Fuzzy Logic in R

By Jeff Heaton

R is a programming language designed for statistical computing. R is widely used in scientific, actuarial and data science computing. The greatest strength of the R programming language is the many third party packages contributed by R's user community. In this article, I will introduce you to fuzzy logic programming in R. This article assumes that the reader already has knowledge of fuzzy logic. If you need a review of fuzzy logic please read "***Warm and Fuzzy … And Real!"*** by Dave Snell. It is also in this issue.

There are several different R packages available for fuzzy logic programming. This article assumes that you are using the "sets" package. If "sets" is not already installed, it can be installed with the following command.

```
install.packages("sets")
```

You only need to install the "sets" package once. After installation, R programs can make use of "sets" by invoking the following command.

```
library(sets)
```

If you get an error from the above command, then "sets" is not properly installed on your system.

I will now show you how to set up a basic fuzzy system. This system will implement a very simplistic underwriting rating system. This will allow you to define linguistic variables about a potential insured. Fuzzy rules will be defined based on those linguistic variables. You will also be able to perform fuzzy inference and ultimately defuzzify to an underwriting rating.

You must first define the range and granularity of your universe. The universe used for this example will be between 0 and 40, with a granularity of 0.1. The inputs for all of your variables must fit within this range. Additionally, the granularity will specify the accuracy of the fuzzy inferences. At a more superficial level, the range also defines the x-axis of your plots. I used the following range and granularity for this example.

```
sets_options("universe", seq(from = 0,
to = 40, by = 0.1))
```

If your individual variables use vastly different ranges, it may be useful to normalize the variables to more consistent ranges.

Linguistic Variables

Linguistic variables allow the use of descriptive words such as "underweight" or "obese" to describe normally numeric variables. Underwriters use many different variables to assign a rating to a potential insured. For this example we will only consider the hemoglobin A1c (HbA1c) blood test, a hypertension class and body mass index (BMI). We will place this set of linguistic variables into a set named "variables."

```
variables <-

 set(
```

Starting with BMI—we define several linguistic values, such as "under," "fit," "over," and "obese." We define the mean for each of these in BMI. For simplicity, I assign a standard deviation of 3.0 to each. There are a variety of fuzzy membership functions available to define your variables. For BMI, I am using a normal distribution.

```
  bmi =

   fuzzy_partition(varnames =

   c(under = 9.25, fit = 21.75,

    over = 27.5, obese = 35),

    sd = 3.0),
```

For the linguistic variable "a1c" I use a conic fuzzy membership function, with a radius of five. I define linguistic values of "l" (for low), "n" (for normal) and "h" (for high).

CONTINUED ON **PAGE 36**

These are assigned to actual a1c test values.

```
a1c =
 fuzzy_partition(varnames =
 c(l = 4, n = 5.25, h = 7),
FUN = fuzzy_cone, radius = 5),
```

The linguistic variable "rating" also defines its membership with a cone. This set defines the underwriter rating for the proposed insured. Underwriter ratings can range from 10 (decline) to 1 (preferred). I define three linguistic values in this range. The linguistic variable "DC" is decline, "ST" is normal and "PF" is preferred.

```
rating =
 fuzzy_partition(varnames =
  c(DC = 10, ST = 5, PF = 1),
  FUN = fuzzy_cone, radius = 5),
```

Finally, I define linguistic variable "bp" to represent blood pressure. Here I normalize the systolic and diastolic readings to a single value. The value 0 represents normal and 30 represents severe hypertension. It does not matter, for this example, exactly how you normalize the actual systolic and diastolic values. I suggest using a table, similar to the following URL.

http://www.mayoclinic.org/diseases-conditions/high-blood-pressure/in-depth/blood-pressure/art-20050982

I use a normal distribution fuzzy membership function, with a standard deviation of 2.5. I define linguistic values of "norm" (normal), "pre" (prehypertension), "hyp" (hypertension) and "shyp" (severe hypertension).

```
  bp =
   fuzzy_partition(varnames =
   c(norm = 0, pre = 10, hyp = 20,
    shyp = 30), sd = 2.5)
)
```

Now that the linguistic variables have been defined, rules can be created.

FUZZY RULES

Fuzzy rules are used to link the linguistic variables of "bmi," "a1c," and "bp" to the linguistic variable "rating." I use three different rules for this example. You can see this rule set here.

```
rules <-
 set(
  fuzzy_rule(bmi %is% under || bmi %is%
obese || a1c %is% l,
        rating %is% DC),
  fuzzy_rule(bmi %is% over || a1c %is% n
|| bp %is% pre,
        rating %is% ST),
  fuzzy_rule(bmi %is% fit && a1c %is% n
&& bp %is% norm,
        rating %is% PF)
 )
```

The first rule states that the rating will be DC (decline) if the BMI is "under," BMI is "obese" or the a1c is "low." The double-pipe (||) represents "or," and "%is%" represents a fuzzy "is" operator. The rules are relatively readable as English sentences.

The second rule specifies that the rating will be ST (standard) if BMI is "over," a1c is "norm," or bp is "pre."

Finally, the third rule states that the rating will be PF (preferred) if BMI is "fit," a1c is "norm," and bp is "norm." Notice here that we use the "and" operator, represented by the double ampersand (&&). Of course, this is just a simple example. The above rules are not meant to define an actual underwriting system.

Now that the rules and linguistic variables have been defined, we can build a system. This is done with the following R code. You can also "print" and "plot" this system.

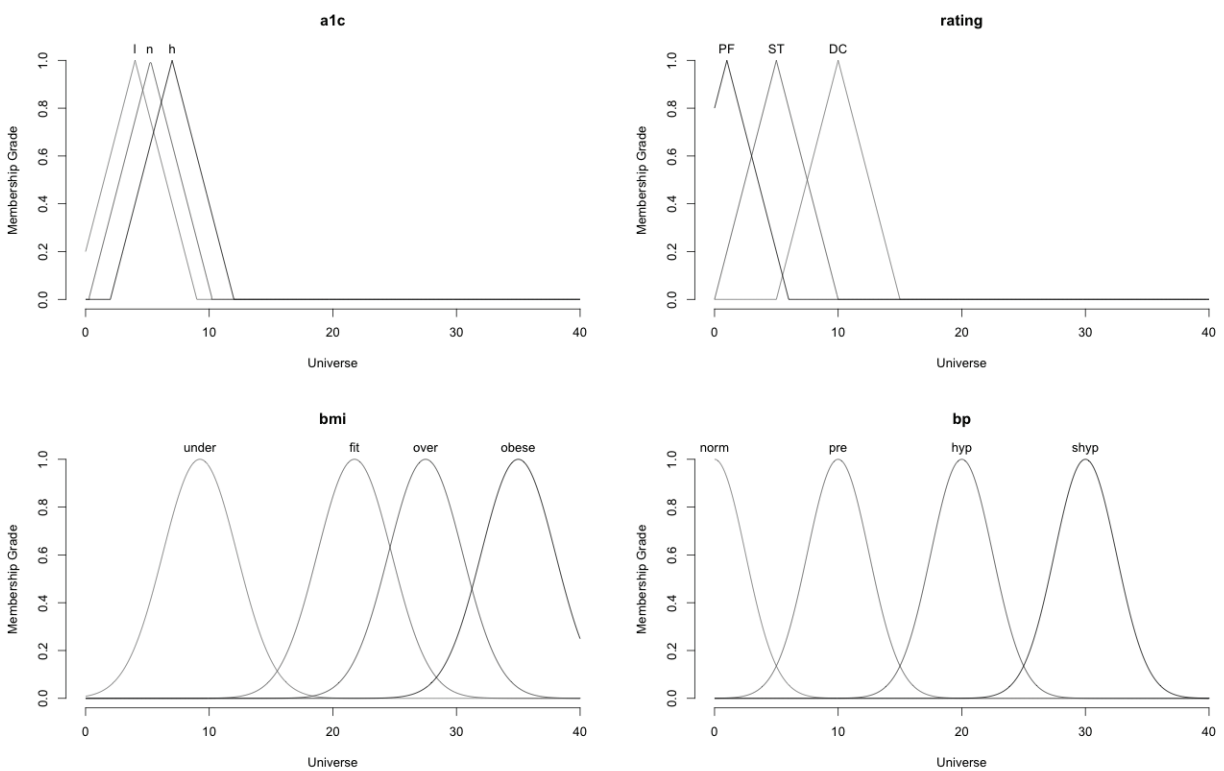
```
system <- fuzzy_system(variables, rules)

print(system)

plot(system)
```

The plot of this system can be seen in Figure 1.

Figure 1: Linguistic Variable System



CONTINUED ON **PAGE 38**

FUZZY INFERENCE AND DEFUZZIFICATION

We can now infer underwriter ratings from the above system. The process of fuzzy inference allows us to specify values for a1c, rating and bmi. This will give us a percent membership in the "rating" linguistic variable. Consider a proposed insured with a BMI of 29, a1c of five, and bp rating of 20. The following command would infer the rating into the variable "fi."

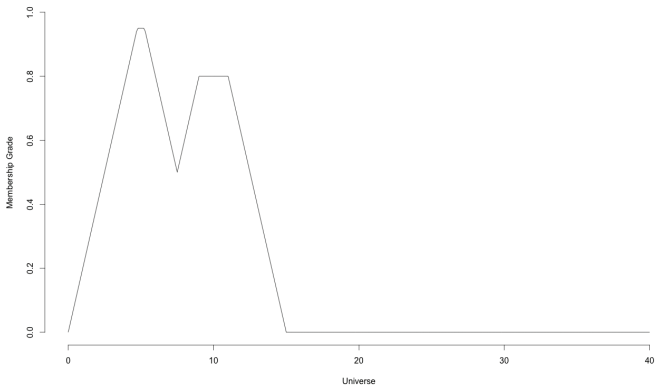
```
fi <- fuzzy_inference(system, list(bmi = 29, a1c=5, bp=20))
```

There is not a single value for "fi"; rather, it is a percent membership in each underwriter rating. The following command plots this to a chart.

```
plot(fi)
```

This chart can be seen in Figure 2. You will notice that the chart has two different membership peaks. One is near 7 and the other near 10.

Figure 2: Inferred Rating Membership



Defuzzification is the process where this membership is taken back to an actual number. There are several algorithms for defuzzification. The following command performs a defuzzification. The defuzzified rating is 7.445238.

```
gset_defuzzify(fi, "centroid")
```

Once you have completed your inferences, it is considered good practice to clear the fuzzy sets. This is done with the following command.

```
sets_options("universe", NULL)
```

CONCLUSIONS

Fuzzy logic offers many advantages over the more traditional "crisp logic" that most computer programs are composed of. Because rules are inferred, it is not necessary to create the vast number of rules that most traditional rule engines grow into. The R source code for my example can be found at this link: *http://www.soa.org/news-and-publications/newsletters/forecasting-futurism/default.aspx*. If you would like to read more about the R "sets" package, you can visit its home page at the following link: *http://cran.r-project.org/web/packages/sets/sets.pdf*

Jeff Heaton

Jeff Heaton, is EHR data scientist at RGA Reinsurance Company and author of several books on artificial intelligence. He can be reached at *jheaton@rgare.com*.