object of class `brittnu_rel` outputted from brittnu
`element`	The specific reliability coefficient to be printed; permitted values are `"singleomnibus"`, `"singlepairwise"`, `"multipleomnibus"`, `"multiplepairwise"`, `"all"`, and `NA`
`...`	Other arguments inherited from the generic `print` function

print.euclidkrip_rel *Print (euclidkrip)*

Description

This helper method calls `summary()` when the user attempts to print an object of class `euclidkrip_rel` outputted from euclidkrip.

Usage

```
## S3 method for class 'euclidkrip_rel'
print(x, element = NA, ...)
```

Arguments

`x`	An object of class `euclidkrip_rel` outputted from euclidkrip
`element`	The specific reliability coefficient to be printed; permitted values are `"singleomnibus"`, `"singlepairwise"`, `"multipleomnibus"`, `"multiplepairwise"`, `"all"`, and `NA`
`...`	Other arguments inherited from the generic `print` function

rdirichlet *Generate Dirichlet Observations*

Description

This helper function generates a set of observations from a Dirichlet distribution with underlying concentration parameters equal to alpha. This function is heavily based on rdirichlet, with modifications to prevent concentration parameters equal to 0.

Usage

```
rdirichlet(n, alpha, zero2 = (10^(-255)))
```

Arguments

n	A numeric value indicating the number of observations to be generated
alpha	A numeric vector indicating the concentration parameters of the Dirichlet distribution from which observations should be generated
zero2	A numeric value representing the minimum allocation permitted for any given Dirichlet category in order to prevent errors

Value

A numeric vector with n Dirichlet observations

restricted_agglomerative_clustering *Restricted Agglomerative Clustering*

Description

This helper function performs agglomerative hierarchical cluster analysis in order to minimize the sum of squared differences between raters.

Usage

```
restricted_agglomerative_clustering(
  x,
  verbose = FALSE,
  clustering_method = "average"
)
```

Arguments

x	A list such that each element is a 2D double vector, with the topics to be reordered representing the rows of each vector
verbose	A boolean value indicating whether to provide progress updates
clustering_method	A string indicating the criterion that will be used to merge or divide clusters; permitted values are "average", "minimum", "maximum", "median", "centroid", and "wald"

Details

This procedure conducts a cluster analysis in order to group topics from each rater into clusters indicating optimal matches between those topics. This effectively serves to reorder the topics constructed by each such that, to the extent possible, each individual topic is matched with its best-fitting counterparts constructed by each other rater before reliability is assessed.

This cluster analysis uses an additional constraint: topics from the same rater can never be combined into the same cluster. This ensures that at the end of the process, the number of clusters will be equal to the number of topics constructed by each rater, and each rater will have yielded one topic belonging to each cluster. If two clusters would be combined such that this rule would be violated, then immediately afterward, all offending topics are removed from the newly merged cluster and each one is treated as an independent cluster to be later merged into other clusters.

Additionally, to prevent infinite loops (e.g., the same topic is repeatedly added to a larger cluster and removed afterward in accordance with the preceding procedure), two additional requirements are imposed in order to merge any pair of clusters. First, when considering any potential cluster merge operation, if executing the merge would in an identical set of clusters as any prior step, that operation is not performed, and the procedure skips to the next pair of clusters to be considered for merging. Second, when considering any potential cluster merge operation, the procedure compares the set of raters from which all topics in each of the two clusters were drawn, and if either one is a perfect subset of the other (e.g., the topics in cluster 1 came from raters 3 and 6, while the topics in cluster 2 were created by raters 1, 2, 3, 4, 6, and 8), then we cannot merge the two clusters, as all observations in one cluster would just be removed from the resulting cluster afterward. Whenever such a perfect subset is observed via this diagnostic, the procedure skips to the next pair of clusters to be considered for merging.

Notably, although infinite loops are avoided, this function still tends to get stuck in long-lasting loops. When this happens, either rotational_shuffling or restricted_divisive_clustering may be a better choice.

Value

A two-element list containing the reordered data set and a matrix indicating the relationship between each reordered topic and its original position in the data, respectively

---

`restricted_divisive_clustering`
Restricted Divisive Clustering

Description

This helper function performs divisive hierarchical cluster analysis in order to minimize the sum of squared differences between raters.

Usage

```
restricted_divisive_clustering(
  x,
  verbose = FALSE,
  clustering_method = "average",
  slow_clustering = FALSE
)
```

Arguments

x	A list such that each element is a 2D double vector, with the topics to be reordered representing the rows of each vector
verbose	A boolean value indicating whether to provide progress updates
clustering_method	A string indicating the criterion that will be used to merge or divide clusters; permitted values are "average", "minimum", "maximum", "median", "centroid", and "wald"
slow_clustering	A boolean value indicating whether the distances between clusters should be recalculated after each individual observation is added to the new cluster (TRUE), which can improve the cohesiveness of the resulting cluster, or whether all observations should be added to the new cluster without recomputing the distances between clusters after every addition (FALSE)

Details

This procedure conducts a cluster analysis in order to group topics from each rater into clusters indicating optimal matches between those topics. This effectively serves to reorder the topics constructed by each such that, to the extent possible, each individual topic is matched with its best-fitting counterparts constructed by each other rater before reliability is assessed.

This cluster analysis uses an additional constraint: all clusters resulting from dividing an existing cluster must contain a number of observations that is a multiple of the number of topics constructed by each rater, and all raters must be equally represented in each of those resulting clusters. This ensures that at the end of the process, the number of clusters will be equal to the number of topics constructed by each rater, and each rater will have yielded one topic belonging to each cluster.

However, this approach sometimes creates clusters based on macro-level dynamics rather than micro-level differences between topics, ultimately yielding weakly matched sets of categories compared with other approaches. In those cases, either rotational_shuffling or restricted_agglomerative_clustering may yield superior results.

Value

A two-element list containing the reordered data set and a matrix indicating the relationship between each reordered topic and its original position in the data, respectively

---

rotational_shuffling *Rotational Shuffling*

Description

This helper function reorders the rows of ratings from multiple raters in order to minimize the sum of squared differences between them.

Usage

```
rotational_shuffling(x, verbose = FALSE, swap_tol = (10^(-12)))
```

Arguments

x	A list such that each element is a 2D double vector, with the topics to be reordered representing the rows of each vector
verbose	A boolean value indicating whether to provide progress updates
swap_tol	A numeric value representing the minimum threshold by which the sum of squared differences must be improved in order to execute a potential swap of two rows; this is necessary in order to prevent infinite loops due to minuscule rounding errors

Value

A two-element list containing the reordered data set and a matrix indicating the relationship between each reordered topic and its original position in the data, respectively

---

shuffle_distributions *Shuffle Distributions*

Description

This helper function reorders the categories in multiple Dirichlet deviates in order to minimize the sum of squared differences between them.

Usage

```
shuffle_distributions(
  x,
  verbose = FALSE,
  shuffle_method = "rotational",
  clustering_method = "average",
  slow_clustering = FALSE,
  shuffle_dimension = "rows"
)
```

Arguments

x	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
verbose	A boolean value indicating whether to provide progress updates
shuffle_method	A string indicating what method (`"rotational"`, `"agglomerative"`, or `"divisive"`) will be used to reorder topics
clustering_method	A string indicating the criterion that will be used to merge or divide clusters if `shuffle_method=="agglomerative"` or `shuffle_method=="divisive"`; permitted values are `"average"`, `"minimum"`, `"maximum"`, `"median"`, `"centroid"`, and `"wald"`
slow_clustering	A boolean value indicating whether the distances between clusters should be recalculated after each individual observation is added to the new cluster (`TRUE`), which can improve the cohesiveness of the resulting cluster, or whether all observations should be added to the new cluster without recomputing the distances

between clusters after every addition (FALSE); this argument only has an effect if shuffle_method=="divisive"

shuffle_dimension
: A string indicating whether "rows" or "columns" should be reordered

Value

A two-element list containing the reordered data set and a matrix indicating the relationship between each reordered topic and its original position in the data, respectively

---

sirt_digamma1 *Sirt Digamma 1*

Description

This helper function estimates the derivative of the digamma function. This function is directly drawn from sirt::sirt_digamma1, as published at https://github.com/cran/sirt/blob/master/R/sirt_digamma1.R. This function does not appear to be included in current releases of sirt and therefore cannot be directly loaded from that package.

Usage

```
sirt_digamma1(x, h = 0.001)
```

Arguments

x	A vector of concentration parameters
h	A numeric value used to define a pair of nearby observations

Value

A list of the estimated concentration parameters, the sum of those estimated concentration parameters, and the ratio between each estimated concentration parameter and the sum of those parameters

---

squarediff *Sum of Squared Differences*

Description

This helper function computes the sum of squared differences between two vectors.

Usage

```
squarediff(x, y)
```

Arguments

x	A numeric vector
y	A numeric vector

Value

A numeric value representing the sum of the squared differences between x and y

summary.brittnu_rel *Summary (brittnu)*

Description

This method prints a summary of the Britt's nu reliability values using an object of class `brittnu_rel` outputted from brittnu.

Usage

```
## S3 method for class 'brittnu_rel'
summary(object, element = NA, ...)
```

Arguments

`object`	An object of class `brittnu_rel` outputted from brittnu
`element`	The specific reliability coefficient to be printed; permitted values are `"singleomnibus"`, `"singlepairwise"`, `"multipleomnibus"`, `"multiplepairwise"`, `"all"`, and `NA`
`...`	Other arguments inherited from the generic `summary` function

summary.euclidkrip_rel *Summary (euclidkrip)*

Description

This method prints a summary of the Euclidean Krippendorff's alpha reliability values using an object of class `euclidkrip_rel` outputted from euclidkrip.

Usage

```
## S3 method for class 'euclidkrip_rel'
summary(object, element = NA, ...)
```

Arguments

`object`	An object of class `euclidkrip_rel` outputted from euclidkrip
`element`	The specific reliability coefficient to be printed; permitted values are `"singleomnibus"`, `"singlepairwise"`, `"multipleomnibus"`, `"multiplepairwise"`, `"all"`, and `NA`
`...`	Other arguments inherited from the generic `summary` function

`symmetric.dirichlet.mle` *Estimate Symmetric Concentration Parameters*

Description

This helper function estimates the concentration parameters of the Dirichlet distribution underlying multiple deviates, assuming that those concentration parameters are all equal. This function is heavily based on `dirichlet.mle`, with modifications to restrict all concentration parameters to be equal and to avoid potential singularities in the estimation procedure.

Usage

```
symmetric.dirichlet.mle(
  x,
  weights = NULL,
  eps = 10^(-5),
  convcrit = 1e-05,
  maxit = 1000,
  oldfac = 0.3,
  progress = FALSE
)
```

Arguments

`x`	A list such that each element is a 2D double vector with observations as the rows and each category within each observation as the columns
`weights`	A numeric vector used to calibrate the initial estimates of concentration parameters
`eps`	A numeric value used as a tolerance parameter to prevent logarithms of zero
`convcrit`	A numeric value indicating the threshold for convergence used to estimate concentration parameters
`maxit`	A numeric value indicating the maximum number of iterations used to estimate concentration parameters
`oldfac`	A numeric value between 0 and 1 used as the convergence acceleration factor
`progress`	A boolean value indicating whether progress updates should be provided

Value

A list of the estimated concentration parameters, the sum of those estimated concentration parameters, and the ratio between each estimated concentration parameter and the sum of those parameters

Index