Package 'SAFD'

October 12, 2022

Type Package

Title Statistical Analysis of Fuzzy Data

Version 2.1

Date 2019-07-03

Author Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

Maintainer Asun Lubiano <lubiano@uniovi.es>

Description The aim of the package is to provide some basic functions for doing statistics with one dimensional Fuzzy Data (in the form of polygonal fuzzy numbers). In particular, the package contains functions for the basic operations on the class of fuzzy numbers (sum, scalar product, mean, median, Hukuhara difference) as well as for calculating (Bertoluzza) distance and sample variance. Moreover a function to simulate fuzzy random variables and bootstrap tests for the equality of means is included. Version 2.1 fixes some bugs of previous versions.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-03 12:00:02 UTC

R topics documented:

SAFD-package *Statistical Analysis of Fuzzy Data*

Description

The aim of the package is to provide some basic functions for doing statistics with one-dimensional Fuzzy Data (in the form of polygonal fuzzy numbers).

Details

Package:	SAFD
Type:	Package
Version:	2.1
Date:	2019-07-03
License:	GPL (>=2)

The aim of the package is to provide some basic functions for doing statistics with one dimensional Fuzzy Data (in the form of polygonal fuzzy numbers). In particular, the package contains functions for the basic operations on the class of fuzzy numbers (sum, scalar product, mean, median, Hukuhara difference) as well as for calculating (Bertoluzza) distance and sample variance. Moreover a function to simulate fuzzy random variables and bootstrap tests for the equality of means is included.
Version 2.1 fixes some bugs of previous versions.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

Maintainer: Asun Lubiano <lubiano@uniovi.es>

References

[1] Bertoluzza, C., Corral, N., Salas, A.: *On a new class of distances between fuzzy numbers*, Mathware Soft Comput., 2, pp. 71-84 (1995)
[2] Colubi, A.: *Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data*, Fuzzy Sets and Systems, 160(3), pp. 344-356 (2009)
[3] Gil, M.A., Lopez, M.T., Lubiano, M.A., Montenegro, M.: *Regression and correlation analyses of a linear relation between random intervals*, Test, 10(1), pp. 183-201 (2001)
[4] Gil, M.A.; Montenegro, M.; Gonzalez-Rodriguez, G.; Colubi, A.; Casals, R.: *Bootstrap approach to the multi-sample test of means with imprecise data*, Computational Statistics and Data Analysis, 51(1), pp. 148-162 (2006)
[5] Gonzalez-Rodriguez, G.; Blanco, A.; Colubi, A.; Lubiano, M.A.: *Estimation of a simple linear regression model for fuzzy random variables*, Fuzzy Sets and Systems, 160(3), pp. 357-370 (2009)
[6] Gonzalez-Rodriguez, G., Colubi, A., Trutschnig, W.: *Simulation of fuzzy random variables*, Information Sciences, 179(5), pp. 642-653 (2009)
[7] Montenegro, M., Colubi, A., Casals, M.R., Gil, M.A.: *Asymptotic and bootstrap techniques for testing the expected value of a fuzzy random variable*, Metrika, 59, pp. 31-49 (2004)
[8] Montenegro, M., Casals, M.R., Lubiano, M.A., Gil, M.A.: *Two-sample hypothesis tests of means of a fuzzy random variable*, Information Sciences, 133(1-2), pp. 89-100 (2001)
[9] Sinova, B., Gil, M.A., Colubi, A., Van Aelst, S.: *The median of a random fuzzy number. The 1-norm distance approach*, Fuzzy Sets and Systems, 200, pp. 99-115 (2012)
[10] Trutschnig, W., *A strong consistency result for fuzzy relative frequencies interpreted as estimator for the fuzzy-valued probability*, Fuzzy Sets and Systems, 159(3), pp. 259-269 (2008)
[11] Trutschnig, W., Gonzalez-Rodriguez, G., Colubi, A.; Gil, M.A.: *A new family of metrics for compact, convex (fuzzy) sets based on a generalized concept of mid and spread*, Information Sciences, 179(23), pp. 3964-3972 (2009)
[12] Viertl, R., Hareter, D.: *Beschreibung und Analyse unscharfer Information: Statistische Methoden fuer unscharfe Daten*, Springer Wien New York, 2006

See Also

http://bellman.ciencias.uniovi.es/smire+codire/

`bertoluzza` *Bertoluzza distance*

Description

Given two polygonal fuzzy numbers `X`, `Y` in the correct format (testing by `checking`) the function calculates the Bertoluzza distance of `X`, `Y`. The parameter `theta` (being the weight of the spread) has to fulfill `theta`>0, by default `theta`=1/3 (which corresponds to the Lebesgue measure as weighting measure on [0,1]). For detailed explanation see the papers [1] and [2] below.

Usage

```
bertoluzza(X, Y, theta = 1/3, pic = 0)
```

Arguments

X	...dataframe (polygonal fuzzy number)
Y	...dataframe (polygonal fuzzy number)
theta	...numeric and >0
pic	...numeric, if pic==1 X and Y are plotted, by default no plot is produced.

Details

See examples

Value

...in case X and Y are in the correct form the code returns the Bertoluzza distance, otherwise NA is returned.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Trutschnig, W., Gonzalez-Rodriguez, G., Colubi, A., Gil, M.A.: *A new family of metrics for compact, convex (fuzzy) sets based on a generalized concept of mid and spread*, Information Sciences, 179(23), pp. 3964-3972 (2009)

[2] Bertoluzza, C., Corral, N., Salas, A.: *On a new class of distances between fuzzy numbers*, Mathware Soft Comput., 2, pp:71-84 (1995)

See Also

See Also as checking, Mmean, Bvar

Examples

```
#Example 1:
data(XX)
X<-translator(XX[[1]],50)
Y<-translator(XX[[2]],50)
Z<-translator(XX[[3]],50)
ZZ<-list(X,Y,Z)
b<-bertoluzza(X,Y,1/3,1)
b

#Example 2: example (SLLN for the FRV)
V<-translator(XX[[3]],100)
```

```
YY<-vector("list",length=50)
  for(i in 1:50){
   YY[[i]]<-generator(V,,,)
   }
M<-Mmean(YY)
head(M)
b<-bertoluzza(M,V,1/3,1)
b

#Example 3:
V<-translator(XX[[3]],100)
YY<-vector("list",length=1000)
  for(i in 1:1000){
   YY[[i]]<-generator(V,,,)
   }
M<-Mmean(YY)
head(M)
b<-bertoluzza(M,V,1/3,1)
b

#Example 4:
X<-data.frame(x=c(0,1,1,2),alpha=c(0,1,1,0))
Y<-data.frame(x=c(0,1,2),alpha=c(0,1,0))
b<-bertoluzza(X,Y,1/3,1)
b

#Example 5:
data(Trees)
X<-Mmean(Trees[[1]])
Y<-Mmean(Trees[[2]])
Z<-Mmean(Trees[[3]])
b1<-bertoluzza(X,Y,1)
b1
b2<-bertoluzza(X,Z,1)
b2
b3<-bertoluzza(Y,Z,1)
b3
```

`btest1.mean` *One-sample bootstrap test for the mean of a FRV*

Description

Given a sample `XX` of polygonal fuzzy numbers and a polygonal fuzzy number `V` the function first checks if each element of `XX` and `V` has the correct format and if the alpha-levels of all input fuzzy numbers coincide. In case yes, the function computes the standardized mean squared Bertoluzza-distance from the sample mean to `V` as test-statistic. Afterwards for `B` bootstrap samples the (bootstrap) statistic is calculated. The returned p-value is calculated as the portion of the obtained values of the bootstrap statistic that are greater than the value of the test-statistic. Furthermore, if `pic=1` sample mean and `V` are plotted. For detailed explanation see papers [1] and [2] below.

Usage

```
btest1.mean(XX, V, theta = 1/3, B = 100, pic = 0)
```

Arguments

XX	...list of polygonal fuzzy numbers (the functions implicitly checks the conditions).
V	...polygonal fuzzy number that is tested to be the mean of the FRV.
theta	...numeric and >0, see bertoluzza
B	...integer, by default B=1000.
pic	...numeric, if pic=1 sample mean and V are plotted. By default pic=1.

Details

See examples

Value

Given input XX and V in the correct format, the function returns the p-value of the two-sided bootstrap test that the expectation is V.

Note

The function is quite slow.
In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Colubi, A.: *Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data*, Fuzzy Sets and Systems, 160(3), pp. 344-356 (2009)
[2] Montenegro, M., Colubi, A., Casals, M.R., Gil, M.A.: *Asymptotic and bootstrap techniques for testing the expected value of a fuzzy random variable*, Metrika, 59, pp. 31-49 (2004)

See Also

See Also as Mmean, Bvar, bertoluzza, btest2.mean, btestk.mean

Examples

```
#Example 1: run for bigger sample sizes:
data(XX)
V<-translator(XX[[3]],50)
V2<-V
SS<-vector("list",length=50)
```

```
for (j in 1:50){
 SS[[j]]<-generator(V2,)
 }
b<-btest1.mean(SS,V2,B=10)
b

#Example 2: takes some time to run:
#data(Trees)
#V<-Trees[[1]][[47]]
#b<-btest1.mean(Trees[[1]],V,100)
#b
```

`btest2.mean` *Two-sample bootstrap test on the equality of mean of two FRVs*

Description

Given two samples `XX` and `YY` of polygonal fuzzy numbers the function first checks if each element of `XX` and `YY` has the correct format and if the alpha-levels of all input fuzzy numbers coincide. In case yes, the function compute the test statistic described in [1] below. Before doing the resampling `Mmean(YY)` is added to each element of `XX` and vice versa. Based on these two new samples `B` values of the test statistic are calculate. The returned p-value is calculated as the portion of the obtained values of the bootstrap statistic that are greater than the value of the test-statistic. If `pic=1` then the sample means of `XX` and `YY` are plotted, otherwise no plot is produced. For detailed explanation see the papers [1] and [2] below.

Usage

```
btest2.mean(XX, YY, theta = 1/3, B = 100, pic = 1)
```

Arguments

`XX`	...should be a list of polygonal fuzzy numbers (the functions implicitly checks the conditions)
`YY`	...should be a list of polygonal fuzzy numbers (the functions implicitly checks the conditions)
`theta`	...numeric and >0
`B`	...integer, by default `B`=1000.
`pic`	...numeric, if `pic`=1 then the sample means of `XX` and `YY` are plotted. By default `pic`=1.

Details

See examples

Value

Given input XX and YY in the correct format, the function returns the p-value of the two-sided bootstrap test.

Note

The function is quite slow.
In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Colubi, A.: *Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data*, Fuzzy Sets and Systems, 160(3), pp. 344-356 (2009)

[2] Montenegro, M., Casals, M.R., Lubiano, M.A., Gil, M.A.: *Two-sample hypothesis tests of means of a fuzzy random variable*, Information Sciences, Vol. 133(1-2), pp. 89-100 (2001)

See Also

See Also as Mmean, Bvar, bertoluzza, btest1.mean, btestk.mean

Examples

```
#Example 1: run for bigger B
data(XX)
X<-translator(XX[[1]],20)
Y<-translator(XX[[2]],20)
XX<-vector("list",length=30)
for (j in 1:30){
 XX[[j]]<-generator(X,)
 }
YY<-vector("list",length=20)
for (j in 1:20){
 YY[[j]]<-generator(Y,)
 }
b<-btest2.mean(XX,YY,B=10)
b

#Example 2: takes some time in the current version:
#data(Trees)
#b<-btest2.mean(Trees[[1]],Trees[[2]],50)
#b
#b<-btest2.mean(Trees[[1]],Trees[[3]],50)
#b
```

`btestk.mean` *Multi-sample bootstrap test for the equality of the mean of FRVs*

Description

Given a list `XXX` of length `k` sublists of polygonal fuzzy numbers the function first checks if each element of the sublists has the correct format and if the alpha-levels of all input fuzzy numbers coincide. The vector `sel` contains the numbers of the sublists the user wants to filter to. After filtering the relevant part of `XXX` the function computes the test-statistic, which compares the sum of the distances of the groups means and the overall mean with the sum of the group variances. Before doing the resampling `length(sel)` new samples are calculated by adding to each element of every fixed group the sum of all means of the other groups. Based on these `length(sel)` new samples `B` values of the (bootstrap) test statistic are calculate. The returned p-value is calculated as the portion of the obtained values of the bootstrap statistic that are greater than the value of the test-statistic. If `pic=1` then the sample means of the via `sel` selected samples from `XXX` and the total mean are plotted, otherwise no plot is produced. For a more detailed explanation see the papers [1] and [2] below.

Usage

```
btestk.mean(XXX, sel, theta = 1/3, B = 100, pic = 1)
```

Arguments

`XXX`	... A list of sublists, each of which contains polygonal fuzzy numbers
`sel`	...vector, selection of number of the samples (sublists) to be considered
`theta`	...numeric and >0
`B`	...integer, by default `B=100`.
`pic`	...numeric, if `pic=1` then the sample means of the via `sel` selected samples from `XXX` and the total mean are plotted. By default `pic=1`.

Details

See examples

Value

Given input `XXX` in the correct format, the function returns the p-value of the two-sided test.

Note

The function is quite slow.
In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Colubi, A.: *Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data*, Fuzzy Sets and Systems, 160(3), pp. 344-356 (2009)

[2] Gil, M.A.; Montenegro, M.; Gonzalez-Rodriguez, G.; Colubi, A.; Casals, R.: *Bootstrap approach to the multi-sample test of means with imprecise data*, Computational Statistics and Data Analysis, 51(1), pp. 148-162 (2006)

See Also

See Also as `Mmean`, `Bvar`, `bertoluzza`, `btest1.mean`, `btest2.mean`

Examples

```
#Example 1: very small B only for testing purpose
data(Trees)
sel<-c(1,2,3)
b<-btestk.mean(Trees,sel,B=5)
b

#Example 2: run for bigger B
#b<-btestk.mean(Trees,sel,100)
#b
```

`Bvar` *(Sample) Variance*

Description

The sample variance of a sample of polygonal fuzzy numbers with respect to the Bertoluzza distance is calculated. Given a list `XX` of polygonal fuzzy numbers the function first checks if each element of the list has the correct form and if the alpha-levels of all elements in the list coincide. If these conditions are fulfilled the Bertoluzza sample variance will be returned (i.e. the average Bertoluzza distance of the elements of `XX` to its mean). If not the `translator` function can be used to transform the elements of the list in the correct format. For details see [1] from below, and replace the kernel K with the expression induced by the Bertoluzza metric. The parameter `theta` has to fulfill `theta`>0.

Usage

```
Bvar(XX, theta = 1/3)
```

Arguments

`XX`	...should be a list of polygonal fuzzy numbers (the functions implicitly checks the conditions) verifying the above mentioned conditions
`theta`	...numeric and >0, see `bertoluzza`

Details

See examples

Value

Given input XX in the correct format the function returns the Bertoluzza variance of the sample XX.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Gonzalez-Rodriguez, G.; Blanco, A.; Colubi, A.; Lubiano, M.A.: *Estimation of a simple linear regression model for fuzzy random variables*, Fuzzy Sets and Systems, 160(3), pp. 357-370 (2009)

See Also

See Also as bertoluzza, Mmean

Examples

```
#Example 1:
data(XX)
X<-translator(XX[[1]],50)
Y<-translator(XX[[2]],50)
Z<-translator(XX[[3]],50)
YY<-list(X,Y,Z)
A<-Bvar(YY,1)
A

#Example 2:
data(XX)
v<-Bvar(list(XX[[1]],XX[[1]]),1/3)
v

#Example 3:
data(Trees)
Species1_Var<-Bvar(Trees$species1,1/3)
Species1_Var
Species2_Var<-Bvar(Trees$species2,1/3)
Species2_Var
Species3_Var<-Bvar(Trees$species3,1/3)
Species3_Var
```

checking *Checking correct data format*

Description

The function checks if the input data is of the correct form of a polygonal fuzzy number, i.e. a dataframe with the columns "x" and "alpha" fulfilling the following conditions: (1) alpha-values have to be in [0,1] with the minimum alpha-level being 0 and maximum being 1, (2) the x-values have to be non-missing and non-decreasing, (3) the alpha-levels have to increase from 0 to 1 and afterwards decrease from 1 to 0 in the same way (i.e. the alpha-column consists of an increasing vector from 0 to 1 plus the same vector in decreasing order). As a consequence the dataframe always has an even number of rows, see examples. The function is used internally in almost all the other functions to do a preliminary checking if the input data is of the correct form.

Usage

```
checking(X, com = 1)
```

Arguments

X	...can be any data frame.
com	...numeric, if com=1 then, in case of an error, a comment is printed. By default com=1.

Details

See examples

Value

The function returns the value 1 if the input fulfills all conditions, if not, 0 is returned.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also as checking2, translator

Examples

```
#Example 1:
data(XX)
a<-checking(XX[[1]],1)
a

#Example 2:
X<-data.frame(y=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,0.9,0.9,0))
a<-checking(X)
a

#Example 3:
X<-data.frame(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,0.9,0.9,0))
a<-checking(X)
a

#Example 4:
X<-data.frame(x=c(-2,-0.75,-0.25,-0.5,1),alpha=c(0,0.6,1,1,0))
a<-checking(X)
a

#Example 5:
X<-data.frame(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.3,0,1,0,0.3))
a<-checking(X)
a

#Example 6:
Y<-data.frame(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.3,1,0,0.3))
a<-checking(Y)
a

#Example 7:
Z<-data.frame(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,1,1,0))
a<-checking(Z)
a

#Example 8:
U<-data.frame(x=c(-1,0,1),alpha=c(0,1,0))
a<-checking(U,)
a
```

`checking2` *Checking correct data format (weak version)*

Description

The function checks if the input data defines a polygonal fuzzy number, i.e. a dataframe with the columns "x" and "alpha" fulfilling the following conditions: (1) alpha-values have to be in [0,1] with the minimum alpha-level being 0 and maximum being 1, (2) the x-values have to be non-missing and non-decreasing, (3) the alpha-levels have to increase from 0 to 1 and afterwards decrease from 1 to 0 (not necessarily in the same way). The function is only used for the `translator` function.

Usage

```
checking2(X, com = 1)
```

Arguments

X	...can be any data frame.
com	...numeric, if com=1 then, in case of an error, a comment is printed. By default com=1.

Details

See examples

Value

The function returns the value 1 if the input fulfills all conditions, if not, 0 is returned.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also as checking, translator

Examples

```
#Example:
 U<-data.frame(x=c(-1,0,1),alpha=c(0,1,0))
 #a<-checking(U,)
 a<-checking2(U,)
a
```

---

decomposer *Decomposer*

Description

Given a dataframe X the function first calls checking in order to test if X is in the desired format. If yes, the dataframe X (polygonal fuzzy number) is expressed as a dataframe with (nrow(X)+1) rows as described in the paper [1] below, if no, NULL is returned. The main aim of decomposer is to provide the simulator-function called generator with the correct input.

Usage

```
decomposer(X)
```

Arguments

X	...dataframe, if checking(X)=1 the decomposed version of X is returned.

Details

See examples

Value

In case checking(X)=1 decomposer returns a dataframe with (nrow(X+1))-rows (see [1]), otherwise NA is returned.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Gonzalez-Rodriguez, G., Colubi, A., Trutschnig, W.: *Simulation of fuzzy random variables*, Inf.Sci., 179(5), pp. 642-653 (2009)

See Also

See also checking, generator

Examples

```
#Example:
 data(XX)
 A<-decomposer(XX[[2]])
 A<-decomposer(XX[[1]])
head(A)
```

`defuzzify` *Defuzzification*

Description

Given a list `XX` of polygonal fuzzy numbers the function defuzzifies all elements of the list and returns the vector of Steiner points (as weighting measure the Lebesgue measure on [0,1] is used).

Usage

```
defuzzify(XX)
```

Arguments

`XX`	...should be a list of polygonal fuzzy numbers (the function implicitly checks the conditions)

Details

See examples

Value

Given input `XX` in the correct format the function returns vector of Steiner points.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

Examples

```
#Example:
data(XX)
V<-translator(XX[[3]],50)
V2<-V
V2$x<-V$x/20
SS<-vector("list",length=150)
for (j in 1:150){
 SS[[j]]<-generator(V2,)
 }
a<-defuzzify(SS)
a
```

`Fmedian` *Median of a fuzzy sample*

Description

Given a list `XX` of polygonal fuzzy numbers the functions first checks (1) if each element of the lists is in the correct form (tested by `checking`) and (2) if the alpha-levels of all elements coincide. If all conditions are fulfilled the function calculates the (levelwise) median (which, by definition, is a fuzzy number too) using a large number of levels, by default `nl`=101.

Usage

```
Fmedian(XX, nl = 101, pic = 1)
```

Arguments

`XX`	...list of polygonal fuzzy numbers with the same alpha levels (the functions implicitly checks the conditions)
`nl`	...number of equidistant alpha-level, by default `nl`=101
`pic`	...numeric, if `pic`=1 the sample, its mean and its median are plotted.

Details

See examples.

Value

Given correct input `XX` the function returns the median of the sample.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Sinova, B., Gil, M.A., Colubi, A., Van Aelst, S.: *The median of a random fuzzy number. The 1-norm distance approach*, Fuzzy Sets and Systems, 200, pp. 99-115 (2012)

See Also

See Also as `Mmean`

Examples

```
#Example 1:
data(XX)
V<-translator(XX[[3]],100)
YY<-vector("list",length=50)
  for(i in 1:50){
   YY[[i]]<-generator(V,,,)
   }
Me<-Fmedian(YY)

#Example 2:
data(Trees)
Species1_Median<-Fmedian(Trees[[1]],nl=11)
Species1_Median
Species2_Median<-Fmedian(Trees[[2]])
Species3_Median<-Fmedian(Trees[[3]])
```

`generator` *Simulation of fuzzy random variables (FRV)*

Description

The second procedure described in [1] is implemented. Given an input dataframe `V` in the correct format (tested by `checking`), which will be the expectation of the simulated FRV first `decomposer(V)` is called. Loosely speaking, the dataframe `Y` returned by `decomposer`, which contains the "coordinates" of `V` with respect to a certain "basis" (see [1]), is perturbated stochastically in order to generate a new polygonal fuzzy number. The distributions used for these perturbations can be selected in the call of the function, however, in this version only a few choices are possible: (1) The perturbation of the centre of the 1-cut `pertV` has to be of the form `norm(0,sigma)` or `unif(-a,a)`, sigma,a>0. (2) The perturbation of the left part of the fuzzy set `pertL` has to be of the form `chisq(1)`, `exp(1)` or `lnorm(a,b)` with expectation=1. (3) The perturbation of the right part of the fuzzy set `pertR` has to be of the same form as that for the left part.

Usage

```
generator(V, pertV = list(dist = "norm", par = c(0, 1)),
             pertL = list(dist = "chisq", par = c(1)),
             pertR = list(dist = "chisq", par = c(1)))
```

Arguments

`V`	...polygonal fuzzy set in the correct format (tested by `checking`)
`pertV`	...list containing elements "dist" and "par". "dist" denotes the chosen distribution family (normal or uniform) and "par" the corresponding parameters.
`pertL`	...list containing elements "dist" and "par". "dist" denotes the chosen distribution family (chisq or lnorm) and "par" the corresponding parameters.
`pertR`	...list containing elements "dist" and "par". "dist" denotes the chosen distribution family (chisq or lnorm) and "par" the corresponding parameters.

Details

See examples

Value

Given correct input data, the function returns a polygonal fuzzy number that can be seen as a realisation of a FRV with expectation `V` (see [1]).

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

References

[1] Gonzalez-Rodriguez, G., Colubi, A., Trutschnig, W.: *Simulation of fuzzy random variables*, Information Sciences, 179(5), pp. 642-653 (2009)

See Also

See Also as `decomposer`

Examples

```
#Example:
data(XX)
V<-translator(XX[[3]],100)
YY<-vector("list",length=100)
  for(i in 1:100){
   YY[[i]]<-generator(V,,,)
   }
M<-Mmean(YY)
plot(M,type="l",xlim=c(-3,4))
lines(V,type="l",col="red",lwd=2)
```

---

hukuhara *Hukuhara Difference*

Description

Given two polygonal fuzzy numbers the functions calculates the Hukuhara difference `Y-X` if it exists. First the input data is tested for having the correct format using the function `checking`. If the Hukuhara difference exists and `pic=1` then the input and the Hukuhara difference is plotted, otherwise no plot is produced.

Usage

```
hukuhara(X, Y, pic = 0)
```

Arguments

X	...polygonal fuzzy number (tested by checking)
Y	...polygonal fuzzy number (tested by checking)
pic	...numeric, if pic=1 then X, Y and Y-X (if existing) is plotted

Details

See examples

Value

In case the input data is in the correct form and the Hukuhara difference exists, the Hukuhara difference is returned, in case not, NULL is returned.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also as checking, Msum

Examples

```
#Example 1:
Y<-data.frame(x=c(0,0,0,1,2,2),alpha=c(0,0.5,1,1,0.5,0))
X<-data.frame(x=c(0,0,0,0,1.5,2),alpha=c(0,0.5,1,1,0.5,0))
Z<-data.frame(x=c(0,0,0,0.75,1.5,1.5),alpha=c(0,0.5,1,1,0.5,0))
h1<-hukuhara(X,Y,1)
h1
h2<-hukuhara(Z,Y,1)
h2

#Example 2: in this case the hukuhara diff has to exist by construction
data(XX)
X<-translator(XX[[1]],50)
shift<-seq(-1,1,length=100)
Y<-X
Y$x<-X$x+shift
h<-hukuhara(X,Y,1)
```

Mmean *Minkowski mean*

Description

Given a sample XX of polygonal fuzzy numbers the Minkowski-mean of the sample is calculated. The function fist calls Msum to check if XX has the correct format and, in case yes, sc_mult is used to calculate the Minkowski-mean of the fuzzy sample XX. If pic=1 then the sample and its mean are plotted, otherwise no plot is produced.

Usage

```
Mmean(XX, pic = 0)
```

Arguments

XX	...should be a list of polygonal fuzzy numbers (the functions implicitly checks the conditions)
pic	...numeric, if pic=1 then the sample mean of XX is printed. By default pic=0.

Details

See examples

Value

Given input XX in the correct format the function returns the Minkowski mean of the polygonal fuzzy numbers contained in the list XX.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also as checking, translator, Msum, sc_mult, Bvar

Examples

```
#Example 1:
data(XX)
A<-Mmean(XX,1)

X<-translator(XX[[1]],50)
Y<-translator(XX[[2]],50)
Z<-translator(XX[[3]],50)
YY<-list(X,Y,Z)
A<-Mmean(YY,pic=1)


#Example 2:
data(Trees)
Species1_Mean<-Mmean(Trees[[1]],1)
Species1_Mean
Species2_Mean<-Mmean(Trees[[2]],1)
Species2_Mean
Species3_Mean<-Mmean(Trees[[3]],1)
Species3_Mean
```

Msum *Minkowski sum*

Description

Given a list XX of polygonal fuzzy numbers the function first checks (1) if each element of the list is in the correct form (tested by checking) and (2) if the alpha-levels of all elements in the list coincide. If these two conditions are fulfilled the levelwise Minkowski-sum of all elements in the sample XX will be returned. If not the translator function can be used to transform the elements of the list in the correct format.

Usage

```
Msum(XX, pic = 0)
```

Arguments

XX	...list of polygonal fuzzy numbers (the function implicitly checks the conditions)
pic	...numeric, if pic=1 then the Minkowski-sum of XX is printed. By default pic=0.

Details

See examples

Value

Given input XX in the correct format the function returns the Minkowski sum of the polygonal fuzzy numbers contained in the list.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also checking, translator, Mmean

Examples

```
#Example 1:
X<-data.frame(x=c(0,1,1.5,3),alpha=c(0,1,1,0))
Y<-data.frame(x=c(1.25,2.75,2.75,5),alpha=c(0,1,1,0))
sum<-Msum(list(X,Y))
sum

#Example 2:
data(XX)
X<-translator(XX[[1]],50)
Y<-translator(XX[[2]],50)
Z<-translator(XX[[3]],50)
YY<-list(X,Y,Z)
M<-Msum(YY)
```

---

sc_mult *Minkowski scalar multiplication*

Description

Given an input dataframe (polygonal fuzzy number) X in the correct format (tested by checking), and a scalar b the fuzzy number bX is calculated using level-wise Minkowski scalar multiplication.

Usage

```
sc_mult(X, b, pic = 0)
```

Arguments

X	...dataframe, if checking(X)=1 the product bX is returned, if not, NA is returned.
b	...numeric
pic	...numeric, if pic=1 then the product of X by the scalar b is printed. By default pic=0.

Details

See examples

Value

Given correct input data, the function returns the polygonal fuzzy number bX.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also checking, translator

Examples

```
#Example 1:
U<-data.frame(x=c(-1,0,1),alpha=c(0,1,0))
E<-sc_mult(U,2)
E

#Example 2:
X<-data.frame(x=c(0,1,1,5),alpha=c(0,1,1,0))
sc_prod<-sc_mult(X,1.5)
sc_prod



#Example 3:
data(XX)
X<-translator(XX[[1]],10)
E<-sc_mult(X,-2,pic=1)
E
```

---

translator *Translator*

Description

The function first calls checking2 in order to check if the input data X defines a polygonal fuzzy number. If all conditions are satisfied the function transforms the input X into a dataframe with the chosen number nl of levels (default is nl=101) by doing linear interpolation, and returns this dataframe.

Usage

```
translator(X, nl = 101, pic = 0)
```

Arguments

X	...dataframe for which checking2 yields 1
nl	...number of levels of the output dataframe (fuzzy number), by default 101, must be at least 2.
pic	...numeric, if pic=1 then the poligonal fuzzy number X with nl alpha-levels is printed. By default pic=0.

Details

See examples

Value

In case checking2(X)=1 translator returns a dataframe (fuzzy number) with nl number of alpha-levels, otherwise the input is returned unchanged.

Note

In case you find (almost surely existing) bugs or have recommendations for improving the functions comments are welcome to the above mentioned mail addresses.

Author(s)

Wolfgang Trutschnig <wolfgang@trutschnig.net>, Asun Lubiano <lubiano@uniovi.es>

See Also

See Also as checking2, checking

Examples

```
#Example 1:
X<-data.frame(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,1,1,0))
E<-translator(X,3)
E


#Example 2:
data(XX)
E<-translator(XX[[3]],11, pic=1)
E
```

Trees *Tree dataset*

Description

Trees is a list containing three sublists, each of which consists of a sample of trapezoidal fuzzy numbers. The data corresponds to the *quality* of the three main species of trees in Asturias, namely birch (*Betula celtiberica*), sessile oak (*Quercus petraea*) and rowan (*Sorbus aucuparia*) within a study about the progress of reforestation in a given area of Asturias (Spain). INDUROT institute (University of Oviedo) has collected a sample of n1=133 birches, n2=109 sessile oaks and n3=37 rowans. Each tree was assigned a trapezoidal fuzzy number that models the experts subjective judgements/perceptions of the tree quality on a scale from 0 to 5 (0 meaning very bad quality to 5 meaning very good quality). Thereby the 1-cut is the interval in which the expert thinks the quality is contained and the support (0-cut) is the interval in which the expert is absolutely sure the quality is contained.

Usage

```
data("Trees")
```

Format

A list with three sublist, each of which contains trapezoidal fuzzy numbers.

Details

See Reference

References

[1] Colubi, A.: *Statistical inference about the means of fuzzy random variables: Applications to the analysis of fuzzy- and real-valued data*, Fuzzy Sets and Systems, 160(3), pp. 344-356 (2009)

Examples

```
data(Trees)
M<-Mmean(Trees[[1]],1)
M
sd<-sqrt(Bvar(Trees[[1]],1))
sd
```

XX *Example data*

Description

XX is a list of three polygonal fuzzy numbers that is used in the given examples.

Usage

```
data("XX")
```

Format

List of three polygonal fuzzy numbers.

Details

See examples

Examples

```
data(XX)
V<-translator(XX[[1]],50)
SS1<-vector("list",length=100)
for (j in 1:100){
 SS1[[j]]<-generator(V,)
 }
M<-Mmean(SS1,1)
head(M)
b<-bertoluzza(M,V,1/3,1)
b
```

Index