Getting Started with Fitting Fuzzy Linear Regression Models in R

Pavel Škrabánek and Natália Martínková

June 6, 2019

Fuzzy regression provides an alternative to statistical regression when the model is indefinite, the relationships between model parameters are vague, sample size is low or when the data are hierarchically structured. The fuzzy regression is thus applicable in cases, where the data structure prevents statistical analysis.

Here, we explain the implementation of fuzzy linear regression methods in the R [7] package `fuzzyreg` [9]. The Chapter 1: Quick start guides the user through the direct steps necessary to obtain a fuzzy regression model from crisp (not fuzzy) data. Followed by the Chapter 4.2, the user will be able to infer and interpret the fuzzy regression model.

Contents

1 Quick start

To install, load the package and access help, run the following R code:

```
> install.packages("fuzzyreg", dependencies = TRUE)
> require(fuzzyreg)
> help(package = "fuzzyreg")
```

Next, load the example data and run the fuzzy linear regression using a wrapper function `fuzzylm` that simulates the established functionality of other regression functions and uses the `formula` and `data` arguments:

```
> data(fuzzydat)
> f = fuzzylm(y ~ x, data = fuzzydat$lee)
```

The result shows the coefficients of the fuzzy linear regression in form of non-symmetric triangular fuzzy numbers.

```
> print(f)

Fuzzy linear model using the PLRLS method

Call:
fuzzylm(formula = y ~ x, data = fuzzydat$lee)

Coefficients in form of non-symmetric triangular fuzzy numbers:

               center left.spread right.spread
(Intercept) 17.761911   0.7619108    2.7380892
x            2.746428   1.2464280    0.4202387
```

We can next plot the regression fit, using shading to indicate the degree of membership of the model predictions.

```
> plot(f, res = 20, col = "lightblue", main = "PLRLS")
```

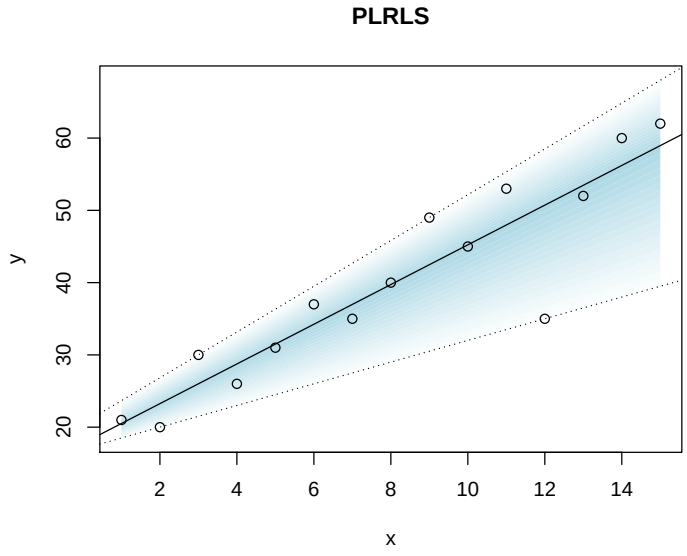

Figure 1: Fuzzy linear regression using the PLRLS method [5].

Shading visualizes the degree of membership to the triangular fuzzy number (TFN) with a gradient from lightblue to white, indicating the decrease of the degree of membership from 1 to 0. The central tendency (thick line) in combination with the left and the right spreads determine a support interval

(dotted lines) of possible values, i.e. values with non-zero membership degrees, of model predictions. The left and the right spreads determine the lower and upper boundary of the interval, respectively, where the degree of membership equals to 0. We can display the model with the `summary` function.

```
> summary(f)

Central tendency of the fuzzy regression model:
y = 17.7619 + 2.7464 * x

Lower boundary of the model support interval:
y = 17 + 1.5 * x

Upper boundary of the model support interval:
y = 20.5 + 3.1666 * x

The total error of fit: 126248409
The mean squared distance between response and prediction: 262.1
```

2 Triangular fuzzy numbers

The package `FuzzyNumbers` [2] provides an excellent introduction into fuzzy numbers and offers a great flexibility in designing the fuzzy numbers. Here, we implement a special case of fuzzy numbers, the triangular fuzzy numbers.

A fuzzy real number $\tilde{A}$ is a fuzzy set defined on the set of real numbers. Each real value number x belongs to the fuzzy set $\tilde{A}$, with a degree of membership that can range from 0 to 1. The degrees of membership of x are defined by the membership function $\mu_{\tilde{A}}(x) : x \to [0, 1]$, where $\mu_{\tilde{A}}(x^*) = 0$ means that the value of x^* is not included in the fuzzy number $\tilde{A}$ while $\mu_{\tilde{A}}(x^*) = 1$ means that x^* is positively comprehended in $\tilde{A}$ (Figure 2).

In `FuzzyNumbers`, the fuzzy number is defined using side functions. In `fuzzyreg`, we simplify the input of the TFNs as a vector of length 3. The first element of the vector specifies the central value x_c, where the degree of membership is equal to 1, $\mu(x_c) = 1$. The second element is the left spread, which is the distance from the central value to a value x_l where $\mu(x_l) = 0$ and $x_l < x_c$. The left spread is thus equal to $x_c - x_l$. The third element of the TFN is the right spread, i.e. the distance from the central value to a value x_r where $\mu(x_r) = 0$ and $x_r > x_c$. The right spread is equal to $x_r - x_c$. The central value x_c of the TFN is its core, and the interval (x_l, x_r) is the support of the TFN.

The crisp number $a = 0.5$ can be written as a TFN A with spreads equal to 0 (Figure 2):

```
> A = c(0.5, 0, 0)
```

The non-symmetric TFN B and the symmetric TFN C are then:

```
> B = c(1.5, 0.8, 0.4)
> C = c(2.5, 0.5, 0.5)
```

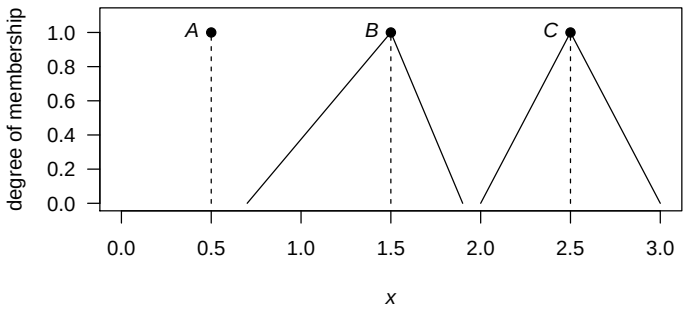

Figure 2: Triangular fuzzy numbers.

2.1 Converting crisp numbers to TFNs

When the collected data do not contain spreads, we can directly apply the PLRLS method to model the relationship between the variables in the fuzzy set framework (Chapter 1: Quick start). For other methods, the spreads must be imputed.

- The TFN might have spreads equal to 0 as shown for A in Figure 2.
- Small random spreads might be generated, e.g. using `abs(runif(n) * 1e-6)`, where `n` is the number of the observations.
- Spreads might represent a known measurement error of the device used to collect the data.
- Spreads might be calculated as a statistic from the data.

Note that the spreads must always be equal to or greater than zero.

2.2 Converting TFN from class `FuzzyNumber`

The conversion from an object of the class `FuzzyNumber` to TFN used in `fuzzyreg` requires adjusting the core and the support values of the `FuzzyNumber` object to the central value and the spreads. For example, let's define a trapezoidal fuzzy number B_1 that is identical with TFN B displayed in Figure 2, but that is an object of class `FuzzyNumber`.

```
> require(FuzzyNumbers)
> B1 = FuzzyNumber(0.7, 1.5, 1.5, 1.9,
+                  left = function(x) x,
+                  right = function(x) 1 - x,
+                  lower = function(a) a,
+                  upper = function(a) 1 - a)
> B1
```

```
Fuzzy number with:
   support=[0.7,1.9],
      core=[1.5,1.5].
```

The core of B_1 will be equal to the central value of the TFN if the `FuzzyNumber` object is a TFN. However, the `FuzzyNumbers` package considers TFNs as a special case of trapezoidal fuzzy numbers. The core of B_1 thus represents the interval, where $\mu_{B_1}(x^*) = 1$, and the support is the interval, where $\mu_{B_1}(x^*) > 0$. We can use these values to construct the TFN B.

```
> xc = core(B1)[1]
> l = xc - supp(B1)[1]
> r = supp(B1)[2] - xc
> c(xc, l, r)

[1] 1.5 0.8 0.4
```

When the trapezoidal fuzzy number has the core wider than one point, we need to approximate a TFN. The simplest method calculates the mean of the core as `mean(core(B1))`.

We can also defuzzify the fuzzy number and approximate the central value with the `expectedValue()` function. However, the expected value is a midpoint of the expected interval of a fuzzy number derived from integrating the side functions. The expected values will not have the degree of membership equal to 1 for non-symmetric fuzzy numbers. Constructing the mean of the core might be a more appropriate method to obtain the central value of the TFN for most applications.

Fuzzy numbers with non-linear side functions may have large support intervals, for which the above conversion algorithm might skew the TFN. The function `trapezoidalApproximation()` can first provide a suitable approximation of the fuzzy number with non-linear side functions, for which the core and the support values will suitably reflect the central value and the spreads used in `fuzzyreg`.

3 Methods for fitting fuzzy regression models

Methods implemented in `fuzzyreg 0.4` fit fuzzy linear models (Table 1).

Table 1: Methods for fitting fuzzy regression with `fuzzyreg`. m - number of allowed independent variables x; x, y, $\hat{y}$ - type of expected number for independent, dependent variables and predictions; s - symmetric; ns - non-symmetric.

Method	m	x	y	$\hat{y}$	Reference
PLRLS	∞	crisp	crips	nsTFN	[5]
PLR	∞	crisp	sTFN	sTFN	[8]
OPLR	∞	crisp	sTFN	sTFN	[3]
FLS	1	crisp	nsTFN	nsTFN	[1]
MOFLR	∞	sTFN	sTFN	sTFN	[6]

Methods that require symmetric TFNs handle input specifying one spread, but in methods expecting non-symmetric TFN input, both spreads must be defined even in cases when the data contain symmetric TFNs.

A possibilistic linear regression (PLR) is a paradigm of whole family of possibilistic-based fuzzy estimators. It was proposed for crisp observations of the explanatory variables and symmetric fuzzy observations of the response variable. `fuzzyreg` uses the min problem implementation [8] that estimates the regression coefficients in such a way that the spreads for the model response are minimal needed to include all observed data. Consequently, the outliers in the data will increase spreads in the estimated coefficients.

The possibilistic linear regression combined with the least squares (PLRLS) method [5] fits the model prediction spreads and the central tendency with the possibilistic and the least squares approach, respectively. The input data represent crisp numbers and the model predicts the response in form of a non-symmetric TFN. Local outliers in the data strongly influence the spreads, so a good practice is to remove them prior to the analysis.

The method by Hung and Yang [3] expands PLR [8] by adding an omission approach for detecting outliers (OPLR). We implemented a version that identifies a single outlier in the data located outside of the Tukey's fences. The input data include crisp explanatory variables and the response variable in form of a symmetric TFN.

Fuzzy least squares (FLS) method [1] supports a simple FLR for a non-symmetric TFN explanatory as well as a response variable. This probabilistic-based method (FLS calculates the fuzzy regression coefficients using least squares) is relatively robust against outliers compared to the possibilistic-based methods.

A multi-objective fuzzy linear regression (MOFLR) method estimates the fuzzy regression coefficients with a possibilistic approach from symmetric TFN input data [6]. Given a specific weight, the method determines a trade-off between outlier penalization and data fitting that enables the user to fine-tune outlier handling in the analysis.

4 Running a fuzzy linear regression

The TFN definition used in `fuzzyreg` enables an easy setup of the regression model using the well-established syntax for regression analyses in `R`. The model is set up from a `data.frame` that contains all observations for the dependent variable and the independent variables. The `data.frame` must contain columns with the respective spreads for all variables that are TFNs.

The example data from Nasrabadi et al. [6] contain symmetric TFNs. The spreads for the independent variable x are in the column `xl` and all values are equal to 0. The spreads for the dependent variable y are in the column `yl`.

```
> fuzzydat$nas
```

```
  x xl    y  yl
1 1  0  6.4 2.2
2 2  0  8.0 1.8
3 3  0 16.5 2.6
4 4  0 11.5 2.6
5 5  0 13.0 2.4
```

Note that the data contain only one column with spreads per variable. This is an accepted format for symmetric TFNs, because the values can be recycled for the left and right spreads.

The `formula` argument used to invoke a fuzzy regression with the `fuzzylm()` function will relate $y \sim x$. The columns `x` and `y` contain the central values of the variables. The spreads are not included in the `formula`. To calculate the fuzzy regression from TFNs, list the column names with the spreads as a character vector in the respective arguments of the `fuzzylm` function.

```
> f2 = fuzzylm(formula = y ~ x, data = fuzzydat$nas,
+              fuzzy.left.x = "xl",
+              fuzzy.left.y = "yl", method = "moflr")
```

Calls to methods that analyse non-symmetric TFNs must include both arguments for the left and right spreads, respectively. The arguments specifying spreads for the dependent variable are `fuzzy.left.y` and `fuzzy.right.y`. However, if we wish to analyse symmetric TFNs using a method for non-symmetric TFNs, both argumets might call the same column with the values for the spreads.

```
> f3 = fuzzylm(y ~ x, data = fuzzydat$nas,
+              fuzzy.left.y = "yl",
+              fuzzy.right.y = "yl", method = "fls")
```

As the spreads are included in the model using the column names, the function cannot check whether the provided information is correct. The issue gains importance when developing a multiple fuzzy regression model. The user must ascertain that the order of the column names for the spreads corresponds to the order of the variables in the `formula` argument.

The fuzzy regression models can be used to predict new data within the range of data used to infer the model with the `predict` function. The reason for disabling extrapolations from the fuzzy regression models lies in the non-negligible risk that the support boundaries for the TFN might intersect the central tendency. The predicted TFNs outside of the range of data might not be defined. The predicted values will replace the original variable values in the `fuzzylm` data structure.

```
> pred2 = predict(f2)
> pred2$y
```

```
     central value left spread right spread
[1,]      7.739997        8.32         8.32
[2,]      9.409999        8.80         8.80
[3,]     11.080000        9.28         9.28
[4,]     12.750002        9.76         9.76
[5,]     14.420003       10.24        10.24
```

4.1 Comparing fuzzy regression models

The choice of the method to estimate the parameters of the fuzzy regression model is data-driven (Table 1). Following the application of the suitable methods, fuzzy regression models can be compared according to the sum of differences

between membership values of the observed and predicted membership functions [4].

In Figure 3, the points represent central values of the observations and whiskers indicate their spreads. The shaded area shows the model predictions with the degree of membership greater than zero. We can compare the models numerically using the total error of fit $\sum E$ [4] with the `TEF()` function:

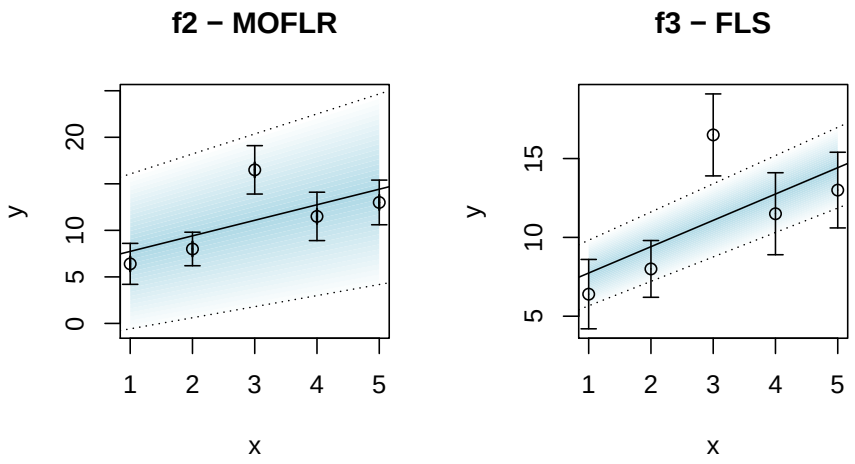

Figure 3: Comparison of fuzzy regression models fitted with the MOFLR and FLS methods. The FLS method has lower total error of fit $\sum E$ and thus better reflects the observations.

```
> TEF(f2)

[1] 16.18616

> TEF(f3)

[1] 6.066592
```

Lower values of $\sum E$ mean that the predicted TFNs fit better with the observed TFNs.

4.2 Interpreting the fuzzy regression model

When comparing the fuzzy linear regression and a statistical linear regression models, we can observe that while the fuzzy linear regression shows something akin to a confidence interval, the interval differs from the confidence intervals derived from a statistical linear regression model (Figure 4).

The confidence interval from a statistical regression model shows the certainty that the modeled relationship fits within. We are 95% certain that the true relationship between the variables is as displayed.

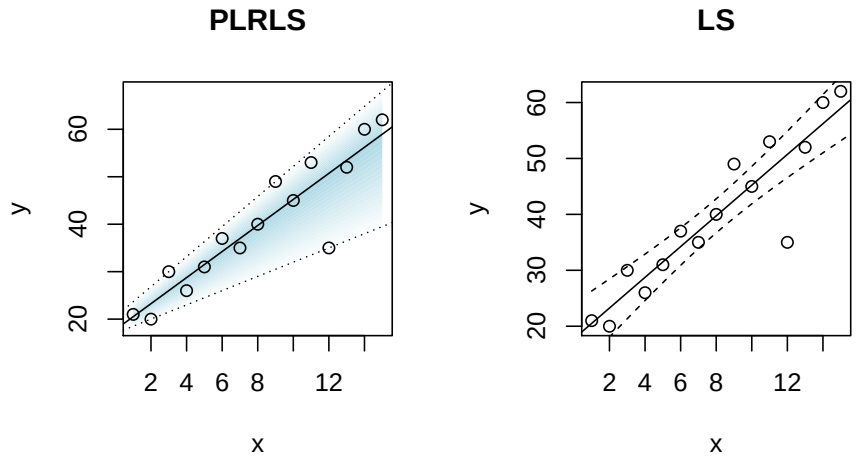

Figure 4: Fuzzy and statistical linear regression. Fuzzy regression displays the central value and the support boundaries determined by the left and right spread (PLRLS - left panel). Statistical least-squares regression shows the 95% confidence interval for the regression fit (LS - right panel).

On the other hand, the support of the fuzzy regression model prediction shows the range of possible values. The dependent variable can reach any value from the set, but the values more distant from the central tendency will have smaller degree of membership. We can imagine the values closer to the boundaries as vaguely disappearing from the set as if they bleached out (gradient towards white in the fuzzy regression model plots).

5 How to cite

To cite `fuzzyreg`, include the reference to the software and the used method.

> Škrabánek P. and Martínková N. 2018. fuzzyreg: An R Package for Fuzzy Linear Regression. In: Čech P., Svozil D. (eds.), *ENBIK2018 Conference Proceedings, Prague*, 7.

The references for the specific methods are accessible through the method help, e.g. the default fuzzy linear regression method PLRLS:

```
> ?plrls
```

References

[1] P. Diamond. Fuzzy least squares. *Information Sciences*, 46(3):141–157, 1988.

[2] M. Gagolewski and J. Caha. *FuzzyNumbers Package: Tools to Deal with Fuzzy Numbers in R*, 2018.

[3] W.-L. Hung and M.-S. Yang. An omission approach for detecting outliers in fuzzy regression models. *Fuzzy Sets and Systems*, 157:3109–3122, 2006.

[4] B. Kim and R. R. Bishu. Evaluation of fuzzy linear regression models by comparing membership functions. *Fuzzy Sets and Systems*, 100:343–352, 1998.

[5] H. Lee and H. Tanaka. Fuzzy approximations with non-symmetric fuzzy parameters in fuzzy regression analysis. *Journal of Operations Research*, 42:98–112, 1999.

[6] M. M. Nasrabadi, E. Nasrabadi, and A. R. Nasrabady. Fuzzy linear regression analysis: a multi-objective programming approach. *Applied Mathematics and Computation*, 163:245–251, 2005.

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

[8] H. Tanaka, I. Hayashi, and J. Watada. Possibilistic linear regression analysis for fuzzy data. *European Journal of Operational Research*, 40:389–396, 1989.

[9] P. Škrabánek and N. Martínková. *fuzzyreg: Fuzzy Linear Regression*, 2018. R package version 0.3.