FuzzyNumbers Package for R: A Quick Tutorial

Marek Gągolewski
Systems Research Institute, Polish Academy of Sciences
ul. Newelska 6, 01-447 Warsaw, Poland
Email: gagolews@ibspan.waw.pl

June 1, 2012

Contents

1 Getting started

R is a free software environment for statistical computing and graphics, which includes an implementation of a very powerful and quite popular high-level language called S. To install R and find some information on the S language please visit R Project's Homepage at www.R-project.org. Please note that the FuzzyNumbers package requires R version ≥ 2.15.

To install the latest version of FuzzyNumbers available on CRAN we type:

```
# ... TO DO ...
```

Alternatively, we may fetch a development (perhaps unstable, but newer) version from RForge:

```
# ... TO DO ...
```

Fortunately, this is done only once.

Each session with FuzzyNumbers should be preceded by a call to:

```
require("FuzzyNumbers")  # Load the package
```

To view the main man page we type:

```
library(help = "FuzzyNumbers")
```

For more information please visit the package homepage at www.ibspan.waw.pl/ gagolews/FuzzyNumbers/. In case of any problems, comments, or suggestions feel free to contact the author. Good luck!

2 How to create instances of fuzzy numbers

2.1 Trapezoidal FNs

First let us create a trapezoidal fuzzy number T such that $\mathrm{core}(T) = [2, 4]$ and $\mathrm{supp}(T) = [1, 7]$:

```
T <- TrapezoidalFuzzyNumber(1, 2, 4, 7)
```

This object is an instance of the following R class:

```
class(T)

## [1] "TrapezoidalFuzzyNumber"
## attr(,"package")
## [1] "FuzzyNumbers"
```

We may print some basic information by calling `print(T)` or simply by typing:

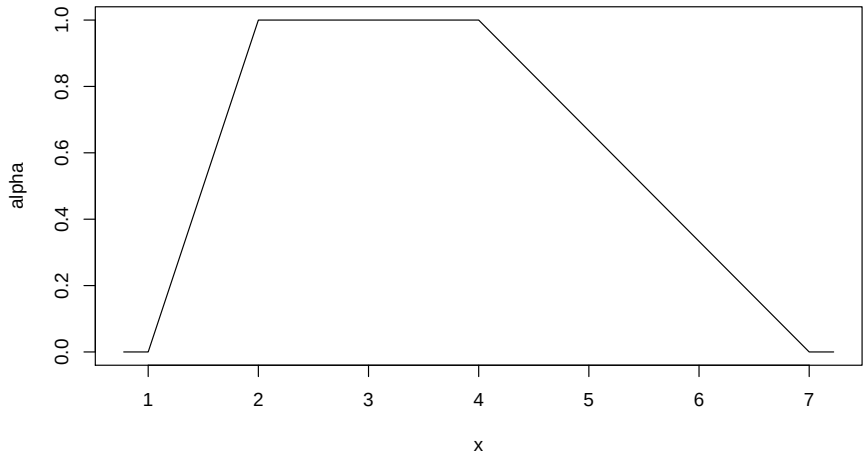
```
T

## Trapezoidal fuzzy number with support=[1,7] and core=[2,4].
```

Bingo! :-)

To depict T we call:

```
plot(T)
```



Trapezoidal fuzzy numbers are among the simplest FNs. Despite their simplicity, however, they include triangular FNs, "crisp" real intervals, and "crisp" reals. Please note that currently no separate classes for these FNs types are implemented in the package.

```
TrapezoidalFuzzyNumber(1, 2, 2, 3)  # triangular FN

## Trapezoidal fuzzy number with support=[1,3] and core=[2,2].

TrapezoidalFuzzyNumber(2, 2, 3, 3)  # crisp interval

## Trapezoidal fuzzy number with support=[2,3] and core=[2,3].

TrapezoidalFuzzyNumber(5, 5, 5, 5)  # crisp real

## Trapezoidal fuzzy number with support=[5,5] and core=[5,5].
```

2.2 Arbitrary FNs

An arbitrary fuzzy number A may be defined by specifying its core, support, and either its left/right side functions or lower/upper α-cut bounds. Please note that many algorithms assume we provide at least the latter, i.e. α-cuts.

Side functions. A fuzzy number A specified by side functions has a membership function:

$$\mu_A(x) = \begin{cases} 0 & \text{if } \quad x < \texttt{a1}, \\ \texttt{left}\left(\frac{x-\texttt{a1}}{\texttt{a2}-\texttt{a1}}\right) & \text{if } \texttt{a1} \le x < \texttt{a2}, \\ 1 & \text{if } \texttt{a2} \le x \le \texttt{a3}, \\ \texttt{right}\left(\frac{x-\texttt{a3}}{\texttt{a4}-\texttt{a3}}\right) & \text{if } \texttt{a3} < x \le \texttt{a4}, \\ 0 & \text{if } \texttt{a4} < x, \end{cases} \tag{1}$$

where $\texttt{a1}, \texttt{a2}, \texttt{a3}, \texttt{a4} \in \mathbb{R}$, $\texttt{a1} \le \texttt{a2} \le \texttt{a3} \le \texttt{a4}$, $\texttt{left} : [0,1] \to [0,1]$ is a nondecreasing function (called left side of A), and $\texttt{right} : [0,1] \to [0,1]$ is a nonincreasing function (right side of A).

Please note that by defining left/right side functions on $[0, 1]$ we really make (in author's humble opinion) the process of generating examples for our publications much easier.

An example:

```
A1 <- FuzzyNumber(1,2,4,7,
     left=function(x) x,
    right=function(x) 1-x
)
class(A1)

## [1] "FuzzyNumber"
## attr(,"package")
## [1] "FuzzyNumbers"

A1

## Fuzzy number with support=[1,7] and core=[2,4].
```

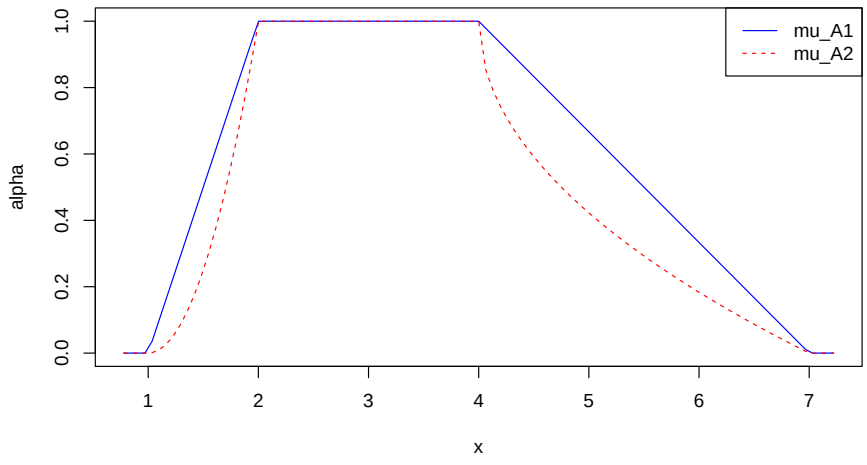
The above is (roughly — we will see why) equivalent to the trapezoidal fuzzy number T defined in Sec. 2.1. The `TrapezoidalFuzzyNumber` class inherits all the goodies from the `FuzzyNumber` class, but is more specific (guarantees faster computations, contains more detailed information, etc.). Thus, is we wanted to define a trapezoidal FN next time, we would rather not do it like that.

Another example:

```
A2 <- FuzzyNumber(1,2,4,7,
     left=function(x) x^2,
    right=function(x) 1-sqrt(x)
)
```

One picture is worth thousand words, so...

```
plot(A1, col="blue")
plot(A2, col="red", lty=2, add=TRUE)
legend("topright", c("mu_A1", "mu_A2"), col=c("blue", "red"), lty=c(1,2))
```



α-cut bounds. Alternatively, A may be defined by specifying its α-cut bounds. We have (for $\alpha \in (0,1)$ and $\texttt{a1} \le \texttt{a2} \le \texttt{a3} \le \texttt{a4}$):

$$A_\alpha = \big[\texttt{a1} + (\texttt{a2} - \texttt{a1}) \cdot \texttt{lower}(\alpha), \texttt{a3} + (\texttt{a4} - \texttt{a3}) \cdot \texttt{upper}(\alpha)\big], \tag{2}$$

where $\texttt{lower} : (0,1) \to (0,1)$ is an increasing function (lower α-cut bound), and $\texttt{upper} : (0,1) \to (0,1)$ is a decreasing function (upper bound).

It is easily seen that for $\alpha \in (0,1)$ we have the relationship with the first representation:

$$\texttt{lower}(\alpha) = \inf\{x : \texttt{left}(x) \ge \alpha\}, \tag{3}$$

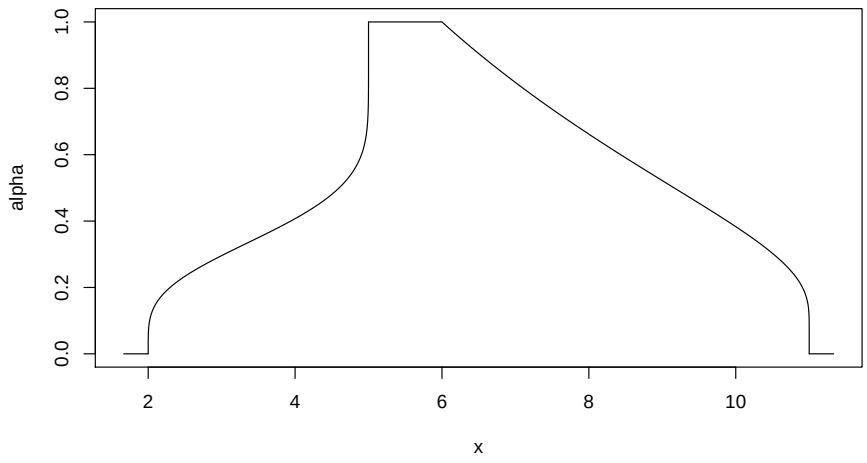
$$\texttt{upper}(\alpha) = \sup\{x : \texttt{right}(x) \ge \alpha\}. \tag{4}$$

If `left` / `right` is continuous then `lower` / `upper` (respectively) is its inverse.

An example:

```
A3 <- FuzzyNumber(2,5,6,11,
         lower=function(alpha) pbeta(alpha, 5, 9), # CDF of a beta distr.
         upper=function(alpha) pexp(1/alpha-1) # transformed CDF of an exp. distr.
);
```

```
plot(A3)
```



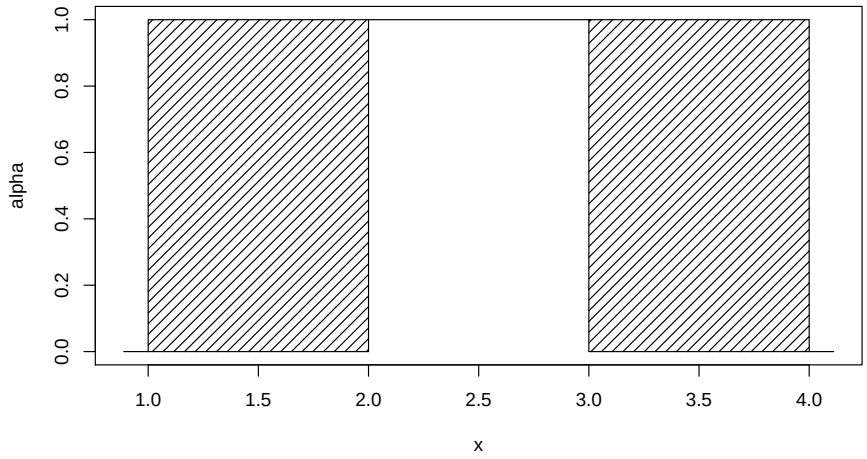
Please note that — up to now — we were passing to the `FuzzyNumber` constructor either side functions or α-cut bounds. Let us study what happens when we omit both of them.

```
A4 <- FuzzyNumber(1, 2, 3, 4)
A4

## Fuzzy number with support=[1,4] and core=[2,3].
```

The object seems to be defined correctly (`R` does not make complaints). However...

```
plot(A4)
```



It turns out that we have obtained a shadowed set! Indeed, this behavior is quite reasonable: we have provided no information on the "partial knowledge" part of our fuzzy number. If fact, the object has been initialized as:

```
A4 <- FuzzyNumber(1, 2, 3, 4, left = function(x) NA, right = function(x) NA,
    lower = function(x) NA, upper = function(x) NA)
```

that is, with functions always returning `NA` (not-available, *any* value).

Does it mean that when we define a FN by side functions, we cannot compute α-cuts? Indeed!

```
alphacut(A3, 0.5)  # A3 has side functions defined

## [1] 4.600 9.161

alphacut(A1, 0.5)  # A1 has not

## [1] NA NA
```

Why? This is because finding a function inverse numerically requires lengthy computations and is always done locally (for a given point, not for "whole" the function at once). `R` is not a symbolic mathematical solver. If we had defined such a procedure (it is really easy to do by using the `uniroot()` function) then an inexperienced user would have used it in his/her algorithms and wondered why everything runs so slow.

The general rule thus is: if you want α-cuts (e.g. for finding trapezoidal approximations of FNs) — specify them. If you wand side functions (what for? luckily, the `plot()` function automatically detects what kind of knowledge we have) — do the same.

Of course, in case of e.g. trapezoidal fuzzy numbers these functions are known *a priori* and there is no need (there is even no possibility) to provide them manually.

TO BE CONTINUED