Package 'RankAggSIgFUR'

January 20, 2025

Title Polynomially Bounded Rank Aggregation under Kemeny's Axiomatic Approach

Version 1.0.0

Description Polynomially bounded algorithms to aggregate complete rankings under Kemeny's axiomatic framework. 'RankAggSIgFUR' (pronounced as rank-agg-cipher) contains two heuristics algorithms: FUR and SIgFUR. For details, please see Badal and Das (2018) <doi:10.1016/j.cor.2018.06.007>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

LazyData true

Imports Rfast, combinat, data.table, plyr

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

URL https://github.com/prakashvs613/RankAggSIgFUR

BugReports https://github.com/prakashvs613/RankAggSIgFUR/issues

Author Hannah Parker [aut],
Rakhi Singh [cre, aut] (<https://orcid.org/0000-0003-3469-295X>),
Prakash Singh Badal [aut] (<https://orcid.org/0000-0003-4097-8444>)

Maintainer Rakhi Singh <agrakhi@gmail.com>

Repository CRAN

Date/Publication 2023-06-19 03:40:09 UTC

Contents

`data100x15` *Simulated 100 × 15 Data*

Description

Data of 100 objects and 15 attributes, in which the first column contains the object names and each subsequent column is a complete ranking of the 100 objects. The included 50 × 15 and 400 × 15 datasets were generated from this dataset (see `data50x15` and `data400x15`).

Usage

```
data(data100x15)
```

Format

A data frame with 100 rows and 16 columns:

Object object name

Ranking 1 ranking on the first attribute

Ranking 2 ranking on the second attribute

Ranking 3 ranking on the third attribute

Ranking 4 ranking on the fourth attribute

Ranking 5 ranking on the fifth attribute

Ranking 6 ranking on the sixth attribute

Ranking 7 ranking on the seventh attribute

Ranking 8 ranking on the eigth attribute

Ranking 9 ranking on the ninth attribute

Ranking 10 ranking on the tenth attribute

Ranking 11 ranking on the eleventh attribute

Ranking 12 ranking on the twelfth attribute

Ranking 13 ranking on the thirteenth attribute

Ranking 14 ranking on the fourteenth attribute

Ranking 15 ranking on the fifteenth attribute

Source

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

Examples

```
data(data100x15)
input_rkgs <- t(as.matrix(data100x15[, -1]))
obj_names <- data100x15[,1]

# Determine the mean seed ranking
mean_seed(input_rkgs)
```

---

data240x4 *PrefLib 240 × 4 Data*

Description

Data of 240 cities across the globe ranked on four criteria from the ED-00015-001.soc dataset in the PrefLib repository. The first column contains the object names and each subsequent column is a complete ranking of the 240 objects with no ties) .

Usage

```
data(data240x4)
```

Format

A data frame with 240 rows and 5 columns:

Object object name

Ranking 1 ranking on the first criterion

Ranking 2 ranking on the second criterion

Ranking 3 ranking on the third criterion

Ranking 4 ranking on the fourth criterion

Source

https://www.preflib.org/

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

Mattei, N., & Walsh, T. (2013, November). Preflib: A library for preferences https://www.preflib.org/. In International conference on algorithmic decision theory (pp. 259-270). Springer, Berlin, Heidelberg.

Examples

```
data(data240x4)
input_rkgs <- t(as.matrix(data240x4[, -1]))
obj_names <- data240x4[,1]

# Determine the mean seed ranking
mean_seed(input_rkgs)
```

---

data400x15 *Simulated 400 × 15 Data*

Description

Data of 400 objects and 15 attributes in which the first column contains the object names and each subsequent column is a complete ranking of the 400 objects. This data set is generated from the 100 × 15 dataset (see data50x15) by adding 100 to the ranks of the objects numbered 1 through 100 to get the ranks of objects numbered 101 through 200. Similarly, by adding 200 to obtain ranking 201 through 300, and by adding 300 to obtain ranking 301 through 400.

Usage

```
data(data400x15)
```

Format

A data frame with 400 rows and 16 columns:

Objects object name

Ranking 1 ranking on the first attribute

Ranking 2 ranking on the second attribute

Ranking 3 ranking on the third attribute

Ranking 4 ranking on the fourth attribute

Ranking 5 ranking on the fifth attribute

Ranking 6 ranking on the sixth attribute

Ranking 7 ranking on the seventh attribute

Ranking 8 ranking on the eigth attribute

Ranking 9 ranking on the ninth attribute

Ranking 10 ranking on the tenth attribute

Ranking 11 ranking on the eleventh attribute

Ranking 12 ranking on the twelfth attribute

Ranking 13 ranking on the thirteenth attribute

Ranking 14 ranking on the fourteenth attribute

Ranking 15 ranking on the fifteenth attribute

Source

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

Examples

```
data(data400x15)
input_rkgs <- t(as.matrix(data400x15[, -1]))
obj_names <- data400x15[,1]

# Determine the mean seed ranking
mean_seed(input_rkgs)
```

---

data50x15 *Simulated 50 × 15 Data*

Description

Data of 50 objects and 15 attributes, which were randomly generated from the 100 × 15 simulated dataset (see data100x15). The first column contains the object names and each subsequent column is a complete ranking of the 50 objects.

Usage

```
data(data50x15)
```

Format

A data frame with 50 rows and 16 columns:

Object object name

Ranking 1 ranking on the first attribute

Ranking 2 ranking on the second attribute

Ranking 3 ranking on the third attribute

Ranking 4 ranking on the fourth attribute

Ranking 5 ranking on the fifth attribute

Ranking 6 ranking on the sixth attribute

Ranking 7 ranking on the seventh attribute

Ranking 8 ranking on the eigth attribute

Ranking 9 ranking on the ninth attribute

Ranking 10 ranking on the tenth attribute

Ranking 11 ranking on the eleventh attribute

Ranking 12 ranking on the twelfth attribute

Ranking 13 ranking on the thirteenth attribute

Ranking 14 ranking on the fourteenth attribute

Ranking 15 ranking on the fifteenth attribute

Source

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

Examples

```
data(data50x15)
input_rkgs <- t(as.matrix(data50x15[, -1]))
obj_names <- data50x15[,1]

# Determine the mean seed ranking
mean_seed(input_rkgs)
```

---

fur *FUR*

Description

FUR is a heuristic algorithm to obtain a consensus ranking. It contains three branches – Fixed, Update, and Range – that use *Subiterative Convergence* and *Greedy Algorithm* iteratively. See 'Details' for more information on each branch.

Usage

```
fur(
  input_rkgs,
  subit_len_list,
  search_radius,
  seed_rkg = c(),
  objNames = c(),
  wt = c()
)
```

Arguments

`input_rkgs`	a `n` by `k` matrix of `k` rankings of `n` objects, where each column is a complete ranking.
`subit_len_list`	a vector containing positive integer(s) for the subiteration lengths to *Subiterative Convergence*. Recommended values are between 2 and 8. Smaller subiteration lengths result in shorter run-time.
`search_radius`	a positive integer for the maximum change in the rank of each object in the *Greedy Algorithm*. The default value of `0` considers all possible rank changes for each object. It is recommended to use a search radius of less than or equal to $\min(30, \lfloor n/2 \rfloor)$.
`seed_rkg`	a vector of length `n` with an initial ranking to begin FUR. If the default value of an empty vector is used, then the `mean seed ranking` is adopted as the initial ranking to FUR.

`objNames`	a `n`-length vector containing object names. An optional parameter.
`wt`	a `k`-length vector containing weights for each judge or attribute. An optional parameter.

Details

The Fixed branch applies *Subiterative Convergence* using one subiteration length from `subit_len_list` at a time.

The Update branch executes *Subiterative Convergence* using the first subiteration length in `subit_len_list`, and then uses its output in the next call to *Subiterative Convergence* with the next subiteration length in the list. This process repeats until `subit_len_list` is exhausted.

The Range branch calls *Subiterative Convergence* on all subiteration lengths in `subit_len_list` and only retains the best ranking among these separate calls.

The output from the *Subiterative Convergence* calls are fed into the *Greedy Algorithm* as its seed ranking, and the FUR algorithm is terminated when the input to the *Greedy Algorithm* converges to the output and all branches have been executed at least once.

Value

A list containing the consensus ranking (expressed as ordering), total Kemeny distance, and average tau correlation coefficient corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

See Also

`mean_seed`, `subit_convergence`, `rap_greedy_alg`, `sigfur`

Examples

```
## One subiteration length
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
subit_len_list <- 2
search_radius <- 1
fur(input_rkgs, subit_len_list, search_radius) # Determined the consensus ranking, total Kemeny
                                               # distance, and average tau correlation coefficient

## Multiple subiteration lengths
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
subit_len_list <- c(2,3)
search_radius <- 1
fur(input_rkgs, subit_len_list, search_radius)

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
```

```
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
## Multiple subiteration lengths
wt = c(1,1,0,1,1)
subit_len_list <- c(2,3)
search_radius <- 1
fur(input_rkgs, subit_len_list, search_radius,wt=wt)

## Using five input rankings with five objects with prepare_data to
## automatically prepare the weight vector
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
out = prepare_data(input_rkgs)
input_rkgs = out$input_rkgs
wt = out$wt
subit_len_list <- c(2,3)
search_radius <- 1
fur(input_rkgs, subit_len_list, search_radius,wt=wt)

## Included dataset of 15 input rankings of 50 objects
data(data50x15)
input_rkgs <- as.matrix(data50x15[, -1])
subit_len_list <- c(2, 3)
search_radius <- 1
fur(input_rkgs, subit_len_list, search_radius)
```

---

mean_seed *Mean Seed Ranking*

Description

Determine the *mean seed ranking* of the given input rankings. The average rank of an object is the sum of its various rankings from each input ranking divided by the total number of rankings. The mean seed ranking is formed by ranking the objects based on their average ranks, and ties are broken by ranking the first tied object with a higher rank.

Usage

```
mean_seed(input_rkgs, wt = c())
```

Arguments

input_rkgs	a k by n matrix of k rankings of n objects, where each row is a complete ranking. Note that this is a transpose of matrix used for functions like fur, sigfur, rap_greedy_alg, and subit_convergence.
wt	a k-length vector containing weights for each judge or attribute. An optional parameter.

Value

A vector containing the mean seed ranking of the input rankings.

See Also

rank, subit_convergence, fur, sigfur

Examples

```
## Four input rankings of five objects
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
mean_seed(t(input_rkgs)) # Found the mean seed ranking

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
wt = c(1,1,0,1,1)
mean_seed(t(input_rkgs),wt=wt) # Found the mean seed ranking

## Included dataset of 15 input rankings of 50 objects
data(data50x15)
input_rkgs <- t(as.matrix(data50x15[, -1]))
mean_seed(input_rkgs)
```

---

mod_kemeny *Modified Kemeny Rank Aggregation*

Description

Modified Kemeny algorithm determines the consensus ranking of n objects using the set of all possible rankings compared to the input rankings. The algorithm is based on Kemeny's axiomatic approach of minimizing the total Kemeny distance from the input rankings. In case of multiple rankings with minimum total Kemeny distance, the consensus ranking is determined using two additional criteria. See 'Details' for additional criteria. The method involves n! comparisons. Hence, it works best on a set of rankings with a small number of objects.

Usage

```
mod_kemeny(input_rkgs, universe_rkgs, obj_pairs, wt)
```

Arguments

- `input_rkgs` a k by n matrix of k rankings of n objects, where each row is a complete ranking. Note that this is a transpose of matrix used for functions like `fur`, `sigfur`, `rap_greedy_alg`, and `subit_convergence`.
- `universe_rkgs` a matrix containing all possible permutations of ranking n objects. Each row in this matrix represents one permuted ranking.
- `obj_pairs` a 2 by n choose 2 matrix of all combinations of object pairs of n objects, where each column contains a pair of object indices.
- `wt` a k-length vector containing weights for each judge or attribute. An optional parameter.

Details

Under Kemeny's axiomatic approach, rankings with minimum total Kemeny distance are considered equally optimal. Modified Kemeny attempts to break the tie among such rankings by imposing two additional criteria on the basis of minimizing (a) the maximum and (b) the variance of individual Kemeny distances, applied sequentially.

Value

A list containing the consensus ranking (expressed as ordering), total Kemeny distance, and average tau correlation coefficient corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

Examples

```
## Consensus ranking from four rankings of five objects
n <- 5
input_rkgs <- matrix(c(3, 2, 5, 1, 2, 3, 1, 2, 5, 1, 3, 4, 4, 5, 4, 5, 1, 4, 2, 3), ncol = n)
uni_rkgs <- matrix(unlist(combinat::permn(c(1:n))), byrow = TRUE, ncol = n)
obj_pairs <- combinat::combn(1:n,2, simplify=TRUE)
wt <- rep(1,nrow(input_rkgs))
mod_kemeny(input_rkgs, uni_rkgs, obj_pairs,wt=wt) # Computed consensus ranking,
                                                  # total Kemeny distance,
                                            #and average tau correlation coefficient
```

prepare_data *Preparing Data*

Description

Prepares the given data for rank aggregation functions. The function returns a matrix of input rankings and a vector indicating weights of the ranking for each judge. Useful when scores need to be converted to rankings. Also helpful in reducing the size of the problem for large p, especially when p > n!.

Usage

```
prepare_data(df, HighertheBetter = 0)
```

Arguments

df	a n by p matrix or dataframe of scores of n objects given by p judges. Each column corresponds to a different judge.
HighertheBetter	an integer with 1 indicating that the higher values in the input correspond to the better rank. An optional parameter. Default value is 0, i.e., the lower the score the better the rank (e.g., score of 1 is the topmost rank).

Value

A list containing a matrix of input rankings (named input_rkgs) and a weight vector corresponding to weights for each judge (named wt). These two objects are used as inputs to subit_convergence, rap_greedy_alg, fur, and sigfur.

See Also

subit_convergence, rap_greedy_alg, fur, sigfur

Examples

```
## Five input rankings with five objects
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
out = prepare_data(input_rkgs)
input_rkgs = out$input_rkgs
wt = out$wt

## Five input rankings with five objects
## testing the higher the better
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
input_rkgs = input_rkgs*2+input_rkgs #artificially create a score matrix
# Testing the higher the better rank
```

```
out = prepare_data(input_rkgs, HighertheBetter = 1)
input_rkgs = out$input_rkgs
wt = out$wt
```

---

rap_greedy_alg — *Greedy Algorithm for Rank Aggregation*

Description

Greedy Algorithm is a heuristic method that hunts for improved rankings by moving one object at a time (up or down). In case an object's movement results in an improved ranking, the next object is moved with respect to this improved ranking. The process is repeated until all objects are considered once.

Usage

```
rap_greedy_alg(
  seed_rkg,
  input_rkgs,
  search_radius = 0,
  objNames = c(),
  wt = c()
)
```

Arguments

seed_rkg	an initial ranking to begin the algorithm. The algorithm is often used in conjunction with *Subiterative Convergence*.
input_rkgs	a n by k matrix of k rankings of n objects, where each column is a complete ranking.
search_radius	a positive integer for the maximum change in the rank of each object. The default value of 0 considers all possible rank changes for each object. Recommended value of search radius is less than or equal to $\min(30, \lfloor n/2 \rfloor)$.
objNames	a n-length vector containing object names. An optional parameter.
wt	a k-length vector containing weights for each judge or attribute. An optional parameter.

Value

A list containing the consensus ranking (expressed as ordering), total Kemeny distance, and average tau correlation coefficient corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

See Also

subit_convergence, fur, sigfur

Examples

```
## Four input rankings of five objects
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
mean_seed_rkg <- mean_seed(t(input_rkgs))
rap_greedy_alg(mean_seed_rkg, input_rkgs, search_radius = 0) # Determined the consensus ranking,
                                                     # total Kemeny distance, and average
                                                           # tau correlation coefficient

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
wt = c(1,1,0,1,1)
mean_seed_rkg <- mean_seed(t(input_rkgs),wt=wt)
rap_greedy_alg(mean_seed_rkg, input_rkgs, search_radius = 0,wt=wt) # Determined the
#consensus ranking,  total Kemeny distance, and average tau correlation coefficient


## Using five input rankings with five objects with prepare_data to
## automatically prepare the weight vector
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
out = prepare_data(input_rkgs)
input_rkgs = out$input_rkgs
wt = out$wt
mean_seed_rkg <- mean_seed(t(input_rkgs),wt=wt)
rap_greedy_alg(mean_seed_rkg, input_rkgs, search_radius = 0,wt=wt) # Determined the
#consensus ranking,  total Kemeny distance, and average tau correlation coefficient

## Included dataset of 15 input rankings of 50 objects
data(data50x15)
input_rkgs <- as.matrix(data50x15[, -1])
mean_seed_rkg <- mean_seed(t(input_rkgs)) # Use the mean seed ranking as the seed ranking
rap_greedy_alg(mean_seed_rkg, input_rkgs, search_radius = 1)
```

---

seed_based_iteration *Seed-Based Iteration*

Description

Seed-Based Iteration is a heuristic-based seed generation used in *SIgFUR* to iteratively perturb the ranking to improve the consensus ranking.

Usage

```
seed_based_iteration(eta, omega, input_rkgs, wt = c())
```

Arguments

eta	a subiteration length for intermittent *Subiterative Convergence*. The recommended values are between 2 and 8. Smaller subiteration lengths result in shorter run-time.
omega	a positive integer for the number of repetitions of perturbing the seed ranking. An omega value of 1 corresponds to a single application of *Subiterative Convergence*.
input_rkgs	a k by n matrix of k rankings of n objects, where each row is a complete ranking. Note that this is a transpose of matrix used for functions like fur, sigfur, rap_greedy_alg, and subit_convergence.
wt	a k-length vector containing weights for each judge or attribute. An optional parameter.

Value

A list containing the consensus ranking (expressed as ordering) and total Kemeny distance corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

See Also

sigfur, subit_convergence, mean_seed

Examples

```
## Four input rankings of five objects
eta <- 2
omega <- 10
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
seed_based_iteration(eta, omega, t(input_rkgs)) # Determined seed-based iterations

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
eta <- 2
omega <- 10
wt = c(1,1,0,1,1)
seed_based_iteration(eta, omega, t(input_rkgs), wt=wt) # Determined seed-based iterations
```

```
## Included dataset of 15 input rankings of 50 objects
eta <- 3
omega <- 5
data(data50x15)
input_rkgs <- as.matrix(data50x15[, -1])
seed_based_iteration(eta, omega, t(input_rkgs)) # Determined seed-based iterations
```

---

sigfur *SIgFUR*

Description

SIgFUR applies *Seed-Based Iteration*, *Greedy Algorithm*, and *FUR* in sequence for each element of `subit_len_list_sbi`. The *mean seed ranking* is used as the input to *Seed-Based Iteration*. The best of all output rankings from *FUR* is considered as the consensus ranking.

Usage

```
sigfur(
  input_rkgs,
  subit_len_list_sbi,
  omega_sbi,
  subit_len_list_fur,
  search_radius,
  objNames = c(),
  wt = c()
)
```

Arguments

`input_rkgs` a `n` by `k` matrix of `k` rankings of `n` objects, where each column is a complete ranking.

`subit_len_list_sbi` a vector containing positive integer(s) for the subiteration lengths to *Seed-Based Iteration*. Recommended values are between 2 and 8. Smaller subiteration lengths result in shorter run-time.

`omega_sbi` a positive integer for the number of repetitions of perturbing the seed ranking in *Seed-Based Iteration*. An `omega_sbi` value of 1 corresponds to a single application of *Subiterative Convergence*.

`subit_len_list_fur` a vector containing positive integer(s) for the subiteration lengths to *FUR*.

`search_radius` a positive integer for the maximum change in the rank of each object in the *Greedy Algorithm* and *FUR*. The default value of `0` considers all possible rank changes for each object. It is recommended to use a search radius of less than or equal to $\min(30, \lfloor n/2 \rfloor)$.

objNames	a n-length vector containing object names. An optional parameter.
wt	a k-length vector containing weights for each judge or attribute. An optional parameter.

Value

A list containing the consensus ranking (expressed as ordering), total Kemeny distance, and average tau correlation coefficient corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

See Also

seed_based_iteration, rap_greedy_alg, fur, mean_seed

Examples

```
## Four input rankings of five objects
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
subit_len_list_sbi <- c(2:3)
omega_sbi <- 10
subit_len_list_fur <- c(2:3)
search_radius <- 1
sigfur(input_rkgs, subit_len_list_sbi, omega_sbi, subit_len_list_fur, search_radius)
# Determined the consensus ranking, total Kemeny distance, and average tau correlation coefficient

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
subit_len_list_sbi <- c(2:3)
omega_sbi <- 10
subit_len_list_fur <- c(2:3)
search_radius <- 1
wt = c(1,1,0,1,1)
sigfur(input_rkgs, subit_len_list_sbi, omega_sbi, subit_len_list_fur, search_radius, wt=wt)
# Determined the consensus ranking, total Kemeny distance, and average tau correlation coefficient

## Using five input rankings with five objects with prepare_data to
## automatically prepare the weight vector
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
out = prepare_data(input_rkgs)
input_rkgs = out$input_rkgs
wt = out$wt
subit_len_list_sbi <- c(2:3)
omega_sbi <- 10
```

```
subit_len_list_fur <- c(2:3)
search_radius <- 1
sigfur(input_rkgs, subit_len_list_sbi, omega_sbi, subit_len_list_fur, search_radius, wt=wt)
# Determined the consensus ranking, total Kemeny distance, and average tau correlation coefficient

## Included dataset of 15 input rankings of 50 objects
data(data50x15)
input_rkgs <- as.matrix(data50x15[, -1])
subit_len_list_sbi <- c(3)
omega_sbi <- 5
subit_len_list_fur <- c(2:3)
search_radius <- 1
sigfur(input_rkgs, subit_len_list_sbi, omega_sbi, subit_len_list_fur, search_radius)
```

---

subit_convergence *Subiterative Convergence*

Description

Subiterative Convergence finds the consensus ranking by iteratively applying the *Modified Kemeny* algorithm on smaller number of objects, η. Starting with a given seed ranking, the consensus ranking is obtained when the algorithm converges.

Usage

```
subit_convergence(
  eta,
  seed_rkg,
  input_rkgs,
  universe_rkgs = c(),
  objNames = c(),
  wt = c()
)
```

Arguments

`eta`	a subiteration length of number of objects to consider in the smaller subset. Recommended `eta` values are between 2 and 8. Smaller `eta` values result in shorter run-time.
`seed_rkg`	an initial ranking to start the algorithm. An ideal seed ranking for *Subiterative Convergence* is the *mean seed ranking* of input rankings.
`input_rkgs`	a `n` by `k` matrix of `k` rankings of `n` objects, where each column is a complete ranking.
`universe_rkgs`	a matrix containing all possible permutations of ranking `n` objects. Each column in this matrix represents one permuted ranking. An optional parameter.
`objNames`	a `n`-length vector containing object names. An optional parameter.
`wt`	a `k`-length vector containing weights for each judge or attribute. An optional parameter.

Value

A list containing the consensus ranking (expressed as ordering), total Kemeny distance, and average tau correlation coefficient corresponding to the consensus ranking.

References

Badal, P. S., & Das, A. (2018). Efficient algorithms using subiterative convergence for Kemeny ranking problem. Computers & Operations Research, 98, 198-210. doi:10.1016/j.cor.2018.06.007

See Also

mod_kemeny, fur, sigfur, mean_seed

Examples

```
## Four input rankings of five objects
eta <- 3
seed_rkg <- c(1, 2, 3, 4, 5)
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
subit_convergence(eta, seed_rkg, input_rkgs) # Determined the consensus ranking, total Kemeny
                                      # distance, and average tau correlation coefficient

## Example with eta=1
eta <- 1
seed_rkg <- c(1, 2, 3, 4, 5)
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1, 2, 4, 5, 3),
    byrow = FALSE, ncol = 4)
subit_convergence(eta, seed_rkg, input_rkgs) # Shows a warning and returns seed ranking

## Five input rankings with five objects
## 2nd ranking == 3rd ranking, so if a third object is weighted as zero,
## we should get the same answer as the first examples
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
eta <- 3
seed_rkg <- c(1, 2, 3, 4, 5)
wt = c(1,1,0,1,1)
subit_convergence(eta, seed_rkg, input_rkgs, wt=wt) # Determined the consensus ranking, total Kemeny
                                      # distance, and average tau correlation coefficient

## Using five input rankings with five objects with prepare_data to
## automatically prepare the weight vector
input_rkgs <- matrix(c(3, 2, 5, 4, 1, 2, 3, 1, 5, 4, 2, 3, 1, 5, 4, 5, 1, 3, 4, 2, 1,
                       2, 4, 5, 3),byrow = FALSE, ncol = 5)
out = prepare_data(input_rkgs)
input_rkgs = out$input_rkgs
wt = out$wt
eta <- 3
seed_rkg <- c(1, 2, 3, 4, 5)
subit_convergence(eta, seed_rkg, input_rkgs, wt=wt) # Determined the consensus ranking, total Kemeny
                                      # distance, and average tau correlation coefficient
```

```
## Included dataset of 15 input rankings of 50 objects
data(data50x15)
input_rkgs <- as.matrix(data50x15[, -1])
mean_seed_rkg <- mean_seed(t(input_rkgs)) # Use the mean seed ranking as the seed ranking
eta <- 2
subit_convergence(eta, seed_rkg = mean_seed_rkg, input_rkgs)
```

Index