A Fuzzy Toolbox for the R Programming Language

Christian Wagner and Simon Miller
Horizon Digital Economy Research /
Intelligent Modelling and Analysis Research Group
University of Nottingham
Nottingham, U.K.
{christian.wagner, s.miller}@nottingham.ac.uk

Jonathan M. Garibaldi
Intelligent Modelling and Analysis Research Group
University of Nottingham
Nottingham, U.K.
jmg@cs.nott.ac.uk

***Abstract*—In this paper, we describe the main functionality of an initial version of a new fuzzy logic software toolkit based on the R language. The toolkit supports the implementation of several types of fuzzy logic inference systems and we discuss and present several aspects of its capabilities to allow the straightforward implementation of type-1 and interval type-2 fuzzy systems. We include source code examples and visualizations both of type-1 and type-2 fuzzy sets as well as output surface visualizations generated using the R toolkit. Finally, we describe the significant benefits of relying on the R language as a language which is employed across several research disciplines (thus enabling access to fuzzy logic tools to a variety of researchers), outline future developments and most importantly call for contributions, comments and feedback to/on this open-source software development effort.**

Keywords; fuzzy inference software; type-1 fuzzy inference; type-2 fuzzy infererence; open-source software; Matlab; R

I. INTRODUCTION

Since its introduction in 1965 [1], fuzzy logic has become widely popular, both as a specific field of research, as well as a tool for the modeling of and computation with data applied in a myriad of disciplines, from the arts, over medical sciences to industrial control. The vast majority of current applications of fuzzy logic are based on type-1 fuzzy logic.

While the research interest in type-2 fuzzy logic has steadily grown in the last 10 years, the number of applications is still very limited. In particular, the majority of existing applications are mostly confined to applications that are research-led, frequently by members of the fuzzy logic research community, rather than third party applications entirely conceived and implemented for example by practitioners in other fields and industry.

It is a well-known phenomenon in computing that an essential criterion for the adoption of a new method or algorithm, the availability of software implementing said algorithm is essential. Analogously, we believe that for the more widespread adoption and application of non-classical fuzzy logic systems (such as non-stationary, interval type-2 and general type-2 fuzzy systems), the provision of free, widely available, open-source software is essential. As part of this paper, we are introducing an open-source software package based on the R language which is approaching the release of its first version.

This work was partly funded by the RCUK's Horizon Digital Economy Research Hub grant, EP/G065802/1.

R is a free, open-source language and importantly, is already used widely across disciplines (from Computer Science to Social Sciences - such as Political Science). As such, providing freely accessible software to employ non-classical fuzzy systems as part of an R package will allow practitioners and researchers from a wide range of areas to make use of the research developed by the fuzzy logic research community who in turn can incorporate the potentially vast amount of feedback and real-world deployment information into their research.

The paper is intended both to introduce the R fuzzy logic toolbox as well as an invitation to the fuzzy community, particularly the type-2 community, to join with the development effort, give feedback and recommendations.

The paper is structured as follows. Section II provides basic background information on the current availability of packages for the development of non-classical fuzzy logic applications as well as the R language in general. Subsequently, Section III describes the R toolbox and reviews some of the currently available functionality. Section IV provides some sample plots of non-classical fuzzy logic membership functions as well as control surfaces, demonstrating the graphical capabilities of the toolbox. Finally, Section V discusses the current development of the toolbox and describes the planned long-term development.

II. BACKGROUND

The use of toolboxes for implementing type-1 fuzzy systems is widespread owing to the availability of such software to facilitate development. Perhaps the most well known of these is the Fuzzy Logic Toolbox provided for MATLAB® (The Mathworks, Inc.) which allows users to create type-1 fuzzy inference systems using MATLAB® command-line functions or a graphical user interface (GUI) [2].

Currently, there is a lack of toolboxes available for creating other types of fuzzy systems including, in particular, type-2 fuzzy systems. We should note that a several researchers (e.g. J. Mendel) and research groups have opened their source code and made it accessible, however the use of source code still requires in-depth familiarity with the material in question (i.e. non-classical fuzzy logic systems) and thus does not fulfill the same role as toolkits in terms of making the material easily accessible for people from a variety of backgrounds.

Castro et al. have previously demonstrated a toolbox for MATLAB® that provides tools for the development of interval

type-2 fuzzy systems [3] which provides a GUI interface similar to that of the MATLAB® Fuzzy Logic Toolbox [2]. While the toolbox allows the implementation of interval type-2 fuzzy systems using MATLAB®, it, being based on the proprietary MATLAB® is restricted in terms of its availability (MATLAB® is subject to license fees) as well as its user base (MATLAB® is predominantly used as part of the physical sciences and engineering). These restrictions as well as the limitation to interval type-2 fuzzy logic limit the adoption potential of the existing MATLAB® based toolbox and currently it is difficult to find examples of its more widespread adoption.

The toolbox introduced as this paper is based on/around the freely available R platform/language. As noted, R offers the advantage of not only being freely available but importantly that it, as a language is accessible to people from a large variety of backgrounds. This in particular should facilitate the experimentation with fuzzy logic and the research and investigation into new problem domains.

Further, the advantage of high-level specialist mathematical languages such as R and MATLAB® is that, in contrast to low-level languages such as C or Java, they provide built-in primitives for representing and manipulating vectors and matrices, which can be used to directly represent and manipulate fuzzy sets. In addition, these languages provide comprehensive built-in graphical capabilities for 2D and 3D plotting, which can be used to visualize fuzzy sets.

R itself is an open-source language and environment for statistical computing and graphics, similar to the S language and environment originally developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. It is a mature, open-source language which is actively maintained by the community. R is similar to MATLAB® in that it is a generic programming environment within which mathematical and statistical techniques are implemented [4]. R is an integrated suite of software facilities for data manipulation, calculation and graphical display, including effective data handling and storage facilities, a suite of operators for calculations on arrays and matrices, a large integrated collection of intermediate tools, graphical facilities for display either on-screen or on hardcopy, and a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities. R can be easily extended via packages. (Text adapted from [4]). R is freely downloadable from the main R Project repository [4], and can easily be installed on many platforms including popular versions of Microsoft Windows, Apple Mac OS X and Linux.

In this paper we present a fuzzy logic toolbox for the programming language R that includes tools for creating type-1, non-stationary [5], interval type-2 [6] and zSlices based general type-2 fuzzy systems [7]. At the present stage, the toolbox is in the final stages of development in preparation for the release of its first version. In particular, features such as the general type-2 aspects are being completed, documentation material is being added. This paper aims to encourage the research community to engage and provide feedback and suggestions for the final release. In the following sections, we provide a high level view of some of the features that are currently included.

III. TOOLKIT FEATURES

The R toolkit has several similarities with the MATLAB® toolbox. This is natural as both toolkits/toolboxes implement established fuzzy logic concepts and algorithms. The similarity between both approaches however has the distinct advantage of greatly facilitating the use of either toolkit for authors familiar with the other one.We proceed by illustrating the similarities based on fundamental fuzzy logic concepts between both toolkits in III.A, followed by a basic overview of some of the currently implemented features in terms of type-1 fuzzy logic in III.B and interval type-2 fuzzy logic in III.C.

Non-stationary fuzzy systems and zSlices based general type-2 fuzzy systems' functionality has largely been included in the toolkit but as part of this paper we will focus on the features of "standard" type-1 and interval type-2 fuzzy systems. We are addressing some features of non-classical fuzzy logic systems in Section IV, in particular demonstrating the visualization capabilities of the current toolkit in terms of 3D set visualizations.

A. Similarity between R and MATLAB® toolkits.

As noted, the R toolkit has been designed to maximize the transparency for people who are already familiar with the parameters and specification of fuzzy sets and systems while also making it easy to learn for people who are new to the area. Incidentally, the command line interface employed in the R language results in a high similarity compared to the command line interface operation of the MATLAB® Fuzzy Logic Toolbox. However, while the R language offers many similar functions to MATLAB® and add-on toolboxes (termed 'packages' in R), the language is a development of the S language and not a 'clone' of MATLAB®. Hence, it has detailed syntactic differences from MATLAB®, as summarized in [8].

For example, in MATLAB®, the command:

```
x = [ 1, 2, 3];
```

creates a vector of length 3, while the equivalent in R is:

```
x = c(1, 2, 3) or   x <- c(1, 2, 3)
```

(these two forms of assignment operator are not completely identical, and the latter version is preferred in the R community). Similarly, the MATLAB® command:

```
x = [ 1, 2, 3; 4, 5 6];
```

creates a matrix of 2 rows by 3 columns, while the equivalent in R for this is:

```
x = rbind(c(1,2,3), c(4,5,6))
```

While there are syntactic difference between R and MATLAB®, in terms of the scope of the implementation the current state of the R toolkit is comparable to the MATLAB® Fuzzy Logic Toolbox in terms of type-1 fuzzy logic, however, the R toolkit further addresses more complex types of fuzzy logic such as interval and general type-2 fuzzy logic. For the

majority of the type-1 functionality, one can establish a direct mapping between the functions provided in the R package and those in the MATLAB® Fuzzy Toolbox as is expected because of the underlying fundamental fuzzy logic theory concepts).

The main functions necessary to create and perform fuzzy inference in Mamdani-style are supplied within the package (note that Sugeno / TSK style inference is not currently supported). A wide variety of forms of membership functions are supported similar to those currently provided by MATLAB®. Additionally, currently the common forms of conjunction, disjunction, implication and aggregation operators are supported (including, minimum, maximum, product and probabilistic OR), as well as Mamdani inference and popular forms of defuzzification as further detailed below.

To illustrate the creation of a complete FLC using the R toolkit we have provided a comparison between implementing a completed fuzzy inference system (FIS) in both MATLAB® and using the R toolkit. The example is based on the 'tipper' example provided as part of the MATLAB® Fuzzy Logic Toolbox documentation. The complete source code to set up the example for both MATLAB® and R is provided in Section VI.A and VI.B respectively. Note the expected high similarity between the constructions of a fuzzy inference system in both platforms. Each example also demonstrates the output capabilities, specifically (we use the R notation here):

- *showfis(fis)*
  Produces a summary of the constructed FIS.
- *showrule(fis)*
  Prints the rules of the specific FIS.
- *evalfis(c(1,2), fis)*
  Evaluates the given FIS using the specified input values.
- *evalfis(rbind(c(3,5), c(2,7)), fis)*
  Evaluates the given FIS for the specified set of inputs.
- *gensurf(fis)*
  Produces the control surface (visualisation) for the given FIS.

Overall, the R toolkit is aimed to generally providing at least the same set of features as are offered as part of the MATLAB® Fuzzy Logic Toolbox. As an open-source toolkit, it is hoped that in the medium and long term the overall feature richness and functionality of the toolkit will be increased through open access and contribution. This could potentially also include further components that employ the provided toolkit, such as a Graphical User Interface (GUI) which could further facilitate the implementation of FLCs in particular for users who are not familiar with the implementation of FLCs and/or have no programming background or interest. Further, it is expected that several community-centred "flavours" of the R toolkit will be developed in time (for example for medical applications, applications in economy, etc.).

In the following Sections we describe some of the basic aspects of creating type-1 and interval type-2 fuzzy systems using the toolkit.

B. Type-1 fuzzy logic features.

As part of describing the type-1 functionality we give examples of the basic membership functions currently supported before briefly reviewing the inference and defuzzification operations.

1) Membership Functions.

a) Triangular

Triangular membership functions are created by providing the left, right and center points of the membership function. For example, the source code snippet below produces the membership function depicted in Figure 1.

```
x <- 1:10
y <- trimf(x,c(2,5,8))
```

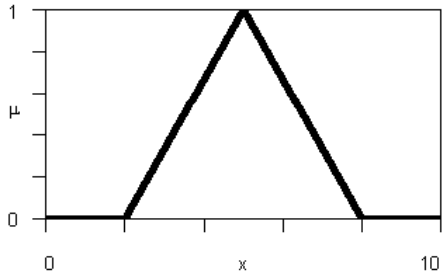

Figure 1 An example of a triangular type-1 membership function produced using the R toolkit.

b) Trapezoidal

For trapezoidal sets, four values describing the left and right slopes of the trapezoid are used to create the membership function. For example, the source code snippet below produces the membership function depicted in Figure 2.

```
x <- 1:10
y <- trapmf(x,c(2,4,6,8))
```

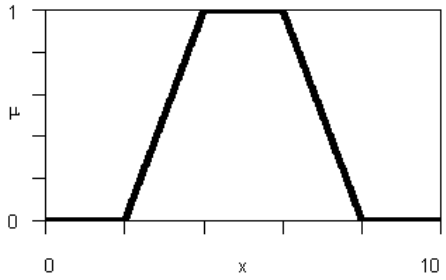

Figure 2 An example of a trapezoidal type-1 membership function produced using the R toolkit.

c) Gaussian

To create a Gaussian membership function the centre of the Gaussian and Sigma (describing the curve) are required. For example, the source code snippet below produces the membership function depicted in Figure 3.

```
x <- 1:10
y <- gaussmf(x,c(2,5))
```

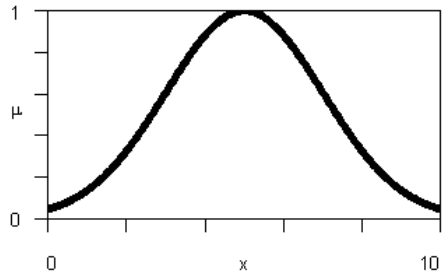

Figure 3 An example of a Gaussian type-1 membership function produced using the R toolkit.

d) Sigmoidal

Sigmoidal membership functions are created using two values representing the centre and width of the slope. For example, the source code snippet below produces the membership function depicted in Figure 4.

x <- 1:10

y <- sigmf(x,c(2,5))

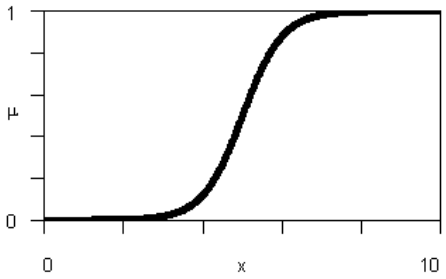

Figure 4 An example of a sigmoidal type-1 membership function produced using the R toolkit.

2) Inference Mechanisms.

The inference mechanisms provided as part of the R toolkit are comparable to those provided by the MATLAB® Fuzzy Logic Toolbox. Specifically, during the rule specification, each rule can be associated with a given weight and the logical connective (AND or OR) can be specified. Considering the 'tipper' example provided in Section VI, we have the example rule specification:

rules= rbind(c(1,1,1,1,2), c(2,0,2,1,1), c(3,2,3,1,2))

The example defines three rules where each rule is described by five parameters (in our example). The parameters are specified analogous to the MATLAB® Fuzzy Logic Toolbox as follows (adopted from [9]) , where *m* is the number of inputs and *n* is the number of outputs:

- The first parameter is the index number for the membership function associated with input 1.
- The second is the index number for the membership function associated with input 2, and so on.
- The next *n* parameters work the same way for the outputs.
- Parameter *m* + *n* + 1 is the weight associated with that rule (typically 1) and parameter *m* + *n* + 2 specifies the connective used (where AND = 1 and OR = 2).

3) Defuzzification Methods.

The following defuzzification methods are currently available as part of the R toolkit:

a) Centroid

This method takes the centre of the area of a membership function on the x-axis.

b) Bisector

Bisector splits the set into two sections of equal area, and takes the vertical that separates the two sections.

c) Mean of maximum

Mean of maximum (MOM) uses the mean value of all discrete x coordinates associated with the maximum membership for the given output set.

d) Smallest of maximum

Smallest of maximum (SOM) uses the smallest value of all discrete x coordinates associated with the maximum membership for the given output set.

e) Largest of maximum

Largest of maximum (LOM) uses the largest value of all discrete x coordinates associated with the maximum membership for the given output set.

Having considered the basic features for type-2 FISs as part of the R toolkit, we proceed to briefly review the currently available interval type-2 features.

C. Interval type-2 fuzzy logic features.

The features available in terms interval type-2 fuzzy systems as part of the R toolkit are briefly summarized below. Specifically, we address the available membership functions in Section III.C.1, followed by the inference capabilities in Section III.C.2 and the type-reduction/defuzzification methods in III.C.3.

1) Membership Functions.

a) Triangular

Type-2 triangular membership functions are created by providing the width of the footprint of uncertainty (FOU), and the left, right and center points of the membership function. For example, the source code snippet below produces the membership function depicted in Figure 5.

x <- 1:10

y <- tri2mf(x,c(1,2,5,8))

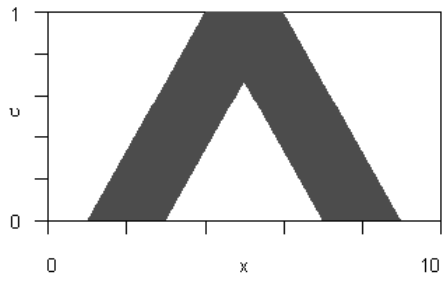

Figure 5 An example of a triangular interval type-2 membership function produced using the R toolkit.

b) Trapezoidal

For type-2 trapezoidal sets, 5 values are required: the width of the FOU, the x coordinate of the leftmost point of the lower membership function, the x coordinate of the rightmost lower membership function, the gradient of the left slope and the gradient of the right slope. For example, the source code snippet below produces the membership function depicted in Figure 5.

```
x <- 1:10
y <- trapmf2(x, c(1,3,7,1,1))
```

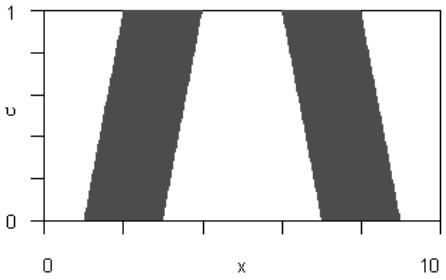

Figure 6 An example of a trapezoidal interval type-2 membership function produced using the R toolkit.

c) Gaussian

To create a Type-2 Gaussian membership function two values of the standard deviation describing the lower and upper membership functions as well as the centre of the Gaussian are required. For example, the source code snippet below produces the membership function depicted in Figure 7.

```
x <- 1:10
y <- gausst2mf(x,c(1,1.5,5))
```

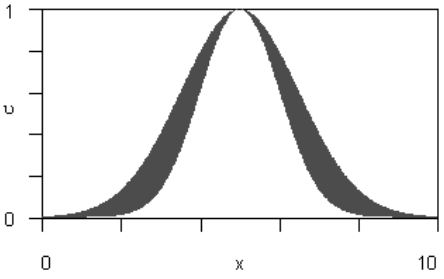

Figure 7 An example of a Gaussian interval type-2 membership function produced using the R toolkit.

2) Inference Mechanisms.

The inference mechanisms are identical to those described in Section III.B.2 for type-1 fuzzy systems but applied to interval type-2 fuzzy sets as described for example in [10].

3) Type-Reduction and Defuzzification.

Type-reduction is currently conducted based on the Karnik-Mendel iterative procedure [11]. The type-reduced set is finally defuzzified by computing the average of its endpoints as in [10].

IV. Sample Visualisations

One of the strengths of R and similar languages is the powerful set of visualization features available. We provide both sample visualizations of linguistic variables (and membership functions) in Section IV.A as well as a comparison of output surface visualizations between the MATLAB and R implementations for the 'tipper' example in Section IV.B.

For all visualizations, the R environment supports a variety of formats for both soft-copy and hard-copy output. The precise details of any implementation depend on the underlying operating system, but in most platforms the production of *jpeg* or *png* graphical formats and *postscript* of Adobe® *pdf* formats is supported.

A. *Linguistic Variable and Membership Function Visualisation.*

The current R toolkit contains a series of functions for plotting membership functions. Figure 8 and Figure 9 depict examples of visualizing the linguistic variable "Temperature" based on the three linguistic labels *cool*, *warm* and *hot*. The linguistic labels are represented by Gaussian Type-1 and Interval Type-2 membership functions in Figure 8 and Figure 9 respectively.

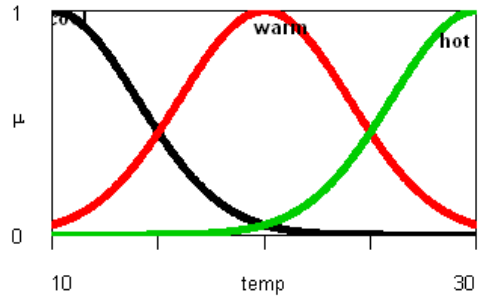

Figure 8 An example of the plotting of multiple type-1 membership functions using the R toolkit.

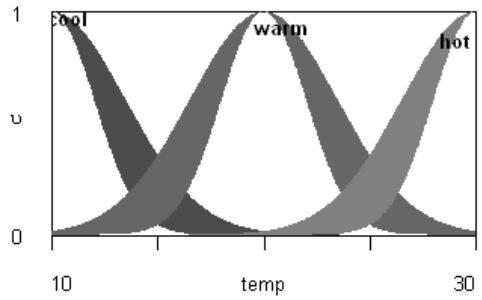

Figure 9 An example of the plotting of multiple interval type-2 membership functions using the R toolkit.

By utilizing the recently released `RGL` library, it is possible to create three dimensional plots of both interval type-2 and general type-2 membership functions. Examples of both are includes as part of Figure 10 and Figure 11 respectively.

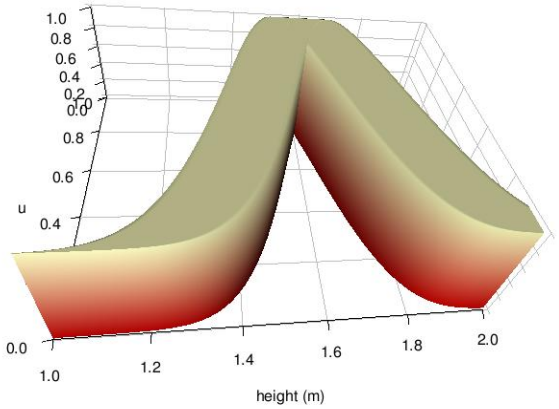

Figure 10 An example of a interval type-2 Gaussian membership function produced using the R toolkit. (shown in 3D)

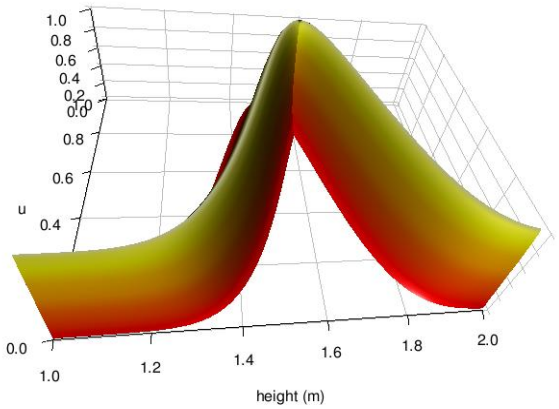

Figure 11 An example of a general type-2 Gaussian membership function produced using the R toolkit.

As well as the visualization of membership functions, the R toolkit also supports the visualization of output surfaces as shown in the following Section.

B. Output Surface Plots

For FIS featuring two input variables and one output variable, the R toolkit supports the generation of a visualization of the output surface using the function `gensurf` (as shown in VI.B).

In order to allow the comparison in terms of visualization between the MATLAB® and R implementations we provide the output surfaces produced for the 'tipper' example (see Appendices) in Figure 12 and Figure 13 respectively.

Note: The control surfaces are shown from a slightly different viewpoint, however the visualization capability of the R toolkit is clearly demonstrated which is the primary aim in this section.

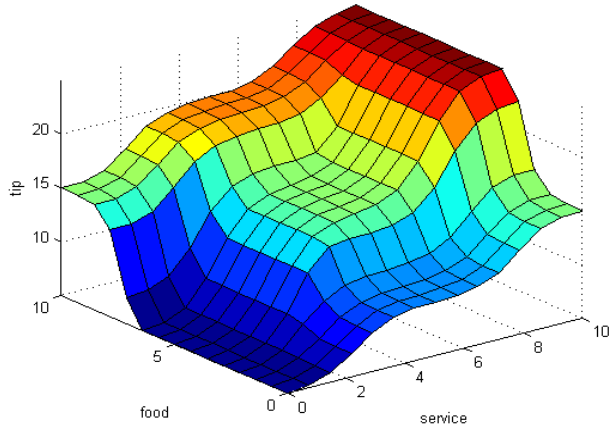

Figure 12 The control surface of the 'tipper' example produced using the MATLAB®.

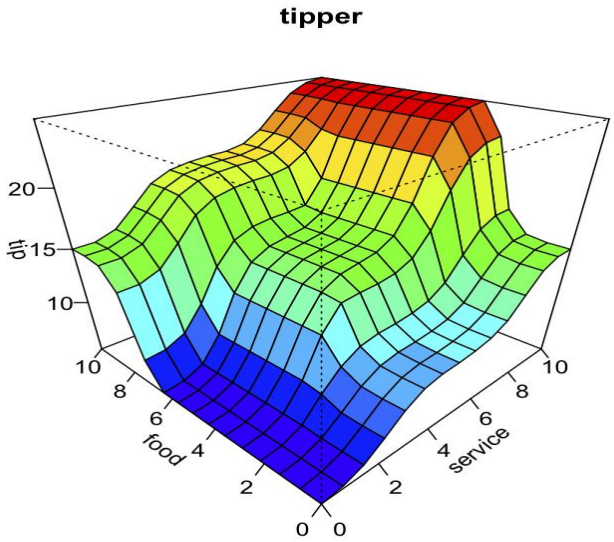

Figure 13 The similar control surface of the 'tipper' example produced using the R toolkit.

V. Conclusions and Future Work

As part of this paper we have given a brief introduction to a Fuzzy Logic toolkit based on the R language. We have provided a brief overview as well as source code samples of the currently available features within the toolkit. Specifically, we have described the functionality in terms of type-1 and interval type-2 fuzzy logic systems and provided examples of the visualization capabilities of the toolkit for type-2 fuzzy systems in general.

The current toolkit already supports other variations of fuzzy systems, including non-stationary fuzzy sets and the basics for zSlices based general type-2 fuzzy systems. We are aiming to complete these and other features and to include appropriate documentation materials before the free release of the toolkit. The open nature of the R language means that the package will be available to be freely expanded by other authors and contributors after its release and additional features such as TSK inference can easily be included.

The purpose of the paper is to get feedback and suggestions from our peers on the current approach, its functionality and applicability, and to gather information on similar efforts that may be underway in order to avoid duplication of work.

R is an ideal choice of language for such a toolbox as it is freely available and used by a wide variety of researchers from a variety of research fields: creating a tool in R will provide a generic infrastructure for a wide range of users and will hopefully lead to an increase in the uptake of fuzzy logic systems (of all flavours).

The result of this work will be an open source toolkit that can be used freely, placing the tools required to create these types of fuzzy system into the hands of all researchers interested in adopting a fuzzy approach.

Acknowledgment

We would like to thank the reviewers for providing a series of helpful and constructive comments and suggestions for the further development of the toolkit.

References

[1] L.A. Zadeh, "Fuzzy sets", Information and Control, 8 (3), 1965, pp. 338—353.

[2] MATLAB Fuzzy Logic Toolbox http://www.mathworks.com/products/fuzzylogic/ (live on 28th Feb 2011)

[3] J.R. Castro, O. Castillo, P. Melin, "An Interval Type-2 Fuzzy Logic Toolbox for Control Applications", Proceedings of the International conference of Fuzzy Systems, pp. 1-6, July 2007, London, UK.

[4] R Development Core Team; "R: A Language and Environment for Statistical Computing", *R Foundation for Statistical Computing*, Vienna, Austria, ISBN: 3-900051-07-0, URL: http://www.R-project.org, 2011.

[5] J.M. Garibaldi, M. Jaroszewski, S. Musikasuwan; "Nonstationary Fuzzy Sets", *IEEE Transactions on Fuzzy Systems*, 16(4): 1072-1086, 2008.

[6] J.M. Mendel, R.I. John and F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple," *IEEE Transactions on Fuzzy Systems*, 14: 808-821, 2006.

[7] C. Wagner and H. Hagras, "Towards general type-2 fuzzy logic systems based on zSlices," IEEE Transactions Fuzzy Systems, vol. 18, no. 4, pp. 637-660, August 2010.

[8] D. Hiebeler; "MATLAB/R Reference", URL: http://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf (live on 28th Feb 2011)

[9] MATLAB® Fuzzy Logic Tool™ 2 Users'Guide, The MathWorks, Inc., Natick, USA, March 2010.

[10] J. Mendel, Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions. Upper Saddle River, NJ, Prentice Hall, 2001.

[11] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," Inform. Sci., vol. 132, pp. 195–220, 2001.

VI. Appendices

A. *The tipper example in MATLAB®.*

The source code is based on the example provided in the tipper application example provided in the MATLAB® Fuzzy Logic Toolbox 2 users' guide.

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addvar(a,'input','food',[0 10]);
a=addvar(a,'output','tip',[0 30]);

a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);

a=addmf(a,'input',2,'rancid','trapmf',[-2 0 1 3]);
a=addmf(a,'input',2,'delicious','trapmf',[7 9 10 12]);

a=addmf(a,'output',1,'cheap','trimf',[0 5 10]);
a=addmf(a,'output',1,'average','trimf',[10 15 20]);
a=addmf(a,'output',1,'generous','trimf',[20 25 30]);

ruleList=[ ...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2 ];
a=addrule(a,ruleList);

showfis(fis)
showrule(fis)

evalfis([1 2], a)
evalfis([3 5; 2 7], a)
gensurf(a)
```

B. *The tipper example using the R toolkit.*

The complete R Fuzzy code for the same example as provided for MATLAB®.

```
fis= newfis('tipper')

fis= addvar(fis, 'input', 'service', c(0,10))
fis= addvar(fis, 'input', 'food', c(0,10))
fis= addvar(fis, 'output', 'tip', c(0,30))

fis= addmf(fis, 'input', 1, 'poor', 'gaussmf', c(1.5,0))
fis= addmf(fis, 'input', 1, 'good', 'gaussmf', c(1.5,5))
fis= addmf(fis, 'input', 1, 'excellent', 'gaussmf', c(1.5,10))
```

```
fis= addmf(fis, 'input', 2, 'rancid', 'trapmf', c(0,0,1,3))
fis= addmf(fis, 'input', 2, 'delicious', 'trapmf', c(7,9,10,10))

fis= addmf(fis, 'output', 1, 'cheap', 'trimf', c(0,5,10))
fis= addmf(fis, 'output', 1, 'average', 'trimf', c(10,15,20))
fis= addmf(fis, 'output', 1, 'generous', 'trimf', c(20,25,30))

rules= rbind(c(1,1,1,1,2), c(2,0,2,1,1), c(3,2,3,1,2))
fis= addrule(fis, rules)

showfis(fis)
showrule(fis)

evalfis(c(1,2), fis)
evalfis(rbind(c(3,5), c(2,7)), fis)
gensurf(fis)
```