2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

# Learning from Data Using the R Package "*frbs*"

Lala Septem Riza, Christoph Bergmeir, Francisco Herrera, and Jose Manuel Benítez

***Abstract*— Learning from data is a process to construct a model according to available training data so that it can be used to make predictions for new data. Nowadays, several software libraries are available to carry out this task. *frbs* is an *R* package which is aimed to construct models from data based on fuzzy rule based systems (FRBSs) by employing learning procedures from Computational Intelligence (e.g., neural networks and genetic algorithms) to tackle classification and regression problems. For the learning process, *frbs* considers well-known methods, such as Wang and Mendel's technique, ANFIS, HyFIS, DENFIS, subtractive clustering, SLAVE, and several others. Many options are available to perform conjunction, disjunction, and implication operators, defuzzification methods, and membership functions (e.g., triangle, trapezoid, Gaussian, etc). It has been developed in the *R* language which is an open-source analysis environment for scientific computing. In this paper, we also provide some examples on the usage of the package and a comparison with other software libraries implementing FRBSs. We conclude that *frbs* should be considered as an alternative software library for learning from data.**

## I. Introduction

FUZZY RULE BASED SYSTEMS (FRBSs) are well-known methods within Computational Intelligence, based on fuzzy concepts proposed by L. Zadeh [1] to address complex real-world problems containing uncertainty, imprecision, and non-linearity. They aim at representing knowledge in a set of fuzzy rules. Therefore, FRBSs can be seen as an extension of classical rule-based systems (also known as production systems or expert systems). Basically, they are expressed in the form "IF A THEN B" where A and B are fuzzy sets. A and B are called the antecedent and consequent parts of the rule, respectively.

The knowledge represented by a set of rules can be constructed by human experts based on their experiences. However, we may face many difficulties in this approach, such as there are no available human experts and difficulties in representing their knowledge. Another way that we consider here is to generate rules automatically from data by employing learning methods.

The *frbs* package [2], which we present in this paper, is an *R* package not only providing prominent FRBS models but also implementing widely used learning procedures in FRBSs. Furthermore, even though we focus on learning from data by employing various learning methods, we facilitate users to build an FRBS model manually from knowledge of human experts. The package is implemented in the *R* programming language and is available from the Comprehensive R Archive Network (CRAN) [2].

*R* is a widely used free and open-source analysis environment for scientific computing and visualization. It was introduced by R. Ihaka and R. Gentleman in their paper in 1996 [3]. *R* is available under the GNU General Public License (GPL) and it is maintained and enhanced by the R Development Core Team [4]. However, a wide range of institutions, universities, companies, and individual scientists constributes mainly in the form of packages.

The remainder of this paper is structured as follows. Section II gives briefly an introduction to *R* and its ecosystem. Section III presents the package architecture and its implementation. Some examples are shown in Section IV. Also, we compare the package with other available fuzzy tools in Section V. Finally, Section VI concludes the paper.

## II. *R* and its Ecosystem

### A. *Introduction to* R

*R* is an analysis environment introduced by R. Ihaka and R. Gentleman under the term of the GPL License used for scientific computing and visualization [3]. It is able to be installed in various operating system such as Linux, Mac OS X, Solaris, and MS Windows.

To obtain the *R* software, we simply go to the site http://www.r-project.org and then follow the download instructions. As of this writing, the current version of *R* is 3.0.3. Since *R* is a command-line environment, the prompt symbol will be shown after installing and running *R*. We can operate simple computations (e.g., addition, multiplication, etc) by typing commands interactively at the *R* prompt. Also, we can create a text file containing the commands or functions. A comprehensive introduction to the *R* language can be found in e.g.,[5].

### B. R *Ecosystem*

*R* is constantly growing and now supplying a wide variety of statistical and graphical techniques, and includes many other areas, such as data mining, machine

Authors affiliated with the Department of Computer Science and Artificial Intelligence, E.T.S. de Ingenierías Informática y de Telecomunicación, CITIC-UGR, University of Granada (email: {lala.s.riza, c.bergmeir, herrera, j.m.benitez}@decsai.ugr.es).

This work was partially supported by the Spanish Ministry of Education, Science and Technology under Project TIN2011-28488, the Andalusian Research Plan P10-TIC-6858, P11-TIC-7765, and P11-TIC-9704, and Regional Project P12-TIC-2958. Lala Septem Riza would like to express his gratitude to the Dept. of Computer Science, Universitas Pendidikan Indonesia, for supporting him to pursue the PhD program, and to the Directorate General of Higher Education of Indonesia, for providing a PhD scholarship.

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

learning, pattern recognition, bioinformatics, etc. The main characteristics of *R* are that (i) it is open source, (ii) available for multiple platforms, and (iii) is easy extended by the *R* community in terms of packages.

A major benefit working with *R* is that it provides a scientific ecosystem of packages contributed by researchers and practitioners. Now, there are over 5000 packages available which are classified into more than 20 task views. For instance, the task view of Machine Learning and Statistical Learning contains more than 30 packages for, e.g., neural networks, recursive partitioning, random forests, support vector machines, and kernel methods, etc.

Mostly, packages developed in the *R* framework are included in CRAN and the Bioconductor project. They can be found at http://cran.r-project.org/ and http://www.bioconductor.org/. One remark showing how packages in *R* are kept in good quality is that every package submitted into the repositories is checked manually and must meet a standard quality, such as representative documentation, running on any operating systems (e.g., MS Windows, Mac OS X, Linux), etc.

## III. The *frbs* Package

The *frbs* package was written in pure *R*. It provides over ten learning methods to construct FRBS models in regression and classification tasks from available data [2]. Currently, *frbs* is in version 2.2-0.

*frbs* allows the user to work with several model types. Examples are (i) the Mamdani model [6], [7] which is built by linguistic values in both the antecedent and consequent parts of rules and (ii) the Takagi Sugeno Kang model [8], [9]. Instead of working on linguistic values on the consequent part as in the Mamdani model, the latter one constructs rules whose consequent parts are represented by a function of input variables. Other models that are implemented in the package are, e.g., approximate Mamdani, fuzzy rule-based classification systems (FRBCS), and lateral tuning models. The detailed description of the models can be found in [2].

Regarding learning approaches to construct the FRBS model, *frbs* classifies them into five groups:

1) FRBS based on space partition: it refers to any approach using a strategy of splitting the variable space, and then considering these partitions to obtain parameters of membership functions. We have implemented Wang and Mendel's technique (WM) [10], the FRBCS using Chi's method (FRBCS.CHI) [11], and the FRBCS using Ishibuchi's method with weight factor (FRBCS.W) [12].
2) FRBS based on gradient descent: it refers to approaches using the gradient descent approach to optimize parameters on both the antecedent and consequent parts of rules, for example fuzzy inference rules with descent method (FIR.DM) [13] and the FRBS using heuristics and the gradient descent method (FS.HGD) [14].
3) FRBS based on genetic algorithms: it refers to the genetic fuzzy systems (GFS) which is a combination of FRBSs with genetic algorithms where the genetic algorithms are used to search and optimize parameters of membership functions and of the fuzzy rule construction process [15], [16]. We have implemented the genetic fuzzy system based on Thrift's method (GFS.Thrift) [17], the genetic fuzzy systems for fuzzy rule learning based on the MOGUL methodology (GFS.FR.MOGUL) [18], Ishibuchi's method based on the genetic cooperative competitive learning (GFS.GCCL) [19], Ishibuchi's method based on the hybridization of genetic cooperative competitive learning (GCCL) and Pittsburgh (FH.GBML) [20], genetic lateral tuning and rule selection of linguistic fuzzy systems (GFS.LT.RS) [21], and the structural learning algorithm on vague environment (SLAVE) [22]. In the case of the GFS.GCCL and FH.GBML algorithms, they introduce a new term which is *don't care* for simplification rule bases. Furthermore, FH.GBML provides the parameter *max.num.rule* for defining the maximum number of rules.
4) FRBS based on neural networks: it combines FRBSs with the neural network framework. In this group, we have considered the adaptive-network-based fuzzy inference system (ANFIS) [23] and the hybrid neural fuzzy inference system (HYFIS) [24].
5) FRBS based on clustering: it refers to FRBSs constructed by clustering approaches through representing cluster centers as fuzzy rules. We have included the subtractive clustering [25] and the dynamic evolving neural fuzzy inference system (DENFIS) [26].

Besides the above algorithms considered in *frbs*, the package allows us to assign various values for triangular norm (*t*-norm) and *s*-norm operators, implicator functions, defuzzification methods, and membership functions. For example, we have implemented *minimum*, *Hamacher*, *Yager*, *product*, and *bounded product* as tnorm operators. Additionally, for performing fuzzification we have considered *triangle*, *trapezoid*, *Gaussian*, *sigmoid*, and *general bell* to be membership functions.

As we mentioned before, *frbs* facilitates human experts to express their knowledge by defining a *matrix* representing membership functions and a set of fuzzy rules. After defining the knowledge, we are able to perform reasoning/inference over new data. Also, we can combine an FRBS model by learning from data and human experts into a single model to obtain a reasonable result. It is easy to do since *frbs* employs the *matrix* data structure to represent objects in the FRBS model. In addition, we are allowed to put hedge linguistics (i.e., *extremely*, *very*, *somewhat*, and *slightly*) and the *don't care* term in constructing fuzzy rules.

A key advantage of using *frbs* is that one can easily

compare the performance of FRBS models against an extensive set of models already available through other *R* packages. They can also be combined to form collaborative, improved solutions. Furthermore, because of the GPL License, other researchers collaborate freely by giving useful feedbacks, contributing their methods into the package, and modifying current methods to tackle their specific problems. From programming perspective, we can also embed code written in other programming languages such as *C*, *C++*, and *FORTRAN* into *R* packages. Additionally, several packages have implemented fuzzy theories, for example the *sets* package implements the fundamental concepts of the fuzzy theory [27].

## IV. Usage Examples

*frbs* contains many functions to implement a particular task in FRBSs. The main interface of the package is rather simple and shown in Table I. So, instead of having to handle all available functions, common users just need to take these into account.

TABLE I
The main functions of the package

| The function | Description |
|---|---|
| *frbs.learn()* | The main function of the package to construct an FRBS model automatically from data. |
| *predict.frbs()* (or *predict()*) | This function performs fuzzy reasoning to obtain predicted values for new data, using a given FRBS model. |
| *frbs.gen()* | This function can be used to construct the FRBS model manually from expert knowledge. |
| *summary.frbs()* | Show a summary of an FRBS model. |
| *plotMF()* | Plot the membership functions. |

In order to use *frbs*, basically the following should be considered.

1) Install and load *frbs*. We need to install *frbs* at the first time use only. We can install it from CRAN directly or from a local file in, e.g., .zip and .tar.gz formats.
2) Prepare data. Usually, data are splitted into two parts: training and testing. Data are not allowed to contain missing values and should be in a *matrix* or *data.frame* type.
3) Construct an FRBS model. We construct an FRBS model by executing *frbs.learn()*.
4) Prediction/inference for new data. We predict new data by calling *predict()*.
5) Report or summarize the model. The package provides functions to make a summary of the model and plot the membership functions.

The following examples show the use of *frbs* in regression and classification problems. Before using *frbs*, firstly we need to install it from CRAN by the following simple command in the *R* environment.

```
R> install.packages("frbs")
```

After installing the package, in any session using *frbs* we need to load it with the command:

```
R> library(frbs)
```

which makes any functions of *frbs* available in the *R* environment. We can see a list of functions included in *frbs* by typing the following code.

```
R> library(help=frbs)
```

All *R* functions available in *frbs* are documented in the *R* hypertext and pdf format. The manual of *frbs* in pdf format can be found in [2]. Furthermore, to get information of a particular function, we can apply the *help* command as follows:

```
R> help(frbs.learn)
```

### A. Regression

In this section, we describe how to use *frbs* to predict real-valued output based on the input variables expressed by a continuous function. The following is a function called the *four hill* function. It involves two input variables $x \in [-2, 2]$ and $y \in [-2, 2]$.

$$f(x, y) = \frac{1}{x^4 + y^4 - 2x^2 - 2y^2 + 3}$$

We need to generate data according to the function in a matrix format as follows. Here, we are using step size 0.14 and assigning the output to $z$. We will obtain a matrix containing 841 rows and 3 columns representing the $X$, $Y$, and $Z$ variables.

```
R> fun <- function(input.xy){
+          z <- 1/(input.xy[1]^4+input.xy[2]^4
+          -2*input.xy[1]^2-2*input.xy[2]^2+3)
+  }
R> input.xy <- expand.grid(seq(-2, 2, by = 0.14),
+    seq(-2, 2, by = 0.14))
R> z <- apply(input.xy, 1, fun)
R> data <- cbind(input.xy, z)
R> colnames(data)<- c("X","Y","Z")
```

After that, we split the data into two parts: training data and testing data. We use 80% of the data for training, and the rest for testing.

```
R> cut.indx <- round(0.8*nrow(data))
R> data.tra <- data[1:cut.indx,]
R> data.tst <- data[(cut.indx+1):nrow(data),1:2]
R> real.val <- data[(cut.indx+1):nrow(data),3,
+    drop=FALSE]
```

Then, we need to calculate the interval of each variable by

```
R> range.data <-apply(data,2,range)
```

So, now our data is ready to use.

In order to construct an FRBS model, we need to assign values to available parameters. All parameters have default values, if we ignore them. For instance, we use the Wang and Mendel's algorithm ("WM") as the learning method and assign it to the parameter

*method.type*. Then, we define other parameters in the *control* parameter, for instance the number of linguistic values, 5. And, we use the center of gravity ("COG"), "MIN", "MAX", and "LUKASIEWICZ" to be our defuzzification method, types of $t$-norm, $s$-norm, and implicator operators, respectively. Finally, let us call our simulation "fourhill" by assigning the parameter *name*.

```
R> method.type <- "WM"
R> control <- list(num.labels = 5,
+    type.mf = "GAUSSIAN", type.defuz = "COG",
+    type.tnorm = "MIN", type.snorm = "MAX",
+    type.implication.func = "LUKASIEWICZ",
+    name="fourhill")
```

It is a simple way to execute the learning method as follows.

```
R> mod.reg <-frbs.learn(data.tra,range.data,
+  method.type,control)
```

Even though we do not display the resulting model because of the limited space here, we can summarize our model by the following command.

```
R> summary(mod.reg)
```

The reader can also refer to the project web site http://dicits.ugr.es/software/FRBS/ in order to see the model. And, we plot the membership functions as seen in Figure 1 by

```
R> plotMF(mod.reg)
```

The final step is to predict testing data using the *predict()* function. It needs two arguments which are *mod.reg* and new data. It can be done as follows.

```
R> res.test <-predict(mod.reg,data.tst)
```

The predicted values are generated in matrix format. They can be compared with the real values using the mean square error (MSE) by

```
R> err.MSE <- mean((real.val-res.test)^2)
R> print(err.MSE)
[1] 0.07852261
```

### B. Classification

In this example, we are using the *iris* data set which is already included in the *R* environment. The *iris* data set is a well-known data set in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other. To use it, we just load the data by the command:

```
R> data(iris)
```

To get a relatively good proportion, usually we randomize the data by

```
R> set.seed(2)
R> irisShuffled <-iris[sample(nrow(iris)),]
```

Because the decision attribute, which is in the last column is represented in a string, we need to convert it into numerical values. Then, the data are split into two parts which are *tra.iris* for training data and *tst.iris* for testing ones.

```
R> irisShuffled[,5] <-unclass(irisShuffled[,5])
R> tra.iris <-irisShuffled[1:105,]
R> tst.iris <-irisShuffled[106:nrow(irisShuffled)
+    ,1:4]
R> real.iris <-matrix(irisShuffled
+    [106:nrow(irisShuffled),5], ncol = 1)
```

Then, even though *frbs* by default calculates the range of the input data, we strongly recommend to define it manually.

```
R> range.data.input <-apply(iris[,-ncol(iris)],
+    2,range)
```

It should be noted that for classification tasks we only need to define the range of input data.

As in the regression example, after our data is ready to use, we need to define some parameters concerning the used method and its *control* parameter. For example, we are using the FRBCS.CHI method and we define three linguistic values, the trapezoid to be the membership function, and minimum, maximum and Zadeh for the types of $t$-norm, $s$-norm, and implicator operators, respectively.

```
R> method.type <-"FRBCS.CHI"
R> control <-list(num.labels = 3,
+    type.mf ="TRAPEZOID", type.tnorm = "MIN",
+    type.snorm = "MAX",
+    type.implication.func = "ZADEH")
```

We generate an FRBS model through the following command.

```
R> mod.class <-frbs.learn(tra.iris,
+  range.data.input, method.type, control)
```

As in the regression example, we do prediction as follows:

```
R> res.test <- predict(mod.class, tst.iris)
```

Then, we can check the result by calculating the percentage error:

```
R> err = 100*sum(real.iris!=res.test)/
+    nrow(real.iris)
R> print(err)
[1] 4.444444
```

The plot of membership functions can be seen in Figure 2. Further information and examples can be found in our project web site.

## V. Comparison with Other Fuzzy Tools

In this section, we review well-known software libraries implementing FRBS concepts: Xfuzzy [28], Fuzzy Logic Toolbox for *MATLAB* [29], Fuzzy Inference System Professional (FisPro) [30], Generating Understandable and Accurate Fuzzy Models in a Java Environment (GUAJE) [31], and Knowledge Extraction based on Evolutionary Learning (KEEL) [32], [33]. All of them support learning from data using various learning procedures.

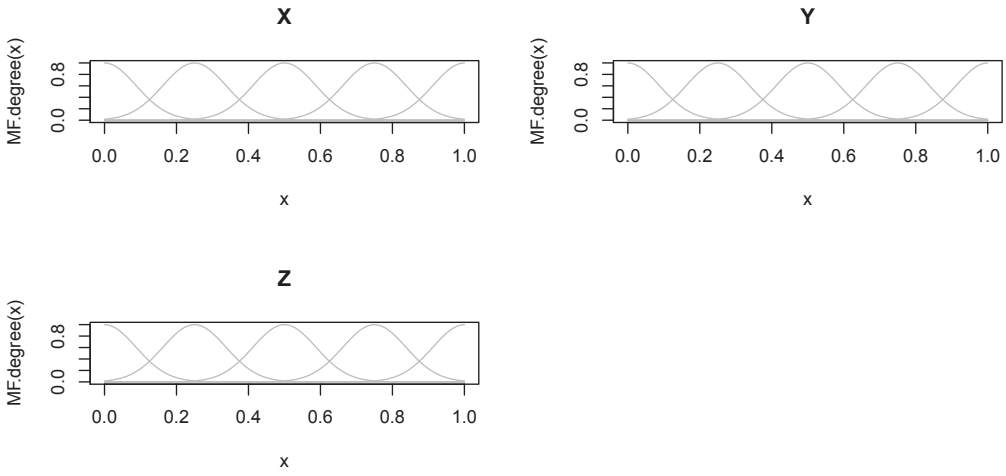


Fig. 1
The plot of membership functions in the regression example.

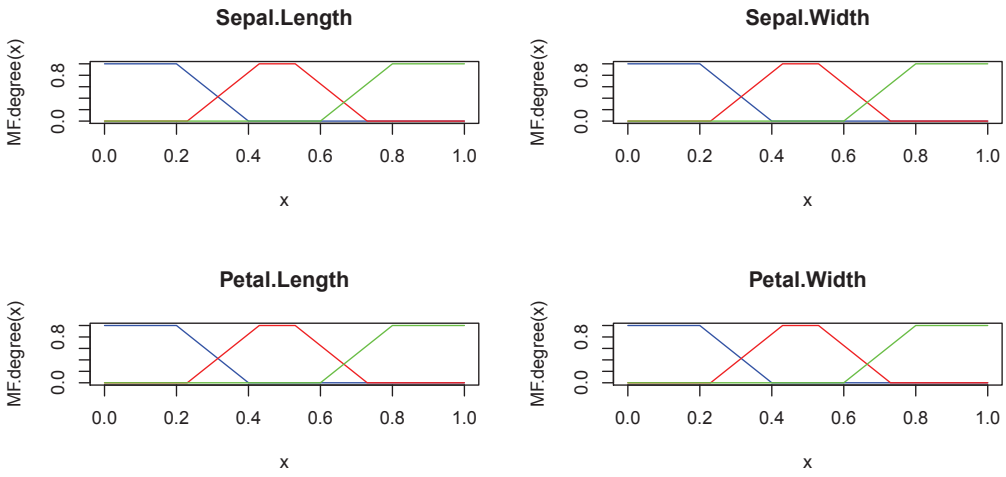


Fig. 2
The plot of membership functions in the classification example.

Xfuzzy is an open-source framework under the term of the GPL License based on fuzzy inference-based systems [28]. The software has an architecture containing several parts which share the proposed language *XFL3*. Using the graphical user interface, these parts have different functionalities, such as *xfedit* which can be used to describe the logical structure needed for the inference process. It is quite similar to the *frbs.gen()* function in *frbs* to perform inference based on knowledge constructed manually by human experts. *xfsl* is a tool used to extract knowledge from data. Therefore, it is obvious that *xfsl* has functionalities similar to *frbs.learn()* in *frbs*. Some learning methods have been considered in this part, such as gradient descent, second-order, Gauss-Newton, and statistical algorithms. It can be seen that *frbs* offers more algorithms for generating FRBS models. For the inference process, Xfuzzy provides the *xfmt* tool which is the same as *predict()* in *frbs*. Other capabilities of Xfuzzy are, e.g., plotting membership functions and converting codes using *xfc*, *xfcpp*, and *xfj* into *XFL3*. Furthermore, many options for $t$-norm, $s$-norm, and implicator operators and defuzzification methods are available as well.

The Fuzzy Logic Toolbox for *MATLAB* is a toolkit for analysis, design, and simulation systems based on fuzzy logic. It provides Simulink, a graphical user interface (GUI), and a command line mode to build FRBS models

[29]. It supports standard Mamdani and Sugeno-type fuzzy inference systems. In order to design an FRBS model, it allows to use ANFIS, substractive clustering, and fuzzy C-means. Two of the three methods are provided in *frbs*, and we provide several others. Additionally, in this toolbox, the *FISEditor* is used to display general information about a fuzzy inference system while the *Membership Function Editor* and *Rule Editor* are used to provide functions to display and edit membership function and rules. Since *frbs* works with a scripting interface, editing is a straightforward process by changing the matrix of the model.

FisPro, built in C++ and Java, implements fuzzy inference systems considering the rule base interpretability and modularity in an open source software. As *frbs*, it provides two modes of FRBS models which are expert rule design and automatic induction for regression and classification cases. It implements $k$-means, hierarchical fuzzy partitioning (HFP), Wang and Mendel (WM), fast prototyping algorithm (FPA), and fuzzy decision trees (FDT) in order to generate fuzzy partitions and rule bases. Additionally, FisPro provides the aggregation operators "MAX" and "SUM" for conjunction, which are also available in *frbs*. FisPro also includes mechanisms for merging and improving fuzzy rules in a separated part which is an optimization modul. In contrast, *frbs* simultaneously performs optimization along learning processes by using learning methods such as genetic fuzzy systems.

GUAJE is an open-source software under the GPL license that implements fuzzy rule-based systems in Java. It is an extension of the Knowledge Base Configuration Tool (KBCT) [34] and aimed to provide interpretable fuzzy systems. Additionally, it integrates several other software programs, such as FisPro, ORE, Espresso, Graphviz, JMetal, and WEKA. From the perspective of the algorithms used to generate fuzzy rules, GUAJE adopts the approaches implemented in FisPro. Also, generated FRBS models can be exported to FisPro, Xfuzzy, and the Fuzzy Logic Toolbox for *MATLAB*. Although GUAJE provides data pre-processing algorithms (e.g., feature selection) and implements other approaches, *frbs* has the advantage of being built in the *R* environment which offers additional benefits since *R* provides more comprehensive and complete algorithms for data preprocessing and other prediction methods.

KEEL is a big and comprehensive software library containing classical knowledge extraction algorithms, preprocessing techniques (e.g., instance selection, feature selection, discretization, etc.), learning algorithms for clustering, regression, and classification problems, and a statistical test module for comparison [32], [33]. It provides three important blocks: data management, design of experiments, and educational experiments. Therefore, it can be seen that KEEL is intended as a research and educational tool. From the perspective of FRBSs, it is focused on implementation of learning methods based on GFS, such as GFS based on Thrift's algorithm [17], SLAVE [22], etc. Some of the algorithms considered in KEEL are implemented in *frbs* as well. Even though KEEL has implemented more than thirty learning algorithms based on FRBSs, it does not provide construction of FRBS models from human experts as in *frbs*.

## VI. Conclusions

A package that implements fuzzy rule-based systems for the *R* programming language, called *frbs*, has been presented. It includes the most commonly used types of FRBSs, namely, Mamdani and Takagi Sugeno Kang models and several variants, for both classification and regression tasks. An explanation of the advantages of using *R* for analysis and learning from data, especially in supporting the development of *frbs* has been given briefly. Usage examples to get started and a comparison with other software libraries are illustrated as well.

## References


[1] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[2] L. S. Riza, C. Bergmeir, F. Herrera, and J. M. Benitez, "frbs: Fuzzy Rule-Based Systems for Classification and Regression Tasks," http://CRAN.R-project.org/package=frbs, 2014

[3] R. Ihaka and R. Gentleman, "R: A Language for Data Analysis and Graphicss," *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, pp. 299-314, 1996.

[4] R Development Core Team, "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, Austria, http://www.r-project.org/foundation/, 2010.

[5] R Development Core Team, "An Introduction to R," *R Foundation for Statistical Computing*, ISBN. 3-900051-12-7, http://www.R-project.org/, 2008.

[6] E. H. Mamdani, "Applications of Fuzzy Algorithm for Control a Simple Dynamic Plant," *In Proceedings of the IEEE*, vol. 121, pp. 1585-1588, 1974.

[7] E. H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," *Int. J. Man Mach. Stud*, vol. 7, pp. 1-13, 1975.

[8] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 51, no. 1, pp. 116-132, 1985.

[9] M. Sugeno and G.T. Kang, "Structure Identification of Fuzzy Model," *Fuzzy Sets Syst.*, vol. 28, pp. 15-33, 1988.

[10] L. X. Wang and J. M. Mendel, "Generating Fuzzy Rules by Learning from Examples," *IEEE Trans. Systems. Man Cybernet.*, vol. 22, no. 6, pp. 1414-1427, 1992.

[11] Z. Chi, H. Yan, T. Pham, "Fuzzy Algorithms with Applications to Image Processing and Pattern Recognition," *World Scientific*, ISBN 9810226977.

[12] H. Ishibuchi and T. Nakashima, "Effect of Rule Weights in Fuzzy Rule-Based Classification Systems," *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 59-64, 2001.

[13] H. Nomura, L. Hayashi, and N. Wakami, "A Learning Method of Fuzzy Inference Rules by Descent Method," *IEEE International Conference on Fuzzy Systems*, pp. 203-210, 1992.

[14] H. Ishibuchi, K. Nozaki, and H. Tanaka, "Empirical Study on Learning in Fuzzy Systems by Rice Taste Analysis," *Fuzzy Set and Systems*, vol. 64, no. 2, pp. 129-144, 1994.

[15] O. Cordon, F. Herrera, F. Hoffmann, and L. Magdalena, "Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases," *Singapore: World Scientific Publishing*, 2001.

[16] F. Herrera, "Genetic Fuzzy Systems: Taxonomy, Current Research Trends and Prospects," *Evolutionary Intelligence*, vol. 1, pp. 27-46, 2008

[17] P. Thrift, "Fuzzy Logic Synthesis with Genetic Algorithms," *Proc. of the Fourth International Conf. on Genetic Algorhtms (ICGA91)*," pp.509-513, 1991.
[18] F. Herrera, M. Lozano, and J. Verdegay, "A Learning Process for Fuzzy Control Rules Using Genetic Algorithms," *Fuzzy Sets and Systems*, vol. 100, pp. 143-158, 1998.
[19] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance Evaluation of Fuzzy Classifier Systems for Multidimensional Pattern Classification Problems," *IEEE trans. on Systems, Man, and Cybernetics - Part B: Sybernetics*, vol. 29, no. 5, pp. 601-618, 1999.
[20] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Hybridization of Fuzzy GBML Approaches for Pattern Classification Problems," *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 35, no. 2, pp. 359-365, 2005.
[21] R. Alcala, J. Alcala-Fdez, and F. Herrera, "A Proposal for the Genetic Lateral Tuning of Linguistic Fuzzy Systems and Its Interaction with Rule Selection," *IEEE Trans. on Fuzzy Systems*, vol. 15, no. 4, pp. 616-635, 2007.
[22] A. Gonzalez and R. Peréz, "Selection of Relevant Features in a Fuzzy Genetic Learning Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 3, pp. 417-425, 2001.
[23] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp.665-685, 1993.
[24] J. Kim and N. Kasabov, "HyFIS: Adaptive Neuro-Fuzzy Inference Systems and Their Application to Nonlinear Dynamical Systems," *Neural Networks*, vol. 12, no. 9, pp. 1301-1319, 1999.
[25] S. Chiu, "Method and Software for Extracting Fuzzy Classification Rules by Subtractive Clustering," *Fuzzy Information Processing Society, NAFIPS*, pp. 461-465, 1996.
[26] N. Kasabov and Q. Song, "DENFIS: Dynamic Evolving Neural-Fuzzy Inference System and Its Application for Time-Series Prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144-154, 2002.
[27] D. Meyer and K. Hornik, "Generalized and Customizable Sets in R," *Journal of Statistical Software*, vol. 31, no. 2, pp. 1-27, 2009.
[28] I. Baturone, F. J. Moreno-Velo, S. Sánchez-Solano, A. Barriga, P. Brox, A. Gersnoviez, and M. Brox, "Using Xfuzzy Environment for the Whole Design of Fuzzy Systems," *Proc. IEEE International Conference on Fuzzy Systems, London*, pp. 1-6, 2007.
[29] The MathWorks, Inc., "The Fuzzy Logic Toolbox for Use with MATLAB version 2," *The MathWorks, Inc.*, 2002.
[30] S. Guillaume and B. Charnomordic, "Learning Interpretable Fuzzy Inference Systems with FisPro," *Information Sciences*, vol. 181, no. 20, pp. 4409-4427, 2011
[31] J. M. Alonso and L. Magdalena, "Generating Understandable and Accurate Fuzzy Rule-Based Systems in a Java Environment," *Lecture Notes in Artificial Intelligence - 9th International Workshop on Fuzzy Logic and Applications, Springer-Verlag, LNAI6857*, pp. 212-219, 2011.
[32] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, "KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems," *Soft Computing*, vol. 13, no. 3, pp. 307-318, 2009.
[33] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 255-287, 2011.
[34] J. M. Alonso, L. Magdalena, and S. Guillaume, "KBCT: A knowledge extraction and representation tool for fuzzy logic based systems," *IEEE International Conference on Fuzzy Systems*, pp. 989–994, 2004.