Package ‘rankUncertainty’

July 27, 2021

Title Methods for Working with Uncertainty in Rankings
Version 1.0.1.0
License MIT + file LICENSE
Encoding UTF-8
RoxygenNote 7.1.1
SystemRequirements GNU, C++11
LinkingTo cpp11, Rcpp
Depends R (>= 3.6),
Imports Rcpp, magrittr,
Suggests ggplot2, testthat (>= 3.0.0)
Config/testthat/edition 3
NeedsCompilation yes
Author Justin Rising [aut, cre]
Maintainer Justin Rising <justin.rising@us.af.mil>
Repository CRAN
Date/Publication 2021-07-27 13:20:13 UTC

R topics documented:

bottomSet ........................................................................ 2
canonicalize .................................................................. 3
coverGraph ...................................................................... 4
generateIntervals ............................................................. 5
indexIntervals ................................................................. 5
isCompatible .................................................................... 6
lessThan .......................................................................... 7
partition .......................................................................... 8
plotIntervals .................................................................. 8
toMatrix ......................................................................... 9
topSet ........................................................................... 10
bottomSet

Compute the k-bottom set for a set of intervals

Description

Suppose that we select one point from each of a set of n intervals and rank them. The k-bottom set is the set of intervals whose points can have a rank of n + 1 - k or higher.

Usage

bottomSet(intervals, k)

Arguments

intervals   data frame (see generateIntervals for the required format)
k          cutoff for inclusion

Details

See section 4.2 of Rising (2021).

Value

Indices of intervals in the k-bottom set.

References


Examples

intervals <- data.frame(left = 1:4, right = 1:4 + 0.5)
bottomSet(intervals, 2)
canonicalize

*Compute a canonical representation of an interval order*

**Description**

This function generates a set of intervals with distinct endpoints such that running any of the functions in this package on the return value gives the same answer as running those functions on the input.

**Usage**

```r
canonicalize(intervals)
```

**Arguments**

- `intervals` data frame (see `generateIntervals` for the required format)

**Details**

See section 3.1 of Rising (2021).

**Value**

A data frame in the same format as the input

**References**


**Examples**

```r
left <- c(0, 0, 0, 1, 2)
right <- c(0, 1, 2, 2, 2)
intervals <- data.frame(left = left, right = right)
toMatrix(intervals)
toMatrix(canonicalize(intervals))
```
coverGraph

*Compute the cover graph of order generated by intervals*

**Description**

The cover graph of the order generated by a set of intervals is the minimal graph whose reachability relation is that order.

**Usage**

```r
coverGraph(intervals, names = NULL)
```

**Arguments**

- `intervals`: data frame (see `generateIntervals` for the required format)
- `names`: names of intervals (1:nrow(intervals) by default)

**Details**

See section 6 of Rising (2021).

**Value**

A list of edges of the cover graph.

**References**


**Examples**

```r
left <- sort(c(1:3, 1:3 + 0.1))
right <- left + 0.7
intervals <- data.frame(left = left, right = right)
coverGraph(intervals)
```
**generateIntervals**

**Generate random intervals**

**Description**

Generate a set of intervals with endpoints uniformly distributed between 0 and 1.

**Usage**

```r
generateIntervals(n, sort = FALSE, f = NULL)
```

**Arguments**

- **n**
  - number of intervals to generate
- **sort**
  - if TRUE, sort the output intervals by their left endpoints
- **f**
  - transformation to apply to each endpoint

**Value**

Data frame with columns 'left' and 'right'. It is guaranteed that every value in 'left' is no greater than the corresponding value in 'right'.

**Examples**

```r
generateIntervals(10)
generateIntervals(20, f = qnorm)
generateIntervals(5, TRUE, f = function(x) { x + 1 })
```

**indexIntervals**

**Generate index intervals for a set of intervals**

**Description**

If we pick one point from each of a set of intervals, the index intervals describe the possible ranks of points in each interval. If this function is given simultaneous 100(1 - alpha)% confidence intervals for a distinct set of parameters, the index intervals are simultaneous 100(1 - alpha)% confidence intervals for the true ranks.

**Usage**

```r
indexIntervals(intervals)
```

**Arguments**

- **intervals**
  - data frame (see `generateIntervals` for the required format)
Details

See section 5.2 of Rising (2021).

Value

data frame (see generateIntervals for the format)

References


Examples

```r
left <- 0:2 * 0.5 + 1
right <- left + 0.75
intervals <- data.frame(left = left, right = right)
indexIntervals(intervals)
```

---

**isCompatible**

*Test whether a ranking is compatible with a set of intervals*

Description

A ranking is compatible with a set of intervals if we can pick a point from each interval such that the ranking of those points is the ranking in question.

Usage

```r
isCompatible(intervals, ranking)
```

Arguments

- **intervals** data frame (see generateIntervals for the required format)
- **ranking** permutation of 1:nrow(intervals)

Details

See section 4.1 of Rising (2021).

Value

TRUE if the ranking is compatible and FALSE otherwise

References

lessThan

Examples

define the intervals
    left <- 0:2 * 0.5 + 1
    right <- left + 0.75
    intervals <- data.frame(left = left, right = right)

calculate compatibility
    isCompatible(intervals, 1:3)
    isCompatible(intervals, c(3, 2, 1))

---

lessThan (Compare intervals)

Description

Given a data frame representing a set of intervals, return true if row i is less than row j under the
order generated by the intervals and false otherwise.

Usage

lessThan(intervals, i, j)

Arguments

- intervals: data frame (see generateIntervals for the required format)
- i: row index of left-hand side of inequality
- j: row index of right-hand side of inequality

Value

Boolean value

Examples

    left <- 0:2 * 0.5 + 1
    right <- left + 0.75
    intervals <- data.frame(left = left, right = right)
    lessThan(intervals, 1, 2)
    lessThan(intervals, 1, 3)
partition

Partition the order generated by a set of intervals

Description

A partition of the order generated by a set of intervals is a partition of their indices with the property
the sets can be ordered so that the right endpoint of every interval in a set is less than the left endpoint
of any interval in any subsequent set.

Usage

partition(intervals)

Arguments

intervals data frame (see generateIntervals for the required format)

Details

See section 3.2 of Rising (2021).

Value

A list whose entries correspond to sets in the partition

References


Examples

left <- sort(c(1:3, 1:3 + 0.1))
right <- left + 0.7
intervals <- data.frame(left = left, right = right)
partition(intervals)

plotIntervals

Plot intervals

Description

Generates a plot of a set of intervals. This is intended for simple visualizations and does not offer
any degree of customization.

Usage

plotIntervals(intervals)
toMatrix

**Arguments**

- **intervals**
  - data frame (see `generateIntervals` for the required format)

**Value**

- ggplot object

**Examples**

```r
intervals <- generateIntervals(10)
p <- plotIntervals(intervals)
p
toMatrix
```

---

**Description**

Represent the order generated by a set of intervals as a boolean matrix. This is a common input format for programs that operate on partial orders.

**Usage**

```r
toMatrix(intervals, strict = FALSE, binary = FALSE)
```

**Arguments**

- **intervals**
  - data frame (see `generateIntervals` for the required format)
- **strict**
  - is this <= or <?
- **binary**
  - output is coded as 0/1 if TRUE and FALSE/TRUE otherwise

**Value**

A boolean matrix. If strict is set to TRUE, the (i, j)th entry is `intervals[i, 'right'] < intervals[j, 'left']`. If strict is set to false, <= is used in place of <.

**Examples**

```r
intervals <- generateIntervals(10)
toMatrix(intervals)
```
Computes the k-top set for a set of intervals

Description
Suppose that we select one point from each of a set of n intervals and rank them. The k-top set is the set of intervals whose points can have a rank of k or lower.

Usage
topSet(intervals, k)

Arguments
- intervals: data frame (see `generateIntervals` for the required format)
- k: cutoff for inclusion

Details
See section 4.2 of Rising (2021).

Value
Indices of intervals in the k-top set.

References

Examples
```r
intervals <- data.frame(left = 1:4, right = 1:4 + 0.5)
topSet(intervals, 2)
```
Index

bottomSet, 2

canonicalize, 3
coverGraph, 4
generateIntervals, 2–5, 5, 6–10

indexIntervals, 5
isCompatible, 6

lessThan, 7

partition, 8
plotIntervals, 8
toMatrix, 9
topSet, 10