Package 'FuzzyToolkitUoN'

February 19, 2015

Type Package

Version 1.0

Date 2013-04-26

Title Type 1 Fuzzy Logic Toolkit

Author Craig Knott, Luke Hovell, Nathan Karimian with supervision from Dr. Jon Garibaldi

Maintainer Jon Garibaldi <jon.garibaldi@ima.ac.uk>

Depends R (>= 2.15.1), splines

Description A custom framework for working with Type 1 Fuzzy Logic, produced by the University of Nottingham IMA Group.

License GPL (>= 2)

URL http://ima.ac.uk

NeedsCompilation No

Repository CRAN

Date/Publication 2013-04-26 18:26:38

R topics documented:

`FuzzyToolkitUoN-package` *Type 1 Fuzzy Logic Toolkit*

Description

A custom framework for working with Type 1 Fuzzy Logic, produced by the University of Nottingham IMA Group.

Details

Package:	FuzzyToolkitUoN
Type:	Package
Version:	1.0
Date:	2013-04-26
License:	GPL (>= 2)
NeedsCompilation:	No
Repository:	CRAN

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Maintainer: Jon Garibaldi <jon.garibaldi@ima.ac.uk>

`addMF` *Insert a membership function.*

Description

Adds a membership function to a variable of a FIS object.

Usage

```
addMF(FIS, varType, varIndex, mf)
```

Arguments

FIS	A FIS structure is to be provided.
varType	Should be either 'input' or 'output', which relates to the type of variable (stored on the existing FIS structure) that the membership function will be added to.
varIndex	Should be an integer value representing the index value of the input or output variable that the membership function will be added to (base 1).
mf	The membership function to be stored in the specified location on the given FIS structure.

Value

A FIS structure with the new membership function added.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MF <<- gaussMF("myMF", 0:10, c(1.5,5,1))
FIS <<- newFIS("MyFIS")
FIS <<- addVar(FIS, "input", "variableName", 0:10)
FIS <<- addMF(FIS, "input", 1, MF)
```

---

addRule *Inserts a rule*

Description

Adds a rule to a FIS object.

Arguments

FIS	A FIS structure is to be provided.
inputRule	A vector of length m + n + 2, where m is the number of input variables of a FIS. Each column in 'm' has a number which refers to the membership function of that input variable. Columns under 'n' refer to an output variable of a FIS, where the value refers to the membership function of that output variable. Finally, the '2' remaining columns refer to the weight to be applied to the rule (m + n + 1) and the fuzzy operator for the rule's antecedent (1 = AND, 2 = OR).

Details

For example, if one has a FIS with 2 input variables, and 1 output variable, each of which have 3 membership functions (the amount of membership functions need not be the same). The following rule: 1 3 2 1 2 will mean m = 2 (for 2 input variables), n = 1 (for 1 output variable), and the last 2 columns represent weight and fuzzy operator for the rule's antecedent respectively.

The first column refers to the first input variable's membership function at index 1.

The second column refers to the second input variable's membership function at index 3.

The third column refers to the first output variable's membership function at index 2.

The fourth column refers to the weight to be applied to the rule.

The fifth column refers to the fuzzy operator for the rule's antecedent (in this case it represents 'OR').

Value

A FIS structure with the new rule added.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
FIS <<- addRule(FIS, c(2,2,1,1,1))
```

---

addVar *Insert a variable*

Description

Adds an input or output variable to a FIS object.

Usage

```
addVar(FIS, varType, varName, varBounds)
```

Arguments

FIS	A FIS must be provided.
varType	Should be either 'input' or 'output' which represents the type of variable to be created and added.
varName	A string representing the name of the variable.
varBounds	Also known as the 'range', this should be a vector giving a range for the variable, such as 1:10.

Value

A FIS with the new variable added.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
FIS <<- addVar(FIS, "input", "MyVariableName", 0:10)
```

`defuzz` *Defuzzify a set of values.*

Description

Defuzzifies a given set of values using a specified range and defuzzification type producing a crisp value.

Usage

```
defuzz(x, vals, type)
```

Arguments

`x`	The range to be applied in the function (numeric vector).
`vals`	The values to be applied in the function (numeric vector).
`type`	The defuzzification method type, which should be either 'centroid', 'bisector', 'mom', 'som' or 'lom'.

Value

Returns a defuzzified crisp value (double).

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
defuzz(1:10,c(1.5, 5, 1),"centroid")
```

evalFIS *Evaluate a Fuzzy Inference System.*

Description

Returns an evaluated crisp value for a given FIS structure.

Usage

```
evalFIS(inputStack, fis, numPoints = 101)
```

Arguments

inputStack	A matrix representing the input stack, number of inputs (columns) by number of outputs (rows).
fis	A FIS must be provided.
numPoints	An optional argument that represents the number of sample points on which to evaluate the membership functions.

Value

Returns a matrix of evaluated values.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MyMatrix <<- matrix((1:2),1,2)
FIS <<- tippertest()
evalFIS(MyMatrix, FIS)
evalFIS(MyMatrix, FIS, 50)
```

evalMF *Evaluation of a membership function.*

Description

Evaluates a membership function dependent on the function type to return plottable values.

Usage

```
evalMF(x, mfParams, mfType)
```

Arguments

x	The variable range, numeric vector.
mfParams	The input parameters that will be used with a membership function. They should be in a numeric vector which is the same length as required by the membership function's mfParams.
mfType	The type of the membership function, a string that can be either 'gaussMF', 'gaussbMF', 'triMF', or 'trapMF'.

Value

Returns a numeric vector containing the values of an evaluated membership function.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian.

Examples

```
Values <<- evalMF(1:10, c(1.5, 5, 1), "gaussMF")
```

---

gaussbMF *Create a gaussian bell membership function.*

Description

Creates two Gaussian Bell curves with different parameters and merges them.

Usage

```
gaussbMF(mfName, x, mfParams)
```

Arguments

mfName	A string representing the name of the membership function.
x	The range of the membership function as a vector, such as 1:10.
mfParams	The input parameters of the membership function. This should be a vector of 5 numbers representing the left sigma, left mean, right sigma, right mean, and height.

Details

To access the values:

<n>$mfParams for the stored parameters.

<n>$mfX for the stored range.

<n>$mfName for the stored name.

<n>$mfVals for the stored evaluated values.

Where <n> is the assigned name of the membership function in the environment.

Value

mfName	The name of the membership function (String)
mfX	A numeric vector representing the range of the variable.
mfParams	A numeric vector representing the given input parameters upon creation. These should be the left sigma, left mean, right sigma, right mean and height.
mfVals	The evaluated values for the membership function.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MyGaussianBell <<- gaussbMF("MyMF", 0:10, c(2,3,3,2,1))
```

---

gaussMF *Creates a gaussian membership function.*

Description

Creates a single Gaussian curve.

Usage

```
gaussMF(mfName, x, mfParams)
```

Arguments

mfName	String representing the name for the membership function.
x	The range of the membership function as a vector, such as 1:10.
mfParams	The input parameters, this should be a vector of 3 numbers representing the sigma, mean and height.

Details

To access the values:

<n>$mfParams for the stored parameters.

<n>$mfX for the stored range.

<n>$mfName for the stored name.

<n>$mfVals for the stored evaluated values.

Where <n> is the assigned name of the membership function in the environment.

Value

mfName	The name of the membership function (String)
mfX	A numeric vector representing the range of the variable.
mfParams	A numeric vector representing the given input parameters upon creation.
mfVals	The evaluated values for the membership function.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MyGaussian <<- gaussMF("MyMF", 0:10, c(1.5, 5, 1))
```

---

gensurf *Produce a graphical evaluated fuzzy inference system.*

Description

Produces a three dimensional graphical view of a specific FIS object.

Usage

```
gensurf(fis, ix1 = 1, ix2 = 2, ox1 = 1)
```

Arguments

fis	A FIS must be provided.
ix1	Optional input (1)
ix2	Optional input (2)
ox1	Optional output

Value

A three dimensional graphical model generated from the FIS and other optional parameters.

Note

As this is a 3D graphical representation it only works for FIS structures with 3 variables. It will only work for 2 inputs, and 1 output.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
gensurf(FIS)
```

`meshgrid` *Union of two vectors.*

Description

Generates the union of two input vectors.

Usage

```
meshgrid(a, b)
```

Arguments

a	Input vector 1
b	Input vector 2

Value

Union of the two input vectors

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
meshgrid((1:10), (11:20))
```

`mfValidate` *Validate the input of a membership function. mfValidate*

Description

Validates all user input when a membership function is created, and will stop function execution if invalid values are given.

Usage

```
mfValidate(mfName, mfParams)
```

Arguments

mfName	Passed string used to check the given name.
mfParams	Passed mfParams (numeric vector) used to check the right amount of mfParams have been given.

Value

No value returned, but can stop operation with stop().

Note

Invalid values can include illegal symbols, strings of length 0, non-character based data types as given argument etc.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

---

nameValidate *Validate the name of an object.*

Description

Validates user input for names of objects.

Usage

```
nameValidate(name)
```

Arguments

name	Passed in string which is tested for validity (can not contain illegal characters, be an empty string etc.).

Note

A valid name is any that has more than 0 characters, and contains no special characters.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

newFIS *Create a FIS newFIS*

Description

Creates a FIS object.

Usage

```
newFIS(FISName, FISType = "mamdani", version = "1.0", andMethod = "min", orMethod = "max", impMethod =
```

Arguments

FISName	String representing the FIS name.
FISType	Type of the FIS, default is 'mamdani'.
version	FIS version, default is '1.0' as a string.
andMethod	The AND method for the FIS, default is 'min'.
orMethod	The OR method for the FIS, default is 'max'.
impMethod	The implication method for the FIS, default is 'min'.
aggMethod	The aggregation method for the FIS, default is 'max'.
defuzzMethod	The defuzzification method for the FIS, default is 'centroid'.

Value

A new FIS structure.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MyFIS <<- newFIS("MyFISName")
```

plotMF *Plots a 2D graph of all membership functions in a variable.*

Description

Plots a 2D graph of all membership functions from the specified variable which must be part of a FIS object.

Usage

```
plotMF(FIS, varType, varIndex)
```

Arguments

FIS	Requires an existing FIS as an argument.
varType	Can be either 'input' or 'output', representing the type of variable.
varIndex	A numerical integer, representing the index of the input or output variable whose membership functions shall be plotted (base 1).

Value

A two dimensional graph displaying all the membership functions of a given variable.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
plotMF(FIS, "input", 1)
```

readFIS *Read a FIS object from a .fis file. readFIS*

Description

Reads a FIS object from a file with the .fis extension, and converts it into a data structure to be used within the environment.

Usage

```
readFIS(fileName)
```

Arguments

fileName	Should be an absolute path given as a string to the file to be read, with escaped backslashes.

Value

A FIS structure with its values generated from that of the files.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

---

removeMF *Remove a membership function.*

Description

Removes a specified membership function from a given variable (which must be part of a FIS object).

Usage

```
removeMF(FIS, varType, varIndex, mfIndex)
```

Arguments

FIS	Requires a FIS.
varType	Should be either 'input' or 'output', representing the type of variable.
varIndex	The index of the input or output variable (Base 1).
mfIndex	The index of the membership function which is to be removed.

Value

A FIS with the membership function removed.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
FIS <<- removeMF(FIS, "input", 2, 1)
```

removeVar *Remove a variable.*

Description

Removes a specified variable and all its attached membership functions from a given FIS object.

Usage

```
removeVar(FIS, varType, varIndex)
```

Arguments

FIS	A FIS is required.
varType	Should be either 'input' or 'output', representing the type of variable to be removed.
varIndex	The index of the variable that is to be removed (Base 1).

Value

A FIS with the specified variable removed.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
FIS <<- removeVar(FIS, "input", 1)
```

showFIS *Show a FIS object.*

Description

Shows a FIS and all its data in an ordered format on the console.

Usage

```
showFIS(FIS)
```

Arguments

FIS	Requires a FIS structure to be displayed.

Value

Nothing is returned, but organised text regarding the FIS is output to console.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
showFIS(FIS)
```

`tippertest` *Produces an example FIS object.*

Description

A function used primarily for example purposes, it creates a FIS with various input, output variables and their membership functions.

Usage

```
tippertest()
```

Value

A FIS is returned.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
FIS <<- tippertest()
```

`trapMF` *Create a trapezoidal membership function.*

Description

Creates a trapezoidal membership function object.

Usage

```
trapMF(mfName, x, mfParams)
```

Arguments

`mfName`	String representing the name of the membership function
`x`	The range of the membership function, such as 1:10
`mfParams`	The parameters, which should be a numeric vector of left foot, left shoulder, right shoulder, right foot and height.

Details

To access the values:

<n>$mfParams for the stored parameters.

<n>$mfX for the stored range.

<n>$mfName for the stored name.

<n>$mfVals for the stored evaluated values.

Where <n> is the assigned name of the membership function in the environment.

Value

`mfName`	The name of the membership function (String)
`mfX`	A numeric vector representing the range of the variable.
`mfParams`	A numeric vector representing the given input parameters upon creation. These should be the left foot, left shoulder, right shoulder and height.
`mfVals`	The evaluated values for the membership function.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
myMembershipFunction <- trapMF("NameOfMembershipFunction", 1:10, c(1,2,4,5,1))
```

`triMF` *Create a triangular membership function.*

Description

Creates a triangular membership function object.

Usage

```
triMF(mfName, x, mfParams)
```

Arguments

`mfName`	String representing the name.
`x`	The range, should be a numerical vector such as 1:10
`mfParams`	The input parameters, which should be a 4 number vector representing the left, mean, right and height.

Details

To access the values:

<n>$mfParams for the stored parameters.

<n>$mfX for the stored range.

<n>$mfName for the stored name.

<n>$mfVals for the stored evaluated values.

Where <n> is the assigned name of the membership function in the environment.

Value

`mfName`	The name of the membership function (String)
`mfX`	A numeric vector representing the range of the variable.
`mfParams`	A numeric vector representing the given input parameters upon creation. These should be the left, mean, right and height.
`mfVals`	The evaluated values for the membership function.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Examples

```
MyMembershipFunction <<- triMF("MyTri", 1:10, c(3,6,8,1))
```

`writeFIS` *Write a FIS object to file.*

Description

Writes a FIS object to a specified file given with an absolute path.

Usage

```
writeFIS(FIS, fileName)
```

Arguments

`FIS`	Requires a FIS which will be written to file.
`fileName`	An absolute path given in a string, with escaped backslashes.

Author(s)

Craig Knott, Luke Hovell, Nathan Karimian

Index