

# Package ‘FisPro’

April 19, 2022

**Type** Package

**Title** Fuzzy Inference System Design and Optimization

**Version** 1.1.1

**Author** Serge Guillaume [aut],  
Brigitte Charnomordic [aut],  
Jean-Luc Lablée [aut, cre],  
Hazaël Jones [ctb],  
Lydie Desperben [ctb],  
INRAE [cph] (National Research Institute for Agriculture, Food and  
Environment, France)

**Maintainer** Jean-Luc Lablée <jean-luc.lablee@inrae.fr>

**URL** <https://www.fispro.org>

**Description** Fuzzy inference systems are based on fuzzy rules, which have a good capability for managing progressive phenomena.

This package is a basic implementation of the main functions to use a Fuzzy Inference System (FIS) provided by the open source software 'FisPro' <<https://www.fispro.org>>.

'FisPro' allows to create fuzzy inference systems and to use them for reasoning purposes, especially for simulating a physical or biological system.

**License** CeCILL

**Encoding** UTF-8

**Depends** R (>= 3.6.0)

**Imports** methods, utils, Rdpack, Rcpp (>= 1.0.0)

**RdMacros** Rdpack

**NeedsCompilation** yes

**LinkingTo** Rcpp, BH

**Suggests** testthat, rlang, knitr, rmarkdown

**RoxygenNote** 7.1.1

**VignetteBuilder** knitr

**Repository** CRAN

**Date/Publication** 2022-04-19 12:22:31 UTC

## R topics documented:

Fis	2
FisIn	5
FisOut	6
FisOutCrisp	7
FisOutFuzzy	8
FisPro	9
Mf	10
MfTrapezoidal	11
MfTrapezoidalInf	11
MfTrapezoidalSup	12
MfTriangular	13
NewFis	13
NewFisIn	14
NewFisOutCrisp	14
NewFisOutFuzzy	15
NewMfTrapezoidal	15
NewMfTrapezoidalInf	16
NewMfTrapezoidalSup	16
NewMfTriangular	17
NewRule	17
Rule	18
<b>Index</b>	<b>19</b>

---

Fis	<i>Class "Fis"</i>
-----	--------------------

---

### Description

Class to manage a Fis "Fuzzy Inference System"

### Fields

name **character** vector, The name of the Fis

conjunction **character** vector, The conjunction operator of rules in the Fis  
 Allowed values are: "min" (the default), "prod" or "luka"

### Constructors

Fis() The default constructor to build an empty Fis

The Fis is initialized with "min" conjunction and empty name

The design must be completed using the available functions to add inputs, outputs and rules before it can be used for inference

**return:** Fis object

`Fis(fis_file)` The constructor to build a Fis from a configuration file  
 The configuration file can be designed using the **FisPro** open source software  
**argument:** `fis_file` **character** vector, The filename of the Fis configuration file  
**return:** **Fis** object

## Methods

`input_size()`  
**return:** **integer** value, The number of inputs in the Fis

`add_input(input)`  
**argument:** `input` **FisIn** object, The input to add in the Fis

`get_input(input_index)`  
**argument:** `input_index` **integer** value, The index (1-based index) of the input in the Fis  
**return:** **FisIn** object

`get_inputs()` Get all inputs in the Fis  
**return:** **list** of **FisIn** objects

`output_size()`  
**return:** **integer** value, The number of outputs in the Fis

`add_output(output)`  
**argument:** `output` **FisOut** object, The output to add in the Fis

`get_output(output_index)`  
**argument:** `output_index` **integer** value, The index (1-based index) of the output in the Fis  
**return:** **FisOut** object

`get_outputs()` Get all outputs in the Fis  
**return:** **list** of **FisOut** objects

`rule_size()`  
**return:** **integer** value, The number of rules in the Fis

`add_rule(rule)`  
**argument:** `rule` **Rule** object, The rule to add in the Fis

`get_rule(rule_index)`  
**argument:** `rule_index` **integer** value, The index (1-based index) of the rule in the Fis  
**return:** **Rule** object

`get_rules()` Get all rules in the Fis  
**return:** **list** of **Rule** objects

`infer(data)` Infers all outputs  
**argument:** `data` **numeric** vector, **matrix** or **data.frame**, The input data or dataset to infer (the vector length or the number of columns must be equal to the number of inputs)  
**return:** **numeric** vector or **matrix** (in case of 2D input data)

`infer_output(data, output_index)` Infers a single output  
**argument:** `data` **numeric** vector, **matrix** or **data.frame**, The input data or dataset to infer (the vector length or the number of columns must be equal to the number of inputs)  
**argument:** `output_index` **integer** value, The index (1-based index) of the output to infer  
**return:** **numeric** value or vector (in case of 2D input data)

**See Also**[NewFis](#)[Fuzzy Logic Elementary Glossary](#)**Examples**

```

# build a Fis from a configuration file
fis_file <- system.file("extdata", "test.fis", package = "FisPro")
fis <- NewFis(fis_file)

# infers all outputs
inferred <- fis$infer(c(0.25, 0.75))

# infers first output
inferred_output1 <- fis$infer_output(c(0.25, 0.75), 1)

# infers second output
inferred_output2 <- fis$infer_output(c(0.25, 0.75), 2)

# infers test_data dataset
test_file <- system.file("extdata", "test_data.csv", package = "FisPro")
dataset <- read.csv(test_file)
inferred_dataset <- fis$infer(dataset)

#####

# or build a Fis from scratch
fis <- NewFis()
fis$name <- "foo"

# build the first input
fisin1 <- NewFisIn(0, 1)
fisin1$name <- "input1"
fisin1$add_mf(NewMfTrapezoidalInf(0, 1))
fisin1$add_mf(NewMfTrapezoidalSup(0, 1))
fis$add_input(fisin1)

# build the second input
fisin2 <- NewFisIn(0, 1)
fisin2$name <- "input2"
fisin2$add_mf(NewMfTrapezoidalInf(0, 0.5))
fisin2$add_mf(NewMfTriangular(0, 0.5, 1))
fisin2$add_mf(NewMfTrapezoidalSup(0.5, 1))
fis$add_input(fisin2)

# build an output
fisout <- NewFisOutCrisp(0, 1)
fisout$name <- "output"
fis$add_output(fisout)

# add rules to the Fis
fis$add_rule(NewRule(c(1, 2), 0))

```

```
fis$add_rule(NewRule(c(2, 0), 1))
```

---

FisIn	<i>Class "Fisin"</i>
-------	----------------------

---

## Description

Class to manage a [Fis](#) input

## Fields

name [character](#) vector, The name of the input

## Constructors

`FisIn()` The default constructor to build an empty input with the default range [0, 1]

**return:** [FisIn](#) object

`FisIn(minimum, maximum)` The constructor to build an empty input

**argument:** minimum [numeric](#) value, The minimum range value of the input

**argument:** maximum [numeric](#) value, The maximum range value of the input

**return:** [FisIn](#) object

`FisIn(number_of_mfs, minimum, maximum)` The constructor to build an input with a regular standardized fuzzy partition

**argument:** number\_of\_mfs [integer](#) value, The number of Mfs in the fuzzy partition

**argument:** minimum [numeric](#) value, The minimum range value of the input

**argument:** maximum [numeric](#) value, The maximum range value of the input

**return:** [FisIn](#) object

`FisIn(breakpoints, minimum, maximum)` The constructor to build an input with an irregular standardized fuzzy partition

**argument:** breakpoints [numeric](#) vector, The breakpoint values (sorted in ascending order) of the Mfs in the fuzzy partition

**argument:** minimum [numeric](#) value, The minimum range value of the input

**argument:** maximum [numeric](#) value, The maximum range value of the input

**return:** [FisIn](#) object

## Methods

`range()`

**return:** [numeric](#) vector, The range of the input (min max values)

`mf_size()`

**return:** [integer](#) value, The number of Mfs in the input partition

`add_mf(mf)` Add an Mf in the input partition

**argument:** mf [Mf](#) object, The Mf to add

`get_mf(mf_index)`

**argument:** `mf_index` [integer](#) value, The index (1-based index) of the mf to return

**return:** [Mf](#) object

`get_mfs()` Get all mfs in the input

**return:** [list](#) of [Mf](#) objects

`is_standardized()`

**return:** [logical](#) value, TRUE if the input is a standardized fuzzy partition, FALSE otherwise

### See Also

[NewFisIn](#)

[Fuzzy Logic Elementary Glossary](#)

### Examples

```
input <- NewFisIn(0, 2)
input$name <- "foo"
input$add_mf(NewMfTrapezoidalInf(0, 1))
input$add_mf(NewMfTriangular(0, 1, 2))
input$add_mf(NewMfTrapezoidalSup(1, 2))
```

---

FisOut

*Class "FisOut"*

---

### Description

The base class of [Fis](#) output (cannot be instantiate)

Use derived classes [FisOutCrisp](#) or [FisOutFuzzy](#)

### Fields

name [character](#) vector, The name of the output

### Methods

`range()`

**return:** [numeric](#) vector, The range of the output (min max values)

---

FisOutCrisp	Class "FisOutCrisp"
-------------	---------------------

---

## Description

Class to manage a [Fis](#) crisp output

## Fields

defuzzification [character](#) vector, The defuzzification operator of the crisp output  
Allowed values are: "sugeno" (the default) or "MaxCrisp"

disjunction [character](#) vector, The disjunction operator of the crisp output  
Allowed values are: "max" (the default) or "sum"

## Inherits

[FisOutCrisp](#) class inherits all fields and methods of [FisOut](#) class

## Constructors

[FisOutCrisp\(\)](#) The default constructor to build a crisp output with the default range [0, 1]

**return:** [FisOutCrisp](#) object

[FisOutCrisp\(minimum, maximum\)](#) The constructor to build a crisp output

**argument:** minimum [numeric](#) value, The minimum range value of the output

**argument:** maximum [numeric](#) value, The maximum range value of the output

**return:** [FisOutCrisp](#) object

## See Also

[NewFisOutCrisp](#)

[Fuzzy Logic Elementary Glossary](#)

## Examples

```
output <- NewFisOutCrisp(0, 1)
output$name <- "foo"
output$defuzzification <- "sugeno"
output$disjunction <- "max"
```

FisOutFuzzy

Class "FisOutFuzzy"

**Description**

Class to manage a [Fis](#) fuzzy output

**Fields**

defuzzification [character](#) vector, The defuzzification operator of the fuzzy output  
Allowed values are: "sugeno" (the default) "MeanMax", or "area"

disjunction [character](#) vector, The disjunction operator of the fuzzy output  
Allowed values are: "max" (the default) or "sum"

**Inherits**

[FisOutFuzzy](#) class inherits all fields and methods of [FisOut](#) class

**Constructors**

FisOutFuzzy() The default constructor to build a fuzzy output with the default range [0, 1]

**return:** [FisOutFuzzy](#) object

FisOutFuzzy(minimum, maximum) The constructor to build a fuzzy output

**argument:** minimum [numeric](#) value, The minimum range value of the output

**argument:** maximum [numeric](#) value, The maximum range value of the output

**return:** [FisOutFuzzy](#) object

FisOutFuzzy(number\_of\_mfs, minimum, maximum) The constructor to build a fuzzy with a regular standardized fuzzy partition

**argument:** number\_of\_mfs [integer](#) value, The number of Mfs in the fuzzy partition

**argument:** minimum [numeric](#) value, The minimum range value of the output

**argument:** maximum [numeric](#) value, The maximum range value of the output

**return:** [FisOutFuzzy](#) object

FisOutFuzzy(breakpoints, minimum, maximum) The constructor to build a fuzzy with an irregular standardized fuzzy partition

**argument:** breakpoints [numeric](#) vector, The breakpoint values (sorted in ascending order) of the Mfs in the fuzzy partition

**argument:** minimum [numeric](#) value, The minimum range value of the output

**argument:** maximum [numeric](#) value, The maximum range value of the output

**return:** [FisOutFuzzy](#) object



## Methods

`mf_size()`

**return:** `integer` value, The number of Mfs in the output partition

`add_mf(mf)` Add an Mf in the output partition

**argument:** `mf` `Mf` object, The Mf to add

`get_mf(mf_index)`

**argument:** `mf_index` `integer` value, The index (1-based index) of the mf to return

**return:** `Mf` object

`get_mfs()` Get all mfs in the output

**return:** `list` of `Mf` objects

`is_standardized()`

**return:** `logical` value, TRUE if the output is a standardized fuzzy partition, FALSE otherwise

## See Also

[NewFisOutFuzzy](#)

[Fuzzy Logic Elementary Glossary](#)

## Examples

```
output <- NewFisOutFuzzy(0, 2)
output$name <- "foo"
output$defuzzification <- "sugeno"
output$disjunction <- "max"
output$add_mf(NewMfTrapezoidalInf(0, 1))
output$add_mf(NewMfTriangular(0, 1, 2))
output$add_mf(NewMfTrapezoidalSup(1, 2))
```

---

FisPro

*FisPro package*

---

## Description

This package is a basic implementation of the main functions to use a "Fuzzy Inference System" that can be used for reasoning purposes, especially for simulating a physical or biological system. It is derived from the [FisPro](#) open source software. Fuzzy inference systems are briefly described in the [Fuzzy Logic Elementary Glossary](#). They are based on fuzzy rules, which have a good capability for managing progressive phenomena. Fuzzy logic, since the pioneer work by Zadeh, has proven to be a powerful interface between symbolic and numerical spaces. One of the reasons for this success is the ability of fuzzy systems to incorporate human expert knowledge with its nuances, as well as to express the behaviour of the system in an interpretable way for humans. Another reason is the possibility of designing data-driven FIS to make the most of available data.

To design a fuzzy system that can be handled by this package the user can use the [FisPro](#) software. If needed, the package can be extended to other functions.

All the mentioned publications are available from the [FisPro](#) web site.

Enjoy **FisPro**!

**Author(s)**

FisPro Team <contact@fispro.org>

**References**

Guillaume S, Charnomordic B (2011). "Learning interpretable Fuzzy Inference Systems with FisPro." *International Journal of Information Sciences*, **181**(20), 4409–4427. doi: [10.1016/j.ins.2011.03.025](https://doi.org/10.1016/j.ins.2011.03.025), Special Issue on Interpretable Fuzzy Systems, <https://doi.org/10.1016/j.ins.2011.03.025>.

Guillaume S, Charnomordic B (2012). "Fuzzy Inference Systems: an integrated modelling environment for collaboration between expert knowledge and data using FisPro." *Expert Systems with Applications*, **39**(10), 8744–8755. doi: [10.1016/j.eswa.2012.01.206](https://doi.org/10.1016/j.eswa.2012.01.206), <https://doi.org/10.1016/j.eswa.2012.01.206>.

**See Also**

<https://www.fispro.org>

---

Mf

*Class "Mf"*

---

**Description**

The base class of all "membership function" classes (cannot be instantiate)

Use derived classes [MfTriangular](#), [MfTrapezoidal](#), [MfTrapezoidalInf](#) or [MfTrapezoidalSup](#)

**Fields**

label [character](#) vector, The label of the membership function

**Methods**

degree(value) Get the membership degree

**argument:** value [numeric](#) value to compute the membership degree

**return:** [numeric](#) value

**See Also**

[Fuzzy Logic Elementary Glossary](#)

---

MfTrapezoidal	<i>Class "MfTrapezoidal"</i>
---------------	------------------------------

---

**Description**

Class to manage a trapezoidal membership function

**Inherits**

MfTrapezoidal class inherits all fields and methods of [Mf](#) class

**Constructors**

MfTrapezoidal(lower\_support, lower\_kernel, upper\_kernel, upper\_support)

**argument:** lower\_support [numeric](#) lower value of support

**argument:** lower\_kernel [numeric](#) lower value of kernel

**argument:** upper\_kernel [numeric](#) upper value of kernel

**argument:** upper\_support [numeric](#) upper value of support

**return:** [MfTrapezoidal](#) object

**See Also**

[NewMfTrapezoidal](#)

**Examples**

```
mf <- NewMfTrapezoidal(0, 1, 2, 3)
mf$degree(0.5)
```

---

MfTrapezoidalInf	<i>Class "MfTrapezoidalInf"</i>
------------------	---------------------------------

---

**Description**

Class to manage a trapezoidal inf membership function

**Inherits**

MfTrapezoidalInf class inherits all fields and methods of [Mf](#) class

**Constructors**

MfTrapezoidalInf(upper\_kernel, upper\_support)

**argument:** upper\_kernel [numeric](#) upper value of kernel

**argument:** upper\_support [numeric](#) upper value of support

**return:** [MfTrapezoidalInf](#) object

**See Also**

[NewMfTrapezoidalInf](#)

**Examples**

```
mf <- NewMfTrapezoidalInf(0, 1)
mf$degree(0.5)
```

---

MfTrapezoidalSup	<i>Class "MfTrapezoidalSup"</i>
------------------	---------------------------------

---

**Description**

Class to manage a trapezoidal sup membership function

**Inherits**

MfTrapezoidalSup class inherits all fields and methods of [Mf](#) class

**Constructors**

```
MfTrapezoidalSup(lower_support, lower_kernel)
```

**argument:** lower\_support [numeric](#) lower value of support

**argument:** lower\_kernel [numeric](#) lower value of kernel

**return:** [MfTrapezoidalSup](#) object

**See Also**

[NewMfTrapezoidalSup](#)

**Examples**

```
mf <- NewMfTrapezoidalSup(0, 1)
mf$degree(0.5)
```

---

MfTriangular	<i>Class "MfTriangular"</i>
--------------	-----------------------------

---

**Description**

Class to manage a triangular membership function

**Inherits**

MfTriangular class inherits all fields and methods of [Mf](#) class

**Constructors**

MfTriangular(lower\_support, kernel, upper\_support)

**argument:** lower\_support [numeric](#) lower value of support

**argument:** kernel [numeric](#) value of kernel

**argument:** upper\_support [numeric](#) upper value of support

**return:** [MfTriangular](#) object

**See Also**

[NewMfTriangular](#)

**Examples**

```
mf <- NewMfTriangular(0, 1, 2)
mf$degree(0.5)
```

---

NewFis	<i>Create object of class "Fis"</i>
--------	-------------------------------------

---

**Description**

Function to create object of class [Fis](#)

**Usage**

```
NewFis(...)
```

**Arguments**

... arguments of [Fis](#) constructor

**Value**

[Fis](#) object

---

NewFisIn	<i>Create object of class "FisIn"</i>
----------	---------------------------------------

---

**Description**

Function to create object of class [FisIn](#)

**Usage**

```
NewFisIn(...)
```

**Arguments**

... arguments of [FisIn](#) constructor

**Value**

[FisIn](#) object

---

NewFisOutCrisp	<i>Create object of class "FisOutCrisp"</i>
----------------	---

---

**Description**

Function to create object of class [FisOutCrisp](#)

**Usage**

```
NewFisOutCrisp(...)
```

**Arguments**

... arguments of [FisOutCrisp](#) constructor

**Value**

[FisOutCrisp](#) object

---

NewFisOutFuzzy	<i>Create object of class "FisOutFuzzy"</i>
----------------	---

---

**Description**

Function to create object of class [FisOutFuzzy](#)

**Usage**

```
NewFisOutFuzzy(...)
```

**Arguments**

... arguments of [FisOutFuzzy](#) constructor

**Value**

[FisOutFuzzy](#) object

---

NewMfTrapezoidal	<i>Create object of class "MfTrapezoidal"</i>
------------------	---

---

**Description**

Function to create object of class [MfTrapezoidal](#)

**Usage**

```
NewMfTrapezoidal(...)
```

**Arguments**

... arguments of [MfTrapezoidal](#) constructor

**Value**

[MfTrapezoidal](#) object

NewMfTrapezoidalInf    *Create object of class "MfTrapezoidalInf"*

---

**Description**

Function to create object of class [MfTrapezoidalInf](#)

**Usage**

```
NewMfTrapezoidalInf(...)
```

**Arguments**

...                    arguments of [MfTrapezoidalInf](#) constructor

**Value**

[MfTrapezoidalInf](#) object

---

NewMfTrapezoidalSup    *Create object of class "MfTrapezoidalSup"*

---

**Description**

Function to create object of class [MfTrapezoidalSup](#)

**Usage**

```
NewMfTrapezoidalSup(...)
```

**Arguments**

...                    arguments of [MfTrapezoidalSup](#) constructor

**Value**

[MfTrapezoidalSup](#) object



---

NewMfTriangular	<i>Create object of class "MfTriangular"</i>
-----------------	--

---

**Description**

Function to create object of class [MfTriangular](#)

**Usage**

```
NewMfTriangular(...)
```

**Arguments**

... arguments of [MfTriangular](#) constructor

**Value**

[MfTriangular](#) object

---

NewRule	<i>Create object of class "Rule"</i>
---------	--------------------------------------

---

**Description**

Function to create object of class [Rule](#)

**Usage**

```
NewRule(...)
```

**Arguments**

... arguments of [Rule](#) constructor

**Value**

[Rule](#) object

---

Rule

Class "Rule"

---

## Description

Class to manage a [Fis](#) rule

## Fields

premises [integer](#) vector, The premises of the rule

A premise is the 1-based index of MF in the [FisIn](#)

0 means the input is not taken into account for this rule, i.e. the rule is incomplete

The vector length must be equal to the number of inputs in the [Fis](#)

conclusions [numeric](#) vector, The conclusions of the rule

A conclusion is a [numeric](#) value for crisp output [FisOutCrisp](#), or the 1-based index of MF in the fuzzy output [FisOutFuzzy](#)

The vector length must be equal to the number of outputs in the [Fis](#)

## Constructors

`Rule()` The default constructor to build an empty rule

The rule is initialized with empty premises and conclusions

**return:** [Rule](#) object

`Rule(premises, conclusions)` The constructor to build a rule

**argument:** premises [integer](#) vector, The premises of the rule (the vector length must be equal to the number of inputs in the [Fis](#))

**argument:** conclusions [numeric](#) vector, The conclusions of the rule (the vector length must be equal to the number of outputs in the [Fis](#))

**return:** [Rule](#) object

## See Also

[NewRule](#)

[Fuzzy Logic Elementary Glossary](#)

## Examples

```
rule1 <- NewRule()
rule1$premises <- c(1, 2, 0)
rule1$conclusions <- c(1, 2)

rule2 <- NewRule(c(2, 1, 1), c(2, 1))
```

# Index

character, [2](#), [3](#), [5–8](#), [10](#)

data.frame, [3](#)

Fis, [2](#), [2](#), [3](#), [5–8](#), [13](#), [18](#)  
FisIn, [3](#), [5](#), [5](#), [14](#), [18](#)  
FisOut, [3](#), [6](#), [7](#), [8](#)  
FisOutCrisp, [6](#), [7](#), [7](#), [14](#), [18](#)  
FisOutFuzzy, [6](#), [8](#), [8](#), [15](#), [18](#)  
FisPro, [9](#)  
FisPro-package (FisPro), [9](#)

integer, [3](#), [5](#), [6](#), [8](#), [9](#), [18](#)

list, [3](#), [6](#), [9](#)  
logical, [6](#), [9](#)

matrix, [3](#)  
Mf, [5](#), [6](#), [9](#), [10](#), [11–13](#)  
MfTrapezoidal, [10](#), [11](#), [11](#), [15](#)  
MfTrapezoidalInf, [10](#), [11](#), [11](#), [16](#)  
MfTrapezoidalSup, [10](#), [12](#), [12](#), [16](#)  
MfTriangular, [10](#), [13](#), [13](#), [17](#)

NewFis, [4](#), [13](#)  
NewFisIn, [6](#), [14](#)  
NewFisOutCrisp, [7](#), [14](#)  
NewFisOutFuzzy, [9](#), [15](#)  
NewMfTrapezoidal, [11](#), [15](#)  
NewMfTrapezoidalInf, [12](#), [16](#)  
NewMfTrapezoidalSup, [12](#), [16](#)  
NewMfTriangular, [13](#), [17](#)  
NewRule, [17](#), [18](#)  
numeric, [3](#), [5–8](#), [10–13](#), [18](#)

Rule, [3](#), [17](#), [18](#), [18](#)