Package 'RSD'

June 21, 2025

Type Package

Title Compares Random Distributions using Stochastic Dominance

Version 0.2.0

Maintainer Shayan Tohidi <shayant@iastate.edu>

Description The Stochastic Dominance (SD) is the classical way of comparing two random prospects, using their distribution functions. Almost Stochastic Dominance (ASD) has also been developed to cover the SD failures due to the extreme utility functions. This package focuses on classical and heuristic methods for testing the first and second SD and ASD methods given the probability mass function (PMF) of the random prospects. The goal is to apply these methods easily, efficiently, and effectively on real-world datasets. For more details see Hanoch and Levy (1969) <doi:10.2307/2296431>, Leshno and Levy (2002) <doi:10.1287/mnsc.48.8.1074.169>, and Tzeng et al. (2012) <doi:10.1287/mnsc.1120.1616>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/ShayanTohidi/RSD

BugReports https://github.com/ShayanTohidi/RSD/issues

RoxygenNote 7.3.2

Imports dplyr, tidyr, ggplot2, magrittr, methods

Depends R (>= 3.5)

NeedsCompilation no

Author Shayan Tohidi [aut, cre],
Sigurdur Olafsson [aut]

Repository CRAN

Date/Publication 2025-06-21 14:30:02 UTC

Contents

`afsd.test` *Compares prospects based on AFSD*

Description

It compares two prospects using AFSD criteria, that is the prospect having the minimum violation area from a classic FSD.

Usage

```
afsd.test(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Details

The returned list has six elements: ‘winner‘ indicates the dominant prospect index. It will be zero if neither dominates the other. ‘epsilon‘ is the ratio of violated area to the total area between the CDFs. ‘area‘ is a vector, where the values show the area between the CDFs correspond to each segment. ‘total.area‘ is the total area between the CDFs. ‘positive.area‘ is the amount of area where the ‘area‘ vector is positive, meaning the ‘cdf1‘ is larger than ‘cdf2‘. ‘negative.area‘ is like ‘positive.area‘ for negative values.

If neither distribution dominates the other by AFSD, the ‘winner‘ output will be zero, and it happens only when the expected value of distributions are equal.

The ‘epsilon‘ and ‘winner‘ output parameters are the ones that should be taken most. The others are the calculation details and are provided for further investigation. A lower the ‘epsilon‘, lower the violation ratio of the dominant distribution, lower the eliminated extreme utilities, higher the number of decision-makers who agree on the dominant distribution.

Value

A list, including all the calculation details.

See Also

[area.btwn.cdfs.calc()] for area calculations.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
afsd.test(sd)
```

`area.below.ssd.calc` *Calculates area below SSD function*

Description

For every segment, the area below SSD function will be computed.

Usage

```
area.below.ssd.calc(outcome, ssd)
```

Arguments

`outcome`	Numeric vector, including outcome values.
`ssd`	Numeric vector, including SSD values.

Value

Numeric vector, including area below every segment of SSD function.

See Also

[calc.area.below.line()] for area below line.

`area.btwn.cdfs.calc` *Calculates area between CDFs*

Description

It calculates the area between the CDFs of two prospects divided by the CDFs and outcomes segments.

Usage

```
area.btwn.cdfs.calc(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Value

A numeric vector, including the area between the segments of CDFs.

`area.btwn.ssd.calc` *Calculates area between SSD functions*

Description

For every segments, the area between SSD functions will be computed.

Usage

```
area.btwn.ssd.calc(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Value

Numeric vector, including area differences in every segments.

See Also

[modif.outcome.ssd.calc(), area.below.ssd.calc()] for more details.

`assd.ll.test` *Compares random prospects by ASSD-LL*

Description

It uses the positive and negative areas that are computed by ASSD-LL and the expected values of the prospects to compare them based on the ASSD-LL rule. If the violation area ratio is less than 0.5 for a prospect, and its expected value is larger, it dominates the other by ASSD-LL.

Usage

```
assd.ll.test(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Details

epsilon shows the ratio of the violation. Smaller epsilon means more decision-makers agree with the result.

The returned list has six elements: ‘winner‘ indicates the dominant prospect index. It will be zero if neither dominates the other. ‘epsilon‘ is the ratio of violated area to the total area between the CDFs. ‘area‘ is a vector, where the values show the area between the CDFs correspond to each segment. ‘total.area‘ is the total area between the CDFs. ‘positive.area‘ is the amount of area where the ‘area‘ vector is positive, meaning the ‘cdf1‘ is larger than ‘cdf2‘ and ‘ssd1‘ is larger than ‘ssd2‘. ‘negative.area‘ is like ‘positive.area‘ for negative values.

If neither distribution dominates the other by ASSD-LL, the ‘winner‘ output will be zero, and it happens only when the distribution with a higher expected value has the ‘epsilon‘ which is larger than 0.5.

Value

A list, including all the calculation details.

See Also

[expected.values(), pos.neg.area.assd.ll(), afsd.test()] for more details.

`assd.test` *Compares prospects based on ASSD methods*

Description

It compares two prospects using ASSD criteria, that is the prospect having the minimum violation area from a classic SSD.

Usage

```
assd.test(sd.obj, type)
```

Arguments

`sd.obj`	StochasticDominance object.
`type`	A character vector, including the name of ASSD methods.

Details

The ‘type‘ argument must be one of the ’ll’ or ’ths’, otherwise it will raise an error.

The ‘epsilon‘ and ‘winner‘ output parameters are the ones that should be taken most. The others are the calculation details and are provided for further investigation. A lower the ‘epsilon‘, lower the violation ratio of the dominant distribution, lower the eliminated extreme utilities, higher the number of decision-makers who agree on the dominant distribution.

Value

A list, including calculation details.

See Also

[assd.ll.test(), assd.ths.test] for more details.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
assd.test(sd, 'll')
assd.test(sd, 'ths')
```

`assd.ths.test` *Compares random prospects by ASSD-THS*

Description

It uses the areas between the SSD functions, and the expected values of the prospects to compare them based on the ASSD-THS rule. If the violation area ratio is less than 0.5 for a prospect, and its expected value is larger, it dominates the other by ASSD-THS.

Usage

```
assd.ths.test(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Details

epsilon shows the ratio of the violation. Smaller epsilon means more decision-makers agree with the result.

The returned list has six elements: ‘winner‘ indicates the dominant prospect index. It will be zero if neither dominates the other. ‘epsilon‘ is the ratio of violated area to the total area between the SSDs. ‘area‘ is a vector, where the values show the area between the SSDs correspond to each segment. ‘total.area‘ is the total area between the SSDs. ‘positive.area‘ is the amount of area where the ‘area‘ vector is positive, meaning the ‘ssd1‘ is larger than ‘ssds2‘. ‘negative.area‘ is like ‘positive.area‘ for negative values.

If neither distribution dominates the other by ASSD-THS, the ‘winner‘ output will be zero, and it happens only when the distribution with a higher expected value has the ‘epsilon‘ which is larger than 0.5.

Value

A list, including all the calculation details.

See Also

[expected.values(), area.btwn.ssd.calc()] for more details.

`calc.area.below.line` *Calculates the area between x-axis and a straight line*

Description

It takes four parameters as the coordinates of two points at the beginning and end of a straight line, and calculates the area exactly below the line (between the line and the x-axis).

Usage

```
calc.area.below.line(x1, x2, y1, y2)
```

Arguments

`x1, x2`	Float values, indicating the first coordinates of start and end points, respectively.
`y1, y2`	Float values, indicating the second coordinates of start and end points, respectively.

Value

A float.

`calc.intersection` *Calculates the intersection point of two lines.*

Description

Given the start and end coordinates of two straight lines, we calculate the intersection point of them.

Usage

```
calc.intersection(x1, x2, y11, y12, y21, y22)
```

Arguments

`x1, x2`	Float values.
`y11, y12`	Float values, corresponding to the first line.
`y21, y22`	Float values, corresponding to the second line.

Details

Having intersection point has been checked before.

Value

A list, including the coordinates of the intersection point.

See Also

[has.intersect()]

`comparison` *Comparing two numeric vectors*

Description

This function compares two numeric vectors. The vector whose all elements smaller or equal to all elements of the other one where at least one element is smaller, will be the winner.

Usage

```
comparison(x, y)
```

Arguments

`x, y`	Numeric vectors.

Details

The function returns the index of the winner, meaning 1 or 2, corresponding to parameters ‘x‘ and ‘y‘, respectively. If no vector meets the condition, 0 will be returned.

Value

An integer, among 0, 1, 2.

`createStochasticDominance` *Constructor of StochasticDominance Class*

Description

It is much easier to use this constructor to create an instance of the StochasticDominance class. It handles calculation of the cdf and ssd values in an efficient way.

Usage

```
createStochasticDominance(outcome1, outcome2, prob1, prob2)
```

Arguments

`outcome1, outcome2`	Numeric vectors. The outcomes corresponding to each prospect.
`prob1, prob2`	Numeric vectors. The probabilities corresponding to each prospect.

Value

StochasticDominance object.

See Also

[StochasticDominance()]

Examples

```
createStochasticDominance(outcome1 = c(1,4,7),
                          outcome2 = c(2,3,5),
                          prob1 = c(1/3,1/3,1/3),
                          prob2 = c(1/6,1/6,2/3))
```

`data_ex` *The example dataset*

Description

It is a real dataset in agriculture, which includes planted cultivars in multi-environment and the resulting yield.

Usage

```
data_ex
```

Format

A data frame with 377 rows and 3 columns:

gen Character vector of genotype (cultivar) names

env Character vector of environment names

yield Numeric vector of yield of planting each cultivar in each environment

Source

The phenotypic data from G2F initiative in 2018: <https://doi.org/10.25739/anqq-sg86>

`expected.values` *Calculating Expected value*

Description

It calculates the expected value of both prospects given their distributions.

Usage

```
expected.values(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Value

A list, including two double elements as the expected value of each prospect.

`fsd.plot` *Drawing the CDFs*

Description

It visualizes the CDFs of both prospects.

Usage

```
fsd.plot(sd.obj, names = c("1", "2"))
```

Arguments

`sd.obj`	StochasticDominance object.
`names`	A character vector, including the names of prospects in order.

Details

The parameter ‘names‘ only accepts character vector, otherwise an error will be raised.

The function shows the step plot, and returns its object for further modifications.

Value

A list, including plot elements.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
fsd.plot(sd, names = c('First', 'Second'))
```

`fsd.test` *Compares random prospects by FSD*

Description

It compares two random prospects by the first-order stochastic dominance (FSD).

Usage

```
fsd.test(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Details

A prospect dominates when its CDF is below the other one. It means that all element of the CDF vector must be equal or smaller, and at least one element should be smaller for the dominant prospect.

If neither prospect dominates the other by FSD, it returns 0. It means that the CDFs intersect each other.

Value

An integer, indicating the index of the dominant prospect.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
fsd.test(sd)
```

`has.intersection` *If two lines have intersection or no.*

Description

It determines if two lines have intersection point or not. The start and end points of both lines have equal first coordinate value (the one corresponds to the x-axis).

Usage

```
has.intersection(x1, x2, y11, y12, y21, y22)
```

Arguments

`x1, x2`	Float values.
`y11, y12`	Float values, corresponding to the first line.
`y21, y22`	Float values, corresponding to the second line.

Value

A Boolean.

`modif.outcome.ssd.calc` *Modify outcome and SSD vectors*

Description

It modifies outcome vector to have all original plus intersection points in ascending order. The corresponding SSD vectors will also be returned.

Usage

```
modif.outcome.ssd.calc(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Value

A list of three elements.

`pos.neg.area.assd.ll` *Calculates positive and negative area between CDFs for ASSD-LL*

Description

It calculates the positive and negative areas between CDFs. The positive is where both CDF and SSD of the first prospect is larger, and vice versa for the negative case.

Usage

```
pos.neg.area.assd.ll(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Value

A list, including three elements corresponding to the positive and negative areas, respectively, and the area vector at the end.

See Also

[modif.outcome.ssd.calc()] for more details.

`ssd.calc` *Calculates the SSD values for a prospect.*

Description

Calculates the SSD values for a prospect.

Usage

```
ssd.calc(outcome, cdf)
```

Arguments

`outcome`	Numeric vector, indicating the outcome values.
`cdf`	Numeric vector, indicating the cumulative probabilities.

Value

Numeric vector, indicating the SSD values.

`ssd.plot` *Drawing the SSD*

Description

It visualize the SSD values of both prospects.

Usage

```
ssd.plot(sd.obj, names = c("1", "2"))
```

Arguments

`sd.obj`	StochasticDominance object.
`names`	A character vector, including the names of prospects in order.

Details

The parameter ‘names‘ only accepts character vector, otherwise an error will be raised.

The function shows the line plot and returns its object for further modification.

Value

A list, including plot elements.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
ssd.plot(sd, names = c('First', 'Second'))
```

`ssd.test` *Compares random prospects by SSD*

Description

It compares two random prospects by the second-order stochastic dominance (SSD).

Usage

```
ssd.test(sd.obj)
```

Arguments

`sd.obj`	StochasticDominance object.

Details

A prospect dominates when its SSD is below the other one. It means that all element of the SSD vector must be equal or smaller, and at least one element should be smaller for the dominant prospect.

If neither prospect dominates the other by SSD, it returns 0. It means the SSDs intersect each other.

Value

A number. Indicating dominant prospect index.

See Also

[ssd.calc()] for the calculation.

Examples

```
sd = createStochasticDominance(outcome1 = c(1,4,7),
                               outcome2 = c(2,3,5),
                               prob1 = c(1/3,1/3,1/3),
                               prob2 = c(1/6,1/6,2/3))
ssd.test(sd)
```

`StochasticDominance-class` *StochasticDominance Class*

Description

Represents two distributions (prospects) that are going to be compared using Stochastic Dominance (SD).

Details

It contains the input validation needed for comparing two prospects. For example, having sorted ‘outcome‘, each of ‘prob1‘ and ‘prob2‘ adds up to one, arguments having the same lengths, and having matched probability, cumulative, and ssd arguments.

Slots

`outcome` Numeric vector. The combined outcome values in ascending order.

`prob1,prob2` Numeric vectors. Probabilities corresponding to the prospects.

`cdf1,cdf2` Numeric vectors. Cumulative values corresponding to the prospects.

`ssd1,ssd2` Numeric vectors. SSD values corresponding to the prospects.

Index