Package 'Calculator.LR.FNs'

May 2, 2018

Type Package

Title Calculator for LR Fuzzy Numbers

Version 1.3

Date 2018-05-01

Author Abbas Parchami (Department of Statistics, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran)

Maintainer Abbas Parchami <parchami@uk.ac.ir>

Description Arithmetic operations scalar multiplication, addition, subtraction, multiplication and division of LR fuzzy numbers (which are on the basis of extension principle) have a complicate form for using in fuzzy Statistics, fuzzy Mathematics, machine learning, fuzzy data analysis and etc. Calculator for LR Fuzzy Numbers package relieve and aid applied users to achieve a simple and closed form for some complicated operator based on LR fuzzy numbers and also the user can easily draw the membership function of the obtained result by this package.

License LGPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-02 12:04:22 UTC

R topics documented:

`Calculator.LR.FNs-package`
Calculator for LR Fuzzy Numbers

Description

Calculator for LR Fuzzy Numbers package, i.e. `Calculator.LR.FNs` package, is an open source (LGPL 3) package for R which provides the generalized four arithmetic operations $+$, $-$, $\times$ and $\div$ on LR fuzzy numbers. Arithmetic operations addition and subtraction are based on Zadeh extension principle. Also the scalar multiplication of a real number into a LR fuzzy number is considered in this package on the basis of Zadeh extension principle. Although the class of LR fuzzy numbers is not theoretically closed under the operations $\times$ and $\div$, but we apply from approximation for multiplication and division of LR fuzzy numbers which lead the users to a LR fuzzy numbers. `Calculator.LR.FNs` package make it easier for researchers, students and any other interested people about fuzzy Mathematics to experience this with a simple calculator. Function `LRFN.plot` is designed in `Calculator.LR.FNs` package for potting the membership function of any LR fuzzy number.

Details

If the Operation has NOT a closed form or is not defined as a LR fuzzy number, one can continue calculations by `FuzzyNumbers` package to achive a the figure of membership function of final result using cuts of the final result.

Author(s)

Abbas Parchami

Maintainer: Abbas Parchami <parchami@uk.ac.ir>

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

See Also

FuzzyNumbers

Examples

```
# Example 1: mean of LR FNs
Left.fun  = function(x)  { (1-x)*(x>=0)}
A = L(6, 1, 2)
B = L(2, 4, 1)
LRFN.plot( A, xlim=c(-3,9), ylim=c(0,1.2), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot(  s.m( 0.5 , s(A,B) ), lwd=2, lty=3, col=1, add=TRUE)
     # ploting the mean of A and B
legend( "topright", c("A = L(6, 1, 2)", "B = L(2, 4, 1)", "(A + B) / 2 = L(4, 2.5, 1.5)")
     , col = c(2, 3, 1), text.col = 1, lwd = c(2,2,2), lty = c(2, 2, 3) )

# Example 2: Compute and ploting  {0.5(A+B)}*A  where A and B are two LR FNs
LRFN.plot( A, xlim=c(-3,41), ylim=c(0,1), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( m( s.m( 0.5 , s(A,B) ) , A ) , lwd=2, lty=3, col=1
     , add=TRUE)  # ploting the mean of A and B
legend( "topright", c("A = L(6, 1, 2)", "B = L(2, 4, 1)", "{(A + B) / 2} * A = L(24, 19, 17)")
     , col = c(2, 3, 1), text.col = 1, lwd = c(2,2,2), lty = c(2, 2, 3) )

# Example 3: The mean of n=10 random LR fuzzy numbers
n = 10
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
xlim=c(2, 18)
ylim=c(0, 1.15)
sum_x = c(0,0,0,0)

for (i in 1:n)
 {
  x = rnorm(1,10,3)
  x_l = runif(1,0,3)
  x_r = runif(1,0,2)
  X = c()
  X = LR(x, x_l, x_r)
  LRFN.plot( X, xlim=xlim, ylim=ylim, lwd=1, lty=1, col=1, add = (i != 1) )
  sum_x = a( sum_x , X )
 }

sum_x
X_bar = s.m( (1/n) , sum_x )
LRFN.plot( X_bar , lwd=2, lty=2, col=2, add = TRUE )
legend( "topright", c("LR FNs", "mean of LR FNs"), col = c(1, 2), text.col = 1
     , lwd = c(1, 2), lty = c(1, 2) )


# Example 4:
Left.fun = function(x)  { (1-x^2)*(x>=0)}
```

```
Right.fun = function(x)  { (1-x)*(x>=0)}

A = LR(2, 0.5, 1)
B = LR(1, 0.1, 0.6)
C = RL(3, 0.7, 1.5)
D = LR(3, 0.5, 0.3)

m(A,B)
s.m( 1.2 , m(A,B) )
d( s.m( 1.2 , m(A,B) ) , C)
m( d( s.m( 1.2 , m(A,B) ) , C) , D)

LRFN.plot( A, xlim=c(-0.2,6), ylim=c(0,1.75), lwd=2, lty=1, col=1)
LRFN.plot( B, lwd=2, lty=1, col=2, add=TRUE)
LRFN.plot( C, lwd=2, lty=1, col=3, add=TRUE)
LRFN.plot( D, lwd=2, lty=1, col=4, add=TRUE)

LRFN.plot( m(A,B), lwd=2, lty=2, col=5, add=TRUE)
LRFN.plot( s.m( 1.2 , m(A,B) ), lwd=2, lty=3, col=6, add=TRUE)
LRFN.plot( d( s.m( 1.2 , m(A,B) ) , C), lwd=2, lty=4, col=7, add=TRUE)
LRFN.plot( m( d( s.m( 1.2 , m(A,B) ) , C) , D), lwd=2, lty=5, col=8, add=TRUE)

legend( "topright", c("A = LR(2, 0.5, 1)", "B = LR(1, 0.1, 0.6)", "C = RL(3, 0.7, 1.5)"
     , "D = LR(3, 0.5, 0.3)", "A * B = LR(2, 0.7, 2.2)", "1.2 (A * B) = LR(2.4, 0.84, 2.4)"
     , "{1.2 (A * B)} / C = LR(0.8, 0.68, 1.067)", "[{1.2 (A * B)} / C] * D = LR(2.4, 2.44, 3.44)")
     , col = c(1:8), text.col = 1, lwd = c(2,2,2,2,2,2,2,2), lty = c(1, 1, 1 ,1 , 2, 3, 4, 5) )


# Example 5:
Left.fun = function(x)  { (1-x^3)*(x>=0)}
Right.fun = function(x)  { (1-x)*(x>=0)}

A = LR(5, 0.5, 1)
B = LR(2, 0.3, 0.6)
C = RL(1, 0.7, 1.5)
D = LR(0.5, 0.5, 1)

E = s.m(a(A,B), 1/2) # The mean of A and B
F = s(s.m(a(A,B), 1/2), C)
G = m(F,D)

LRFN.plot( A, xlim=c(-1,6), ylim=c(0,1.5), lwd=3, lty=1, col=1)
LRFN.plot( B, lwd=3, lty=1, col=2, add=TRUE)
LRFN.plot( C, lwd=3, lty=1, col=3, add=TRUE)
LRFN.plot( D, lwd=3, lty=1, col=4, add=TRUE)

LRFN.plot( E, lwd=3, lty=2, col=5, add=TRUE)
LRFN.plot( F, lwd=3, lty=3, col=6, add=TRUE)
LRFN.plot( G, lwd=3, lty=4, col=7, add=TRUE)

legend( "topleft", c("A = LR(5, 0.5, 1)", "B = LR(2, 0.3, 0.6)", "C = RL(1, 0.7, 1.5)",
   "D = LR(0.5, 0.5, 1)", "(A + B)/2 = LR(3.5, 0.4, 0.8)", "[(A + B)/2] - C = LR(2.5, 1.9, 1.5)",
    "{[(A + B)/2] - C} * D  = LR(1.25, 2.2, 3.2)" ),  col = c(1:7), text.col = 1,
```

```
lwd = c(2,2,2,2,2,2,2), lty = c(1, 1, 1, 1 , 2, 3, 4), bty = "n" )
```

---

addition *Addition of two LR fuzzy numbers*

Description

This function calculates the addition (summation) of two LR fuzzy numbers $M = (m, \alpha, \beta)_{LR}$ and $N = (n, \delta, \gamma)_{LR}$ on the basis of Zadeh extension principle by the following formula:

$$M \oplus N = (m + n, \alpha + \delta, \beta + \gamma)_{LR}$$

Usage

```
a(M, N)
```

Arguments

M	The first LR (or RL or L) fuzzy number
N	The second LR (or RL or L) fuzzy number

Value

A LR (or RL or L) fuzzy number

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

Examples

```
# Example 1:
Left.fun  = function(x)  { (1/(1+x^2))*(x>=0)}
Right.fun = function(x)  { (1/(1+(2*abs(x))))*(x>=0)}
M = LR(1, 0.6, 0.2)
N = LR(3, 0.5, 1)
a(N, M)

# commutative property for addition on LR fuzzy numbers (Jabejaei)
P = RL(5, 0.1, 0.3)
a(N, P)
a(P, P)

# associative property for addition on LR fuzzy numbers (Sherekat-paziri)
a(N, a(M, M))
a(a(N, M), M)

# Example 2:
A = LR(2, 1, 3)
B = LR(3, 1.2, 1.8)
LRFN.plot( A, xlim=c(-3,12), ylim=c(0,1.25), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=1, col=5, add=TRUE)
LRFN.plot( a(A, B), lwd=2, col=1, add=TRUE)
legend( "topright", c("A = LR(2, 1, 3)", "B = LR(3, 1.2, 1.8)", "A + B = LR(5, 2.2, 4.8)")
     , col = c(2, 5, 1), text.col = 1, lwd = c(2,2,2), lty = c(2, 1, 1) )


## The function is currently defined as
function (M, N)
{
    options(warn = -1)
    if (messages(M) != 1) {
        return(messages(M))
    }
    if (messages(N) != 1) {
        return(messages(N))
    }
    if (M[4] != N[4]) {
        return(noquote(paste0("Addition has NOT a closed form of a LR fuzzy number")))
    }
    else {
        a1 = M[1] + N[1]
        a2 = M[2] + N[2]
        a3 = M[3] + N[3]
        a4 = (M[4] + N[4])/2
        print(noquote(paste0("the result of addition is  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
```

```
            }
            else {
                paste0(" L")
            }))
        return(invisible(c(a1, a2, a3, a4)))
    }
}
```

`division` *Division of two LR fuzzy numbers*

Description

This function calculates the division of two LR fuzzy numbers. Although on the basis of Zadeh extension principle, the class of LR fuzzy numbers is not closed under the operations multiplication and division, but we consider the following approximation for division of fuzzy number $M = (m, \alpha, \beta)_{LR}$ by fuzzy number $N = (n, \gamma, \delta)_{RL}$ to work easy in the class of LR fuzzy numbers:

$$M \oslash N \simeq \left(\frac{m}{n}, \frac{m\delta + n\alpha}{n^2}, \frac{m\gamma + n\beta}{n^2}\right)_{LR}$$

Usage

```
d(M, N)
```

Arguments

`M`	The first LR (or RL or L) fuzzy number
`N`	The second LR (or RL or L) fuzzy number

Value

A LR (or RL or L) fuzzy number

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

Examples

```
# Example 1:
Left.fun  = function(x)  { (1-x)*(x>=0)}
A = L(6, 1, 2)
B = L(3, 2, 3)
xlim=c(-1.5,9)
LRFN.plot( A, xlim=xlim, lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( d(A,B), lwd=2, lty=1, col=1, add=TRUE)
legend( "topright", c("A = L(6, 1, 2)", "B = L(3, 2, 3)", "A / B = L(2, 2.33, 2)")
     , col = c(2, 3, 1), text.col = 1, lwd = c(2,2,2), lty = c(2, 2, 1) )


# Example 2:
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun = function(x)  { (1-x^2)*(x>=0)}
A = LR(8, 0.5, 1)
B = RL(2, 1, 1.5)

d(A,B)
d(LR(8, 0.5, 1), RL(2, 1, 1.5))

d(A,A)

d(d(A,B),B)
d(A,d(B,B))

C = LR(-3, 0.5, 1)
d(A,C)
LRFN.plot( A, xlim=c(-3,9.5), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( d(A,B), lwd=2, lty=3, col=4, add=TRUE)
LRFN.plot( d(d(A,B),B), lwd=2, lty=4, col=1, add=TRUE)
legend( "topleft", c("A = LR(8, 0.5, 1)", "B = RL(2, 1, 1.5)", "A / B = LR(4, 3.25, 2.5)"
    , "(A / B) / B = LR(2, 3.125, 2.25)"), col = c(2, 3, 4, 1), text.col = 1, lwd = c(2,2,2)
     , lty = c(2, 2, 3, 4) )


## The function is currently defined as
function (M, N)
{
    options(warn = -1)
```

```
    if (messages(M) != 1) {
        return(messages(M))
    }
    if (messages(N) != 1) {
        return(messages(N))
    }
    if ((M[4] == 1 & N[4] == 0) | (M[4] == 0 & N[4] == 1) | (M[4] ==
        0.5 & N[4] == 0.5)) {
        if ((sign(M) == "Positive") & (sign(N) == "Positive")) {
            a1 = M[1]/N[1]
            a2 = ((M[1] * N[3]) + (N[1] * M[2]))/(N[1]^2)
            a3 = ((M[1] * N[2]) + (N[1] * M[3]))/(N[1]^2)
            a4 = M[4]
            print(noquote(paste0("the result of division is approximately  (core = ",
                a1, ", left spread = ", a2, ", right spread = ",
                a3, ")", if (a4 == 0) {
                  paste0(" LR")
                }
                else if (a4 == 1) {
                  paste0(" RL")
                }
                else {
                  paste0(" L")
                })))
            return(invisible(c(a1, a2, a3, a4)))
        }
        else {
         return(noquote(paste0(
"A regular approximation is not defined for division since at least one of LR FNs is not positive"
                )))
        }
    }
    else {
        return(noquote(paste0("Division has NOT a closed form of a LR fuzzy number")))
    }
  }
```

L *Introducing the form of L fuzzy number*

Description

Considering the definition of LR fuzzy number in LR, if the left and the right shape functions of a LR fuzzy number are be equal (i.e., $L(.) = R(.)$), then LR fuzzy number is a L fuzzy number which denoted by $(n, \alpha, \beta)L$. Function L introduce a total form for L fuzzy number to computer.

Usage

```
L(m, m_l, m_r)
```

Arguments

m	The core of L fuzzy number
m_l	The left spread of L fuzzy number
m_r	The right spread of L fuzzy number

Value

This function help to users to define any L fuzzy number after introducing the left shape function L. This function consider L fuzzy number L(m, m_l, m_r) as a vector with 4 elements. The first three elements are m, m_l and m_r respectively; and the fourth element is considerd equal to 0.5 for distinguish L fuzzy number from LR and RL fuzzy numbers.

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Examples

```
# First introduce the left shape function of L fuzzy number
Left.fun  = function(x)  { (1-x^2)*(x>=0)}
A = L(20, 12, 10)
LRFN.plot(A, xlim=c(0,60), col=2, lwd=2)

## The function is currently defined as
function (m, m_l, m_r)
{
    c(m, m_l, m_r, 0.5)
  }
```

LR *Introducing the form of LR fuzzy number*

Description

Function LR introduce a total form for LR fuzzy number. Note that, if the membership function of fuzzy number N is

$$N(x) = \begin{cases} L\left(\frac{n-x}{\alpha}\right) & if \quad x \leq n \\ R\left(\frac{x-n}{\beta}\right) & if \quad x > n \end{cases}$$

where L and R are two non-increasing functions from $R^+ \cup \{0\}$ to $[0,1]$ (say left and right shape function) and $L(0) = R(0) = 1$ and also $\alpha, \beta > 0$; then N is named a LR fuzzy number and we denote it by $N = (n, \alpha, \beta)LR$ in which n is core and α and β are left and right spreads of N, respectively.

Usage

```
LR(m, m_l, m_r)
```

Arguments

m	The core of LR fuzzy number
m_l	The left spread of LR fuzzy number
m_r	The right spread of LR fuzzy number

Value

This function help to users to define any LR fuzzy number after introducing the left shape and the right shape functions L and R. This function consider LR fuzzy number LR(m, m_l, m_r) as a vector with 4 elements. The first three elements are m, m_l and m_r respectively; and the fourth element is considerd equal to 0 for distinguish LR fuzzy number from RL and L fuzzy numbers.

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Examples

```
# First introduce left and right shape functions of LR fuzzy number
Left.fun  = function(x)  { (1-x^2)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
A = LR(20, 12, 10)
LRFN.plot(A, xlim=c(0,60), col=1)

## The function is currently defined as
function (m, m_l, m_r)
{
    c(m, m_l, m_r, 0)
  }
```

LRFN.plot *Plotting and drawing LR fuzzy numbers*

Description

By this function one can plot and draw any kind of LR, RL and L fuzzy numbers.

Usage

```
LRFN.plot(M, Left.fun = NULL, Right.fun = NULL, ... )
```

Arguments

M	A LR, RL or L fuzzy number
Left.fun	The left-shape function which usually defined before using LRFN.plot (see examples in bellow)
Right.fun	The right-shape function which usually defined before using LRFN.plot (see examples in bellow)
...	Any argument of curve() function, such as xlim, ylim, lwd, lty, col, add and ... is acceptable for this function

Details

Befor useing "LRFN.plot" function, first define the left shape and the right shape functions of LR fuzzy number. Also, xlim argument must (is better to) be defined for the first fuzzy number.

Author(s)

Abbas Parchami

Examples

```
# Example 1:
# First introduce left-side and right-side functions of LR fuzzy number
Left.fun  = function(x)  { (1-x^2)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
A = LR(20, 12, 10)
LRFN.plot(A, xlim=c(0,60), col=1)
LRFN.plot(A, lty=2, lwd=3, col=2, add=TRUE)


# Example 2:
# for first LR fuzzy number:
Left.fun  = function(x)  { (1-x^2)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
LRFN.plot( LR(17,5,3), xlim=c(5,40), lwd=2, lty=2, col=2)

# for second LR fuzzy number:
```

```
Left.fun  = function(x)  { (1/(1+x^2))*(x>=0)}
Right.fun = function(x)  { (1/(1+(2*abs(x))))*(x>=0)}
LRFN.plot( RL(20,2,3), lwd=2, col=1, add=TRUE)

# for third LR fuzzy number:
Left.fun  = function(x)  { (1-x)*(x>=0)}
LRFN.plot( L(30,15,5), lwd=2, lty=3, col=4, add=TRUE)
legend( "topright", c("LR(17, 5, 3)", "RL(20, 2, 3)", "L(30, 15, 5)"), col = c(2, 1, 4)
     , text.col = 1, lwd = c(2,2,2), lty = c(2, 1, 3) )


## The function is currently defined as
function (M, Left.fun = NULL, Right.fun = NULL, ...)
{
if ( messages(M) != 1 )  { return( messages(M) ) }

m = M[1]
m_l = M[2]
m_r = M[3]

x <- NULL

if ( M[4] == 0 ) { y = function(x) Left.fun((m-x)/m_l) * (x<=m) + Right.fun((x-m)/m_r) * (m<x) }
 else if (M[4]==1) { y = function(x) Right.fun((m-x)/m_l) * (x<=m) + Left.fun((x-m)/m_r) * (m<x) }
 else if (M[4]==0.5) { y = function(x) Left.fun((m-x)/m_l) * (x<=m) + Left.fun((x-m)/m_r) * (m<x)}
 else{return(noquote(paste0("The fourth element of each LR fuzzy number must be 0 or 0.5 or 1!")))}

return( curve(y(x) * (0<=y(x) & y(x)<=1), ...) )
 }
```

`messages` *messages*

Description

The purpose of this function is supporting the functions of this package (by introducing some nested "if-else" conditions) from all possible messages which are defined in functions of this package. The "messages" function is used in most of functions of this package.

Usage

```
messages(M)
```

Arguments

`M`	A L, LR or RL fuzzy number

Value

Some special messages like: "NOT additive", "NOT productive", ... If any message is not necessary for this function, then the value 1 will be return by this function which is used in the text and the body of other functions.

Note

This function has not any applications for users of package and it considered only for shortening the length of programming.

Author(s)

Abbas Parchami

Examples

```
messages("NOT additive")
messages( LR(3,1,1) )

## The function is currently defined as
function (M)
{
    options(warn = -1)
    if (M[1] == "Addition has NOT a closed form of a LR fuzzy number") {
        return(noquote(paste0("Addition has NOT a closed form of a LR fuzzy number")))
    }
    else if (M[1] == "Subtraction has NOT a closed form of a LR fuzzy number") {
       return(noquote(paste0("Subtraction has NOT a closed form of a LR fuzzy number")))
    }
    else if (M[1] == "Production has NOT a closed form of a LR fuzzy number") {
        return(noquote(paste0("Production has NOT a closed form of a LR fuzzy number")))
    }
    else if (M[1] == "Division has NOT a closed form of a LR fuzzy number") {
        return(noquote(paste0("Division has NOT a closed form of a LR fuzzy number")))
    }
   else if (M[1] == " The fourth element of each LR fuzzy number must be 0 or 0.5 or 1! ") {
    return(noquote(paste0(" The fourth element of each LR fuzzy number must be 0 or 0.5 or 1! ")))
    }
    else if (M[1] == " The scalar multiplication is not defined for zero ") {
        return(noquote(paste0(" The scalar multiplication is not defined for zero ")))
    }
    else if (M[1] ==
"A regular approxi. is not defined for multiplication since at least one of FNs is non + and non -"
         ) {
        return(noquote(paste0(
"A regular approxi. is not defined for multiplication since at least one of FNs is non + and non -"
              )))
    }
    else if (M[1] ==
"A regular approximation is not defined for division since at least one of LR FNs is not positive"
         ) {
        return(noquote(paste0(
```

```
"A regular approximation is not defined for division since at least one of LR FNs is not positive"
              )))
    }
    else {
        return(1)
    }
  }
```

`multiplication` *Product of two LR fuzzy numbers*

Description

This function calculates the multiplication (product) of two LR fuzzy numbers. Although on the basis of Zadeh extension principle, the class of LR fuzzy numbers is not closed under the operations multiplication and division, but we consider the following approximation for the product of two LR fuzzy numbers $M = (m, \alpha, \beta)_{LR}$ and $N = (n, \gamma, \delta)_{LR}$ in this package to work easy in the class of LR fuzzy numbers:

$$M \otimes N \simeq \begin{cases} (mn, m\gamma + n\alpha, m\delta + n\beta)_{LR} & if \quad M \succ 0 \; and \; N \succ 0 \\ (mn, m\gamma - n\beta, m\delta - n\alpha)_{RL} & if \quad M \succ 0 \; and \; N \prec 0 \\ (mn, -n\beta - m\delta, -n\alpha - m\gamma)_{RL} & if \quad M \prec 0 \; and \; N \prec 0 \end{cases}$$

Usage

```
m(M, N)
```

Arguments

`M`	The first LR (or RL or L) fuzzy number
`N`	The second LR (or RL or L) fuzzy number

Value

A LR (or RL or L) fuzzy number

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

Examples

```
# Example 1:
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun = function(x)  { (1/(1+(2*abs(x))))*(x>=0)}
A = LR(1, 0.6, 0.2)
B = LR(-3, 0.5, 1)
m(A, B)
m(B, A)
xlim = c(-5,4)
LRFN.plot( A, xlim=xlim, lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
legend( "topright", c("A = LR(1, 0.6, 0.2)", "B = LR(-3, 0.5, 1)"), col = c(2, 3)
     , text.col = 1, lwd = c(2,2), lty = c(2, 2) )


# Example 2:
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
A = LR(1.5, 1, 2)
B = LR(3, 2, 1)
LRFN.plot( A, xlim=c(-3,20), ylim=c(0,1), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( m(A,B), lwd=2, lty=3, col=1, add=TRUE)
legend( "topright", c("A = LR(1.5, 1, 2)", "B = LR(3, 2, 1)", "A * B = LR(4.5, 6, 7.5)")
     , col = c(2, 3, 1), text.col = 1, lwd = c(2,2,2), lty = c(2, 2, 3) )


# Example 3:
M = LR(1.2, 0.6, 0.2)
N = LR(3, 0.5, 1)

m(M,N)
m( LR(1.2, 0.6, 0.2) , LR(3, 0.5, 1) )

m(N,m(M,M))
m(m(N,M),M)

LRFN.plot( M, xlim=c(-2,10), ylim=c(0,1.4), lwd=2, lty=2, col=2)
LRFN.plot( N, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( m(M,N), lwd=2, lty=3, col=4, add=TRUE)
```

```
LRFN.plot( m(M,M), lwd=2, lty=4, col=5, add=TRUE)
LRFN.plot( m(m(N,M),M), lwd=2, lty=5, col=1, add=TRUE)
legend( "topright", c("M = LR(1.2, 0.6, 0.2)", "N = LR(3, 0.5, 1)", "M * N = LR(3.6, 2.4, 1.8)"
    , "M * M = LR(3.6, 2.4, 1.8)", "(N * M) * M = LR(4.32, 5.04, 2.88)"), col = c(2, 3, 4, 5, 1),
     text.col = 1, lwd = c(2,2,2,2,2), lty = c(2, 2, 3, 4, 5) )


## The function is currently defined as
function (M, N)
{
    options(warn = -1)
    if (messages(M) != 1) {
        return(messages(M))
    }
    if (messages(N) != 1) {
        return(messages(N))
    }
    if (M[4] != N[4]) {
        return(noquote(paste0("Production has NOT a closed form of a LR fuzzy number")))
    }
    else if ((sign(M) == "Positive") & (sign(N) == "Positive")) {
        a1 = M[1] * N[1]
        a2 = (M[1] * N[2]) + (N[1] * M[2])
        a3 = (M[1] * N[3]) + (N[1] * M[3])
        a4 = (M[4] + N[4])/2
        print(noquote(paste0("the result of multiplication is approximately  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
            })))
        return(invisible(c(a1, a2, a3, a4)))
    }
    else if ((sign(M) == "Negative") & (sign(N) == "Negative")) {
        a1 = M[1] * N[1]
        a2 = -(M[1] * N[2]) - (N[1] * M[2])
        a3 = -(M[1] * N[3]) - (N[1] * M[3])
        a4 = abs(M[4] - 1)
        print(noquote(paste0("the result of multiplication is approximately  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
```

```
            }))))
        return(invisible(c(a1, a2, a3, a4)))
    }
    else if ((sign(M) == "Positive") & (sign(N) == "Negative")) {
        a1 = M[1] * N[1]
        a2 = (M[1] * N[2]) - (N[1] * M[3])
        a3 = (M[1] * N[3]) - (N[1] * M[2])
        a4 = abs(M[4] - 1)
        print(noquote(paste0("the result of multiplication is approximately  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
            })))
        return(invisible(c(a1, a2, a3, a4)))
    }
    else if ((sign(M) == "Negative") & (sign(N) == "Positive")) {
        a1 = M[1] * N[1]
        a2 = (N[1] * M[2]) - (M[1] * N[3])
        a3 = (N[1] * M[3]) - (M[1] * N[2])
        a4 = abs(N[4] - 1)
        print(noquote(paste0("the result of multiplication is approximately  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
            })))
        return(invisible(c(a1, a2, a3, a4)))
    }
    else {
        return(noquote(paste0(
"A regular approxi. is not defined for multiplication since at least one of FNs is non + and non -"
            )))
    }
  }
```

RL *Introducing the form of RL fuzzy number*

Description

Considering the definition of LR fuzzy number in LR, it is obvious that $(n, \alpha, \beta)RL$ will be a RL fuzzy number. Function RL introduce a total form for RL fuzzy number to computer.

Usage

```
RL(m, m_l, m_r)
```

Arguments

m	The core of RL fuzzy number
m_l	The left spread of RL fuzzy number
m_r	The right spread of RL fuzzy number

Value

This function help to users to define any RL fuzzy number after introducing the left shape and the right shape functions L and R. This function consider RL fuzzy number RL(m, m_l, m_r) as a vector with 4 elements. The first three elements are m, m_l and m_r respectively; and the fourth element is considerd equal to 1 for distinguish RL fuzzy number from LR and L fuzzy numbers.

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Examples

```
# First introduce left and right shape functions of RL fuzzy number
Left.fun  = function(x)  { (1-x^2)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
A = RL(40, 12, 10)
LRFN.plot(A, xlim=c(0,60), col=1)

## The function is currently defined as
function (m, m_l, m_r)
{
    c(m, m_l, m_r, 1)
  }
```

`scalar multiplication` *Scalar multiplication on LR fuzzy numbers*

Description

This function calculates the scalar multiplication of any non-zero real number to any LR fuzzy number on the basis of Zadeh extension principle by the following formula which is for any LR fuzzy number $M = (m, \alpha, \beta)_{LR}$ and real number $\lambda \in R - \{0\}$:

$$\lambda \odot M = M \odot \lambda = \begin{cases} (\lambda m, \lambda\alpha, \lambda\beta)_{LR} & if \quad \lambda > 0 \\ (\lambda m, -\lambda\beta, -\lambda\alpha)_{RL} & if \quad \lambda < 0 \end{cases}$$

Usage

```
s.m(k, N)
```

Arguments

`k`	A non-zero real number
`N`	A LR (or RL, or L) fuzzy number

Details

This function has commutative property, i.e $k \odot M = M \odot k$.

Value

A LR (or RL or L) fuzzy number

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

Examples

```
# Example 1:
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun  = function(x)  { (1-x)*(x>=0)}
k = 2
M = LR(1, 0.6, 0.2)
N = L(3, 0.6, 1)
P = RL(5, 0.1, 0.3)
s.m(k, N)

# commutative property for scalar multiplication on LR fuzzy numbers (Jabejaei)
s.m(k, M)
s.m(M, k)

s.m(k, P)
s.m(-2, LR(4,2,1))

s.m(2, s.m(-2, LR(4,2,1)))

# Example 2:
Left.fun  = function(x)  { (1/(1+x^2))*(x>=0)}
Right.fun = function(x)  { (1/(1+(2*abs(x))))*(x>=0)}
A = RL(3,2,1)
LRFN.plot( A, xlim=c(-4,15), lwd=2, lty=2, col=2)
LRFN.plot( s.m(0.5, A), lwd=2, lty=3, col=1, add=TRUE)
LRFN.plot( s.m(2, A), lwd=2, lty=4, col=1, add=TRUE)
legend( "topright", c("A = RL(3, 2, 1)", "0.5 A", "2 A"), col = c(2, 1, 1), text.col = 1
     , lwd = c(2,2,2), lty = c(2, 3, 4))


## The function is currently defined as
function (k, N)
{
    if (messages(N) != 1) {
        return(messages(N))
    }
    if (messages(k) != 1) {
        return(messages(k))
    }
    if (length(k) == 4 & length(N) == 1) {
        zarf = N
        N[1] = k[1]
        N[2] = k[2]
        N[3] = k[3]
        N[4] = k[4]
        k = zarf
    }
```

```
    if (k == 0) {
        return(noquote(paste0(" The scalar multiplication is not defined for zero ")))
    }
    else {
        a1 = k * N[1]
        a2 = k * (N[2] * (k > 0) - N[3] * (k < 0))
        a3 = k * (N[3] * (k > 0) - N[2] * (k < 0))
        a4 = N[4]
        print(noquote(paste0("the result of scalar multiplication is  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
            })))
        return(invisible(c(a1, a2, a3, a4)))
    }
}
```

`sign` *Sign of LR fuzzy number*

Description

To distinguish and determining the sign of a LR fuzzy number one can use from this function. In other words, the function `sign` is able to categorize the class of all LR fuzzy numbers into three kinds positive, negative and non of them (non-positive and non negative).

Usage

```
sign(M)
```

Arguments

`M`	A LR, RL or L fuzzy number

Value

The "sign" function only return three charactical values: "Positive", "Negative" or "non-positive and non negative".

Author(s)

Abbas Parchami

Examples

```
Left.fun  = function(x)  { (1-x)*(x>=0)}

M = L(2,4,3)
support(M)
sign(M)

sign( L(5,4,3) )

( sign( L(5,4,3) ) == "Positive" )

## The function is currently defined as
function (M)
{
    supp = support(M)
    if (supp[1] > 0) {
        return(noquote(paste0("Positive")))
    }
    else {
        if (supp[2] < 0) {
            return(noquote(paste0("Negative")))
        }
        else {
            return(noquote(paste0("non-positive and non negative")))
        }
    }
  }
```

`subtraction` *Subtraction of two LR fuzzy numbers*

Description

This function calculates subtraction (difference) of two fuzzy numbers $M = (m, \alpha, \beta)_{LR}$ and $N = (n, \gamma, \delta)_{RL}$ on the basis of Zadeh extension principle by the following formula:

$$M \ominus N = (m - n, \alpha + \delta, \beta + \gamma)_{LR}$$

Usage

```
s(M, N)
```

Arguments

`M`	The first LR (or RL or L) fuzzy number
`N`	The second LR (or RL or L) fuzzy number

Value

A LR (or RL or L) fuzzy number

Author(s)

Abbas Parchami

References

Dubois, D., Prade, H., Fuzzy Sets and Systems: Theory and Applications. Academic Press (1980).

Dubois, D., Prade, H., Operations on fuzzy numbers. International Journal of Systems Science 9 (1978), 613-626.

Dubois, D., Prade, H., Fuzzy numbers: An overview. In In: Analysis of Fuzzy Information. Mathematical Logic, Vol. I. CRC Press (1987), 3-39.

Dubois, D., Prade, H., The mean value of a fuzzy number. Fuzzy Sets and Systems 24 (1987), 279-300.

Kaufmann, A., Gupta, M.M., Introduction to Fuzzy Arithmetic. van Nostrand Reinhold Company, New York (1985).

Taheri, S.M, Mashinchi, M., Introduction to Fuzzy Probability and Statistics. Shahid Bahonar University of Kerman Publications, In Persian (2009).

Viertl, R., Statistical Methods for Fuzzy Data. John Wiley & Sons, Chichester (2011).

Zadeh, L.A., The concept of a linguistic variable and its application to approximate reasoning-I. Information Sciences 8 (1975), 199-249.

Examples

```
# Example 1:
Left.fun  = function(x)  { (1/(1+x^2))*(x>=0)}
Right.fun = function(x)  { (1/(1+(2*abs(x))))*(x>=0)}
M = LR(1, 0.6, 0.2)
N = RL(3, 0.5, 1)

s(N, M)
s(M, N)
s(M, M)
s(s(N, M), M)

# Example 2:
Left.fun  = function(x)  { (1-x)*(x>=0)}
A = L(5,3,2)
B = L(3,2,1)
LRFN.plot( A, xlim=c(-3,12), lwd=2, lty=2, col=2)
LRFN.plot( B, lwd=2, lty=2, col=3, add=TRUE)
LRFN.plot( s(A, B), lwd=2, lty=3, col=1, add=TRUE)
legend( "topright", c("A = L(5, 3, 2)", "B = L(3, 2, 1)", "A - B = L(2, 4, 4)"), col = c(2, 3, 1)
     , text.col = 1, lwd = c(2,2,2), lty = c(2, 2, 3) )

## The function is currently defined as
function (M, N)
{
    options(warn = -1)
    if (messages(M) != 1) {
```

```
        return(messages(M))
    }
    if (messages(N) != 1) {
        return(messages(N))
    }
    if ((M[4] == 1 & N[4] == 0) | (M[4] == 0 & N[4] == 1) | (M[4] ==
        0.5 & N[4] == 0.5)) {
        a1 = M[1] - N[1]
        a2 = M[2] + N[3]
        a3 = M[3] + N[2]
        a4 = M[4]
        print(noquote(paste0("the result of subtraction is  (core = ",
            a1, ", left spread = ", a2, ", right spread = ",
            a3, ")", if (a4 == 0) {
                paste0(" LR")
            }
            else if (a4 == 1) {
                paste0(" RL")
            }
            else {
                paste0(" L")
            })))
        return(invisible(c(a1, a2, a3, a4)))
    }
    else {
       return(noquote(paste0( "Subtraction has NOT a closed form of a LR fuzzy number" )))
    }
}
```

---

support *Support of LR fuzzy number*

Description

To determining the support of a LR fuzzy number one can use from this function. In other words, the support function is able to compute the smallest and biggest values x for which $\mu(x) > 0$.

Usage

```
support(M, Left.fun = NULL, Right.fun = NULL)
```

Arguments

M	A LR, RL or L fuzzy number
Left.fun	The left-shape function which usually defined before using LRFN.plot (see examples in bellow)
Right.fun	The right-shape function which usually defined before using LRFN.plot (see examples in bellow)

Value

The "support" function return a interval-valued vector in which the membership function value of LR fuzzy number is bigger than zero.

Author(s)

Abbas Parchami

Examples

```
Left.fun  = function(x)  { (1-x)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
T = LR(1, 0.6, 0.2)
support(T)
LRFN.plot( T, xlim=c(-5,20), lwd=2, lty=3, col=4)

N = RL(3, 0.5, 2)
support(N)

Left.fun  = function(x)  { (1-x)*(x>=0)}
M = L(2,4,3)
support(M)

Left.fun  = function(x)  { (1-x^2)*(x>=0)}
Right.fun = function(x)  { (exp(-x))*(x>=0)}
support( LR(17,5,3))


## The function is currently defined as
function (M, Left.fun = NULL, Right.fun = NULL)
{
    range1 = M[1] - M[2] - M[3] - 100
    range2 = M[1] + M[2] + M[3] + 100
    x = seq(range1, range2, len = 2e+05)
    if (M[4] == 0) {
        y = Left.fun((M[1] - x)/M[2]) * (x <= M[1]) + Right.fun((x -
            M[1])/M[3]) * (M[1] < x)
    }
    else if (M[4] == 1) {
        y = Right.fun((M[1] - x)/M[2]) * (x <= M[1]) + Left.fun((x -
            M[1])/M[3]) * (M[1] < x)
    }
    else if (M[4] == 0.5) {
        y = Left.fun((M[1] - x)/M[2]) * (x <= M[1]) + Left.fun((x -
            M[1])/M[3]) * (M[1] < x)
    }
    supp = c()
    supp[1] = min(x[0 < y & y < 1])
    supp[2] = max(x[0 < y & y < 1])
    if (supp[1] == min(x)) {
        supp[1] = -Inf
    }
```

```
  if (supp[2] == max(x)) {
      supp[2] = +Inf
  }
  return(supp)

}
```

Index