Package ‘SAFD’

February 15, 2013

Type Package

Title Statistical Analysis of Fuzzy Data

Version 0.4

Date 2012-08-10

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

Maintainer Wolfgang Trutschnig <wolfgang@trutschnig.net>

Depends R (>= 2.10.0)

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, Hukuhara difference, quantiles) as well as for calculating (Bertoluzza) distance, sample variance, sample covariance, sample correlation, and the Dempster-Shafer (levelwise) histogram. Moreover a function to simulate fuzzy random variables, bootstrap tests for the equality of means, and a function to do linear regression given trapezoidal fuzzy data is included. Version 0.4 fixes some bugs of version 0.3 and includes an additional function to calculate quantiles of samples of polygonal fuzzy numbers.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2012-08-10 13:46:59

NeedsCompilation no
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAFD-package</td>
<td>2</td>
</tr>
<tr>
<td>Bcor</td>
<td>4</td>
</tr>
<tr>
<td>Bcov</td>
<td>5</td>
</tr>
<tr>
<td>bertoluzza</td>
<td>7</td>
</tr>
<tr>
<td>btest.mean</td>
<td>9</td>
</tr>
<tr>
<td>btest2.mean</td>
<td>10</td>
</tr>
<tr>
<td>btestk.mean</td>
<td>12</td>
</tr>
<tr>
<td>Bvar</td>
<td>13</td>
</tr>
<tr>
<td>checking</td>
<td>15</td>
</tr>
<tr>
<td>checking2</td>
<td>17</td>
</tr>
<tr>
<td>decomposer</td>
<td>18</td>
</tr>
<tr>
<td>defuzzify</td>
<td>19</td>
</tr>
<tr>
<td>DSfrequency</td>
<td>20</td>
</tr>
<tr>
<td>DShistogram</td>
<td>22</td>
</tr>
<tr>
<td>Fmedian</td>
<td>24</td>
</tr>
<tr>
<td>Fquantile</td>
<td>25</td>
</tr>
<tr>
<td>generator</td>
<td>26</td>
</tr>
<tr>
<td>hukuhara</td>
<td>28</td>
</tr>
<tr>
<td>lrmodel</td>
<td>29</td>
</tr>
<tr>
<td>Mmean</td>
<td>31</td>
</tr>
<tr>
<td>Msum</td>
<td>32</td>
</tr>
<tr>
<td>sc_mult</td>
<td>33</td>
</tr>
<tr>
<td>translator</td>
<td>34</td>
</tr>
<tr>
<td>Trees</td>
<td>36</td>
</tr>
<tr>
<td>XX</td>
<td>37</td>
</tr>
</tbody>
</table>

---

### 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:** 0.4
- **Date:** 2012-08-10
- **License:** GPL (>=2)
- **LazyLoad:** yes
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, Hukuhara difference, quantiles) as well as for calculating (Bertoluzza) distance, sample variance, sample covariance, sample correlation, and the Dempster-Shafer (levelwise) histogram. Moreover a function to simulate fuzzy random variables, bootstrap tests for the equality of means, and a function to do linear regression given trapezoidal fuzzy data is included.

Version 0.4 fixes some bugs of version 0.3 and includes an additional function to calculate quantiles of samples of polygonal fuzzy numbers.

Author(s)

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

Maintainer: Wolfgang Trutschnig <wolfgang@trutschnig.net>

References


See Also

http://bellman.ciencias.uniovi.es/SMIRE/
Description

The sample correlation of two samples of polygonal fuzzy numbers with respect to the Bertoluzza distance is calculated. Given lists $XX$, $YY$ of polygonal fuzzy numbers the functions first checks (1) if each element of the two lists is in the correct form (tested by checking), (2) if the alpha-levels of all elements in the two lists coincide and (3) if the lists have the same length. If all conditions are fulfilled the Bertoluzza correlation will be returned. If not the \texttt{translator} function can be used to transform the elements of the lists in the correct format. For details see \cite{1} from below, and replace the kernel $K$ with the expression induced by the Bertoluzza metric. The parameter $\theta$ has to fulfill $\theta>0$, by default $\theta=1/3$.

Usage

\begin{verbatim}
Bcor(XX, YY, theta = 1/3)
\end{verbatim}

Arguments

- $XX$ ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as $YY$
- $YY$ ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as $XX$
- $theta$ ...numeric and $>0$, see \texttt{bertoluzza}

Details

See examples. The code makes use of \texttt{Bcov} and \texttt{Bvar}.

Value

Given input $XX$, $YY$ in the correct format the function returns the Bertoluzza correlation of the samples $XX$, $YY$.

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

See Also

See Also as checking, Bcov, Bvar, bertoluzza

Examples

# Example 1:
data(XX)
V<-translator(XX[[3]],100)
XX<-vector("list",length=10)
YY<-XX
  for(i in 1:10){
    XX[[i]]<-generator(V,,)
    YY[[i]]<-XX[[i]]
    YY[[i]]$x<-5*YY[[i]]$x+1
  }
b<-Bcor(XX,YY,1/3)
b

# Example 2:
data(XX)
V<-translator(XX[[3]],51)
XX<-vector("list",length=200)
YY<-XX
  for(i in 1:200){
    XX[[i]]<-generator(V,,)
    YY[[i]]<-XX[[i]]
    YY[[i]]$x<-5*YY[[i]]$x+1
  }
b<-Bcor(XX,YY,1/3)
b

Bcov

(Sample) Covariance

Description

The sample covariance of two samples of polygonal fuzzy numbers with respect to the Bertoluzza distance is calculated. Given lists XX, YY of polygonal fuzzy numbers the functions first checks (1) if each element of the two lists is in the correct form (tested by checking), (2) if the alpha-levels of all elements in the two lists coincide and (3) if the lists have the same length. If all conditions are fulfilled the Bertoluzza covariance will be returned. If not the translator function can be used to transform the elements of the lists 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

Bcov(XX, YY, theta = 1/3)
Arguments

XX ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as YY

YY ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as XX

theta ...numeric and >0, see bertoluzza

Details

See examples

Value

Given input XX, YY in the correct format the function returns the Bertoluzza covariance of the samples XX, YY.

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


See Also

See Also as checking, Mmean, lrmodel, Bcor, Bvar

Examples

#Example 1:
data(XX)
V<-translator(XX[3],1/zero.noslash/zero.noslash)
XX<-vector("list",length=1/zero.noslash)
YY<-XX
for(i in 1/zero.noslash){
   XX[i]<-generator(V,,)
   YY[i]<-XX[i]
   YY[i]$x<-5*YY[i]$x+1
}
b<-Bcov(XX,YY,1/3)
b

#Example 2:
```r
data(XX)
V<-translator(XX[[3]],51)
XX<-vector("list",length=500)
YY<-XX
for(i in 1:500){
  XX[[i]]<-generator(V,)
  YY[[i]]<-XX[[i]]
  YY[[i]]$x<-5*YY[[i]]$x+1
}
b<-Bcov(XX,YY,1/3)
b
b<-Bcor(XX,YY,1/3)
b
```

---

### 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**

```r
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 fulfill the conditions tested implicitly by checking 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

See Also
See Also as Mmean, checking

Examples
#
data(XX)
X<-translator(XX[[1]],5/zero.noslash)
Y<-translator(XX[[2]],5/zero.noslash)
Z<-translator(XX[[3]],5/zero.noslash)
ZZ<-list(X,Y,Z)
b<-bertoluzza(X,Y,1/3,1)
b

#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

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

#
X<-data.frame(x=c(0,1,2),alpha=c(0,1,0))
Y<-data.frame(x=c(0,1,2),alpha=c(0,1,0))
b<-bertoluzza(X,Y,1/3,1)
**btest.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 Bertoluzzadi
distance from the sample mean to \(V\) as test-statistic. Afterwards for \(B\) bootstrap samples the (boot-
strap) 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 in one window and the ecdf of the bootstrap statistic in another one.
For detailed explanation see papers [1] and [2] below.

**Usage**

```r
btest.mean(XX, V, B = 1000, pic=1)
```

**Arguments**

- **XX** ...list of polygonal fuzzy numbers (the functions implicitly checks the condi-
tions).
- **V** ...polygonal fuzzy number that is tested to be the mean of the FRV.
- **B** ...integer, by default \(B=1000\).
- **pic** ...numeric, if \(pic=1\) sample mean and \(V\) are plotted in one window and the ecdf
  of the bootstrap statistic in another one.

**Details**

See examples

**Value**

Given input \(XX\) and \(V\) in the correct format, the function returns the p-value of the two-sided boot-
strap test that the expectation is \(V\).

**Note**

The function is quite slow - we will try to solve this problem in the next versions of the package.
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>
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 in one window and the ecdf of the bootstrap statistic in another one, otherwise no plot is produced. For detailed explanation see the papers [1] and [2] below.

Usage
btest2.mean(XX, YY, B = 1000, pic = 1)

References

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

Examples
#run for bigger sample sizes:
data(XX)
V<-translator(XX[[3]],5/zero.noslash)
V2<-V
SS<-vector("list",length=5/zero.noslash)
for (j in 1:50){
  SS[[j]]<-generator(V2,)
}
b<-btest.mean(SS,V2,1/zero.noslash)

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

B
...integer, by default B=1000.

pic
...numeric, if pic=1 then the sample means of XX and YY are plotted in one window and the ecdf of the bootstrap statistic in another one. 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 - we will try to solve this problem in the next versions of the package. 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


See Also

See Also as Mmean, Bvar, bertoluzza

Examples

#run for bigger B
data(XX)
X<-translator(XX[[1]],2/zero.noslash)
Y<-translator(XX[[2]],2/zero.noslash)
XX<-vector("list",length=3/zero.noslash)
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,10)
b

#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 in one window and the ecdf of the bootstrap statistic in another one, otherwise no plot is produced. For a more detailed explanation see the papers [1] and [2] below.

Usage

btestk.mean(XXX, sel, B = 50, 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
B ...integer, by default B=50.
pic ...numeric, if pic=1 then the sample means of the via sel selected samples from XXX and the total mean are plotted in one window and the ecdf of the bootstrap statistic in another one. 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 - we will try to solve this problem in the next versions of the package. 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


See Also

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

Examples

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

#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 (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 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


See Also

See Also as bertoluzza, Mmean, Bcov, Bcor

Examples

```r
#Example 1:
data(XX)
X<-translator(XX[[1]],5/zero.noslash)
Y<-translator(XX[[2]],5/zero.noslash)
Z<-translator(XX[[3]],5/zero.noslash)
YY<-list(X,Y,Z)
A<-Bvar(YY,1)
A
```

```r
#Example 2:
```
data(XX)
v<-Bvar(list(XX[[1]],XX[[1]]),1/3)
v
#Example 3:
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.3,0,1,0,0.3)))
Y<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.3,1,0,0.3)))
Z<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,1,1,0)))
ZZ<-list(X,Y,Z)
v<-Bvar(ZZ,1)
v
#Example 4
data(XX)
A<-Bvar(XX,1)
A

---

**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-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

data(XX)
a<-checking(XX[[1]],1)
a
#
X<-data.frame(cbind(y=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,0.9,0.9,0)))
a<-checking(X)
a
#
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,0.9,0.9,0)))
a<-checking(X)
a
#
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,-0.5,1),alpha=c(0,0.6,1,1,0)))
a<-checking(X)
a
#
Y<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.3,0,1,0,0.3)))
a<-checking(Y)
a
#
Z<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.6,1,1,0)))
a<-checking(Z)
a
#
U<-data.frame(cbind(x=c(-1,0,1),alpha=c(0,1,0)))
a<-checking(U)
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-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
U<-data.frame(cbind(x=c(-1,0,1),alpha=c(0,1,0)))
#a<-checking(U,)
a<-checking2(U,)
a
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 $(\text{nrow}(X)+1)$ 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

- $\text{comp1}$ In case checking($X$)=1 decomposer returns a dataframe with $(\text{nrow}(X)+1))$-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


See Also

See also generator, checking
**defuzzify**

**Examples**

```r
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**

```r
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

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

#
X <- data.frame(cbind(x=c(-2, -0.75, -0.25, 0.5, 1), alpha=c(0.3, 0, 1, 0, 0.3)))
Y <- data.frame(cbind(x=c(-2, -0.75, -0.25, 0.5, 1), alpha=c(0.03, 1, 0, 0, 0.3)))
Z <- data.frame(cbind(x=c(-2, -0.75, -0.25, 0.5, 1), alpha=c(0, 0.6, 1, 1, 0)))
FF <- list(Z, X, Y)
a <- defuzzify(FF)
a
RR <- vector("list", length=1503)
RR[1:3] <- FF
RR[4:1503] <- SS
a <- defuzzify(RR)
a
RR <- vector("list", length=1501)
RR[2:1501] <- SS
a <- defuzzify(RR)
a
```

---

### DSfrequency

**Levelwise Dempster-Shafer frequency**

**Description**

Given a sample `XX` of polygonal fuzzy numbers and an interval `IV` the levelwise Dempster-Shafer frequency of the interval is calculated, i.e. for the chosen number `nl` of equidistant alpha-cuts it is checked how many of the elements of the sample have an alpha-cut that is contained in the interval (lower frequency) and how many have an alpha-cut hitting the interval (upper frequency). These family of intervals is afterwards aggregated to another polygonal fuzzy number with the corresponding number of alpha-cuts, which is returned. For details see [1] below. Preliminary the input data is tested for the correct format using the checking function.

**Usage**

```r
DSfrequency(XX, IV = c(0, 1), pic = 1, nl = 101)
```
**DSfrequency**

**Arguments**

- **XX** ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions)
- **IV** ...numeric vector of length two, by default IV=c(0,1)
- **pic** ...numeric, in case pic=1 the frequency is plotted, otherwise no plot is produced
- **nl** ...number of equidistant alpha-levels, by default nl=101

**Details**

See examples

**Value**

Given correct input data, the function returns the levelwise Dempster-Shafer frequency of the chosen interval (again in the correct form of a polygonal fuzzy number).

**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**


**See Also**

See Also as DShistogram

**Examples**

```r
# Example 1:
data(XX)
V<-XX[[1]]
SS<-vector("list",length=500)
for (j in 1:500){
  SS[[j]]<-generator(V,)
}
A<-DSfrequency(SS,c(1,3),1,100)
head(A)
```
**Description**

Based on a sample $XX$ of polygonal fuzzy numbers (tested by checking), a chosen interval $\text{limx}$, a chosen number $\text{npart}$ of partitions elements and a chosen number $\text{nl}$ of equidistant alpha-levels the levelwise Dempster-Shafer frequency for each partition element is calculated. If $\text{pic}=\text{TRUE}$ then a 3d plot as well as an image-plot of the histogram is produced. In case of $\text{pdf}=\text{TRUE}$ a pdf containing these plots is produced automatically.

**Usage**

```
DShistogram(XX, limx = NA, npart = 10, nl = 101, pic = TRUE, pdf = FALSE)
```

**Arguments**

- $XX$ ...list of polygonal fuzzy numbers (the function implicitly checks the conditions)
- $\text{limx}$ ...numeric vector of length two, by default $\text{limx}=\text{c}(0,1)$, that determines the x-range for which the histogram is plotted
- $\text{npart}$ ...integer, number of partitions elements
- $\text{nl}$ ...number of equidistant alpha-level, by default $\text{nl}=101$
- $\text{pic}$ ...if $\text{pic}=1$, a 3d- and an image-plot of the histogram is produced
- $\text{pdf}$ ...if $\text{pdf}=1$, a 3d- and an image-plot of the histogram is automatically exported as pdf, by default $\text{pdf}=\text{FALSE}$

**Details**

See examples

**Value**

If the input data is in the correct form the function returns a list with the following elements:

- $\text{gridx}$ x-grid for plotting the histogram
- $\text{gridy}$ y-grid for plotting the histogram
- $M$ values of the histogram as function on the grid
- $\text{breaks}$ breaks of the histogram

**Note**

Speed to be improved in future versions of the package,
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


See Also

See Also as Dsfrequency

Examples

# an example with a random variable with small spread of the expectation
# run for bigger sample size and finer partition
data(XX)
V<-translator(XX[[3]],30)
V2<-V
V2$x<-V$x/5
SS<-vector("list",length=300)
for (j in 1:300){
  SS[[j]]<-generator(V2,)
}
A<-DShistogram(SS,c(-3,3),npart=6,nl=51)

# run for bigger sample size and finer partition
data(XX)
V<-translator(XX[[3]],30)
V2<-V
V2$x<-V$x/10
pertV<-list(dist="unif",par=c(-2,2))
SS<-vector("list",length=300)
for (j in 1:300){
  SS[[j]]<-generator(V2,pertV,)
}
A<-DShistogram(SS,npart=5,nl=51)

# takes some time but produces nice result
# data(XX)
# V<-translator(XX[[3]],30)
# V2<-V
# V2$x<-V$x/10
# pertV<-list(dist="unif",par=c(-2,2))
# pertL<-list(dist="lnorm",par=c(-2,2))
# SS<-vector("list",length=1000)
# for (j in 1:1000){
#  SS[[j]]<-generator(V2,pertV,pertL,)
# }
# A<-DShistogram(SS,npart=15,nl=51)
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).

Usage

\[\text{Fmedian}(XX, \text{pic} = 1)\]

Arguments

\(XX\) ...list of polygonal fuzzy numbers with the same alpha levels (the functions implicitly checks the conditions)

\(\text{pic}\) ...numeric, if \(\text{pic}==1\) the sample, its mean and its median are plotted.

Details

See examples.

Value

Given correct input \(XX\) the function returns a 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>

See Also

See Also as \texttt{Fquantile}
Fquantile

Examples

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

Fquantile Quantiles 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 chosen (levelwise) quantiles. As in the basic R-
function quantile a vector p of quantiles can be calculated. The functions returns a list of quantiles
(which are, by definition, fuzzy numbers too).

Usage

Fquantile(XX, p = c(0.25, 0.5, 0.75, 1))

Arguments

XX ...list of polygonal fuzzy numbers with the same alpha levels (the functions implicitely checks the conditions)
p ...numeric vector of probabilities with values in [0,1]

Details

See examples.

Value

Given correct input XX the function returns a list of quantiles.

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>
### 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 \( \text{decomposer}(V) \) is called. Loosely speaking, the dataframe \( Y \) returned by \( \text{decomposer} \), 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 0.1 only a few choices are possible: (1) The perturbation of the centre of the 1-cut \( \text{pertV} \) has to be of the form \( \text{norm}(0, \sigma) \) or \( \text{unif}(-a, a) \), \( \sigma, a > 0 \). (2) The perturbation of the left part of the fuzzy set \( \text{pertL} \) has to be of the form \( \chi^2(1) \), \( \exp(1) \) or \( \text{lnorm}(a, b) \) with expectation=1. (3) The perturbation of the right part of the fuzzy set \( \text{pertR} \) has to be of the same form as that for the left part.

#### Usage

```r
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


See Also

See Also as decomposer, checking

Examples

```r
data(XX)
V<-translator(XX[[3]],100)
YY<-vector("list",length=100)
for(i in 1:100){
   YY[[i]]<-generator(V,)
}
M<-Mmean(YY)
M
plot(M,type="l",xlim=c(-3,4))
lines(V,type="l",col="red",lwd=2)
```
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 $\text{pic}=1$ then the input and the Hukuhara difference is plotted, otherwise no plot is produced.

Usage

```r
hukuhara(X, Y, pic = 1)
```

Arguments

- **X**: ...polygonal fuzzy number (tested by checking)
- **Y**: ...polygonal fuzzy number (tested by checking)
- **pic**: ...numeric, if $\text{pic}=1$ 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, 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>

See Also

See Also as `checking`, `lrmodel`
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(0,0,0,1.5,2,2),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))
h <- hukuhara(X,Y,1)
h
h <- hukuhara(Z,Y,1)
h

#Example 2:
Y <- data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.3,0,1,0,0.3)))
X <- data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,1,1,0)))
h <- hukuhara(X,Y)
h

#Example 3: in this case the hukuhara diff has to exist by construction
data(XX)
X <- translator(XX[[1]],5)
shift <- seq(-1,1,length=1)
Y <- X
Y$x <- X$x + shift
h <- hukuhara(X,Y,1)

#Example 4: in this case the hukuhara diff has to exist by construction
data(XX)
X <- translator(XX[[1]],5)
shift <- seq(-1,1,length=1)
Y <- X
Y$x <- X$x + shift
H <- hukuhara(X,Y,1)
H

lrmodel

Linear Regression model

Description

The linear regression model Y=aX+eps for trapezoidal fuzzy numbers as described in [1] has been implemented, whereby the least-squares-minimization (with constraints) is done with respect to the Bertoluzza metric (with theta=1/3). Given lists XX, YY of polygonal fuzzy numbers the functions first checks (1) if each element of the two list is in the correct form (tested by checking), (2) if the alpha-levels of all elements in the two lists coincide and (3) if the lists have the same length. If all conditions are fulfilled the function automatically converts the fuzzy numbers in XX and YY in trapezoidal ones and returns the estimations for the parameters a and B - in case of double solutions both solutions are returned.
Usage

lrmodel(XX, YY)

Arguments

XX ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as YY

YY ...list of polygonal fuzzy numbers (the functions implicitly checks the conditions) having the same length as XX

Details

See examples

Value

Given input XX, YY in the correct format the function returns a list containing the estimates for a and b (in case of double solutions both solutions are 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


See Also

See Also as checking, Mmean, hukuhara, Bvar, Bcov, bertoluzza

Examples

#Example 1 (crisp case)
XX<-vector("list",length=2)
XX[[1]]<-data.frame(cbind(x=c(1,1,1),alpha=c(0,1,1)))
XX[[2]]<-data.frame(cbind(x=c(2,2,2),alpha=c(0,1,1)))
YY<-list(length=2)
YY[[1]]<-data.frame(cbind(x=c(1,1,1),alpha=c(0,1,1)))
YY[[2]]<-data.frame(cbind(x=c(2,2,2),alpha=c(0,1,1)))
m<-lrmodel(XX,YY)
m
# Example 2:
data(XX)
V <- translator(XX[[3]], 100)
XX <- vector("list", length = 50)
YY <- XX
for (i in 1:50){
  XX[[i]] <- generator(V,,)
  YY[[i]] <- XX[[i]]
  YY[[i]]$x <- 5 * YY[[i]]$x + 1
}
m <- lrmodel(XX, YY)
m

---

**Mmean**

_Minkowski mean_

**Description**

Given a sample XX of polygonal fuzzy numbers the Minkowski-mean of the sample is calculated. The function first calls `Msum` to check if XX has the correct format and, in case yes, `sc_mult` 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, Msum, sc_mult, translator

Examples

data(XX)
A<-Mmean(XX,1)
A

X<-translator(XX[[1]],5/zero.noslash)
Y<-translator(XX[[2]],5/zero.noslash)
Z<-translator(XX[[3]],5/zero.noslash)
YY<-list(X,Y,Z)
A<-Mmean(YY,pic=1)
A
#
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.3,0.1,0.0.3)))
Y<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.0.3,1,0.0.3)))
Z<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0.0.6,1,1,0)))
ZZ<-list(X,Y,Z)
M<-Mmean(ZZ)
M

Msum

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)

Arguments

XX ...list of polygonal fuzzy numbers (the function implicitly checks the conditions)

Details
See examples
sc_mult

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, Mmean, translator

Examples
#
data(XX)
X<-translator(XX[[1]],5/zero.noslash)
Y<-translator(XX[[2]],5/zero.noslash)
Z<-translator(XX[[3]],5/zero.noslash)
YY<-list(X,Y,Z)
M<-Msum(YY)
M

#
X<-data.frame(cbind(x=c(-2,-/zero.noslash.75,-/zero.noslash.25,/zero.noslash.5,1),alpha=c(/zero.noslash.3,/zero.noslash,1,/zero.noslash,/zero.noslash.3)))
Y<-data.frame(cbind(x=c(-2,-/zero.noslash.75,-/zero.noslash.25,/zero.noslash.5,1),alpha=c(/zero.noslash,/zero.noslash.3,1,/zero.noslash,/zero.noslash.3)))
Z<-data.frame(cbind(x=c(-2,-/zero.noslash.75,-/zero.noslash.25,/zero.noslash.5,1),alpha=c(/zero.noslash,/zero.noslash.6,1,1,/zero.noslash)))
ZZ<-list(X,Y,Z)
M<-Msum(ZZ)

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)
Arguments

X  ...dataframe, if checking(X)=1 the product bX is returned, if not, NA is returned.
b  ...numeric

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

Examples

data(XX)
X<-translator(XX[[1]],1/zero.noslash)
E<-sc_mult(X,-2)
E

U<-data.frame(cbind(x=c(-1,/zero.noslash,1),alpha=c(/zero.noslash,1,/zero.noslash)))
E<-sc_mult(U,2)
E

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)

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.

Details

See examples

Value

comp1 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

#example1:
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.3,1,0,0.3)))
E<-translator(X,4)
E

#example2
X<-data.frame(cbind(x=c(-2,-0.75,-0.25,0.5,1),alpha=c(0,0.6,1,1,0)))
E<-translator(X,3)
head(E)
plot(X,type="l")
lines(E,type="l",col="red")

#example3
data(XX)
E<-translator(XX[[3]],11)
plot(XX[[3]],type="l")
lines(E,type="l",col="red")
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) the 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


Examples

data(Trees)
M <- Mmean(Trees[[1]], 1)
head(M)
M <- Mmean(Trees[[1]], 1)
head(M)
Mmean(Trees[[1]], 1)
head(M)
**Description**

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

**Usage**

```r
data(XX)
```

**Format**

List of three polygonal fuzzy numbers

**Details**

See examples

**Examples**

```r
data(XX)
V<-translator(XX[[1]],5/zero.noslash)
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

*Topic arith
  hukuhara, 28
  Msum, 32
  sc_mult, 33
*Topic attribute
  checking, 15
  checking2, 17
  decomposer, 18
  translator, 34
*Topic classes
  checking, 15
  checking2, 17
  decomposer, 18
  translator, 34
*Topic datagen
  defuzzify, 19
  DSfrequency, 20
  DShistogram, 22
  generator, 26
*Topic datasets
  Trees, 36
  XX, 37
*Topic dplot
  DShistogram, 22
*Topic ecdf
  Fmedian, 24
  Fquantile, 25
*Topic htest
  btest.mean, 9
  btest2.mean, 10
  btestk.mean, 12
*Topic list
  Trees, 36
*Topic manip
  bertoluzza, 7
  Bvar, 13
  defuzzify, 19
  hukuhara, 28
  Mmean, 31
  Msum, 32
  sc_mult, 33
*Topic math
  bertoluzza, 7
*Topic models
  lrmmodel, 29
*Topic nonparametric
  generator, 26
*Topic package
  SAFD-package, 2
*Topic quantiles
  Fmedian, 24
  Fquantile, 25
*Topic regression
  Bcor, 4
  Bcov, 5
  DSfrequency, 20
  lrmmodel, 29
*Topic univar
  Bcor, 4
  Bcov, 5
  btest.mean, 9
  btest2.mean, 10
  btestk.mean, 12
  Bvar, 13
  Mmean, 31
  Bcor, 4, 6, 14
  Bcov, 5, 5, 14, 30
  bertoluzza, 5, 7, 10, 11, 13, 14, 30
  btest.mean, 9, 13
  btest2.mean, 10, 10, 13
  btestk.mean, 10, 12
  Bvar, 5, 6, 10, 11, 13, 13, 30
  checking, 5, 6, 8, 15, 17, 18, 27, 28, 30, 32–35
  checking2, 16, 17, 35
  decomposer, 18, 27
  defuzzify, 19
INDEX

DSfrequency, 20, 23
DShistogram, 21, 22

Fmedian, 24, 26
Fquantile, 24, 25

generator, 18, 26

hukuhara, 28, 30

lrmodel, 6, 28, 29

Mmean, 6, 8, 10, 11, 13, 14, 26, 30, 31, 33
Msum, 32, 32

SAFD (SAFD-package), 2
SAFD-package, 2
sc_mult, 32, 33

translator, 16, 17, 32, 33, 34
Trees, 36

XX, 37