lfl: An R Package for Linguistic Fuzzy Logic

Michal Burda, Martin Štěpnička*

Institute for Research and Applications of Fuzzy Modeling, University of Ostrava, CE IT4Innovations, 30. dubna 22, 701 03 Ostrava, Czech Republic

Abstract

This paper presents an R package that supports the use of fuzzy relational calculus and linguistic fuzzy logic in data processing applications. The **lfl** package enables computing compositions of fuzzy relations enhanced with distinct extensions, such as excluding features, unavoidable features, or generalized quantifiers. Furthermore, it provides tools for transformation of data into fuzzy sets representing linguistic expressions, mining of linguistic fuzzy association rules, and performing an inference on fuzzy rule bases using the Perception-based logical deduction (PbLD). The package also enables to use the Fuzzy rule-based ensemble, a tool for time series forecasting based on an ensemble of forecasts from several individual methods implemented in R. To the best of the authors' knowledge, there is no other open source software that would provide free tools covering the above-described fragments of the fuzzy modeling area. Therefore, we find highly desirable to allow the community to get familiar with the tools as well as their implementation.

Keywords: fuzzy sets, fuzzy natural logic, linguistic fuzzy logic, association rules, compositions of fuzzy relations, R

*Corresponding author. Tel.: +420 59 709 1410; fax: +420 596 120 478.
Email addresses: Michal.Burda@osu.cz (Michal Burda), Martin.Stepnicka@osu.cz (Martin Štěpnička)

1. Introduction

The aim of this paper is to present particular package for the R statistical environment [1, 2] named **lfl** that enables the use of the linguistic fuzzy logic in data processing applications. The package provides implemented tools for using the results of original work of [3, 4, 5, 6], and others, and it provides executable routines that are not freely available anywhere else.

Indeed, there already exist several packages for R that are focused on vagueness and fuzziness. For instance, the **sets** package [7] introduces many basic operations on fuzzy sets, the **FuzzyNumbers** package [8] provides classes and methods to deal with fuzzy numbers, the **SAFD** package [9] contains tools for elementary statistics on fuzzy data, and the **fclust** [10, 11] brings the fuzzy K-Means clustering technique to the environment of the R system. For an exhaustive study on existing software implementations of fuzzy methods we refer to the recent survey by [12].

The **lfl** package described in this paper focuses on creation of systems based on fuzzy logic and their usage in classification and prediction. A similar task is performed also by the **fugeR** package [13] that introduces an evolutionary algorithm for a construction of a fuzzy system from a training data set, or by the **frbs** package [14] that provides many widely accepted approaches for building the fuzzy systems, based on space partition, neural networks, clustering, gradient descent, or genetic algorithms. However, the tools implemented in **lfl** cover, in our opinion, areas and theories not covered by any other existing software or programming package.

The algorithms provided by the **lfl** package are tightly connected with the notion of the *fuzzy natural logic* (FNL), formerly also called the *linguistic fuzzy logic* (LFL), that was initially developed in [3]. Moreover, it covers some other closely related areas, for example fuzzy relational calculus [15, 16, 17] that includes the latest generalizations [18, 19], the algebraic structures for partial fuzzy logics [20, 21], and the connection of both topics [22].

Evaluative linguistic expression – a central notion of the fuzzy natural logic

– is the expression of the form

⟨linguistic hedge⟩⟨atomic expression⟩

that vaguely evaluates a position on the real line, for example, *"very small"*, *"roughly medium"*, or *"extremely big"*. The atomic expression takes values usually from the triplet *"small"*, *"medium"*, and *"big"* and its vague information can be adjusted by the used linguistic hedge (such as *"very"*, *"extremely"*, *"roughly"* or *"more or less"*). The particular *fuzzy sets* that model the semantics of the evaluative linguistic expressions including the justification can be found in [3], see also Figure 1. A mathematical framework for manipulation and reasoning with such linguistic expressions is provided in a specific inference method called *Perception-based Logical Deduction* (PbLD), which was tailored to the above-mentioned expressions, see [23, 4, 24].

Unlike the traditional Mamdani-Assilan approach [25] that build the rule base as a disjunction of conjunctions of antecedents and consequents, the PbLD approach is closer to the implicative approach [26, 27] since it employs genuine residuated implications to connect antecedents and consequents. However, it does not aggregate them conjunctively and it considers the rule base as a list of fuzzy rules from which only single or a very few are fired. The function choosing the particular rules to be fired is called *perception* and it takes into account the *specificity* of the antecedents of the rules. For instance, the antecedent *"age is very small"* is more specific than the antecedent *"age is small"*, see the inclusion of the respective fuzzy sets in Figure 1. In PbLD, rules with more specific antecedents take the precedence over the rules with more general antecedents, assuming that both of them fire in the same degree. That enables, e.g., to employ big discontinuous jumps in the control actions according to the needs of the particular application. We refer to [4, 5] for all the details on PbLD. It is important to note that PbLd is an inference procedure that is implemented in **lfl** however, not the only one that may be modelled in this package. As the **lfl** contains a rich choice of residuated algebraic structures as well as distinct ways of partitioning the universes, one may easily construct, for example, the

above-mentioned Mamdani-Assilian or implicative models.

The **lfl** package also provides functions for searching for *fuzzy association rules* [28]. Together with PbLD, they can be used as a machine learning tool for classification or regression problems, see [29]. The package also includes the *Fuzzy Rule-based Ensemble* (FRBE), a tool for time series forecasting [6], which is built on top of the fuzzy association rules search algorithm and PbLD.

Alternatively to machine learning, classification tasks may be solved based on human expert knowledge by using the technique of *compositions of fuzzy relations*, see [19, 18, 30]. Such an approach is especially useful when lacking a large amount of data needed for automated predictor construction.

It is important to note that **lfl** package was firstly released in 2015 and described in [31]. The differences are, however, essential. Indeed, many of the above-mentioned results that are implemented in the current version of **lfl** were not even published and thus, could hardly be incorporated in the original version. In particular, e.g., algebras for partial fuzzy logics including the novel structures such as Dragonfly algebras or lower estimation algebras; fuzzy relational calculus including the extensions such as the use of generalized quantifiers, excluding features, or unavoidable features. Furthermore, apart from these latest results, it newly includes even fundamental foundations such as the basic fuzzy relational compositions that allow to employ the extensions and also to deal with fuzzy inputs when incorporating fuzzy rule-based systems, and regarding the latter notion, the standard fuzzy relational models (Mamdani-Assilian and the implicative one) have been added as well including the related defuzzification techniques. This changes the original nature of **lfl** package from an R-package implementation of the linguistic control software LFLC, see [32] with a few extensions (associations rules and FRBE), to a brand new complex package that can be used for distinct purposes and for building more complex tools due to the presence of sound mathematical foundations of fuzzy modeling techniques.

1.1. Overview of the paper

The aim of the paper is to provide readers with a concise description of the **lfl** package not only from the implementation point of view, but also from the point of view of the theoretical tools that are at disposal. The description of the functions of the package are accompanied with examples and theoretical foundations. The paper is organized as follows. Section 2 presents basic algebraic operations for fuzzy sets and fuzzy logic including extensions for missing values. Section 3 describes compositions of fuzzy relations, a framework for classification based on expert knowledge. Section 4 introduces the concept of evaluative linguistic expressions, a mathematical model of vague linguistic notions, which allows to consider the numeric information in terms that are very close to human language. Section 5 discusses an application of evaluative linguistic expressions, the fuzzy association rules mining algorithm provided by **lfl** that extracts potentially useful and interesting knowledge from data and presents it in the form of fuzzy if/then rules. Perception-based logical deduction is an inference mechanism tightly connected with evaluative linguistic expressions too; it is introduced in Section 6. Section 8 concludes the paper.

1.2. How to obtain the *lfl* package

To obtain the **lfl** package, a working instance of the R statistical environment has to be installed first and then the

```
install.packages("lfl")
```

command automatically downloads the latest stable version of the **lfl** package from CRAN[1] together with all its dependencies, compiles, and installs it. The **lfl** package works on all platforms supported by the R software including Microsoft Windows, GNU/Linux, and MacOS. Alternatively, the development version may be installed directly from GitHub by issuing following commands within the R session:

[1] CRAN is the Comprehensive R Archive Network, a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R.

```
install.packages("devtools")
devtools::install_github("beerda/lfl")
```

After the installation is successful, the following command causes loading of the package into the working space so that the user can start using it:

```
library("lfl")
```

The **lfl** package is distributed under the terms of the GNU General Public License (GPL), which guarantees the user the freedom to use it, study, share and modify.

2. Fuzzy logic and fuzzy sets

We assume the readers are familiar with the fundamental definitions of fuzzy sets, operations on fuzzy sets, and the algebraic background. So, we only briefly recall the environment on which we work and fix the denotation for the rest of the paper.

We consider a *fuzzy set* A on a non-empty universe U (denoted by $A \in \mathcal{F}(U)$) as a mapping $A : U \to [0, 1]$, $A(u)$ is called *membership degree of u in A*. A *cardinality* $|A|$ of a fuzzy set A on a finite universe U can be defined as the sum of the membership degrees [33]:

$$|A| = \sum_{\forall u \in U} A(u).$$

The algebra of operations on fuzzy sets forms a residuated lattice structure $\langle [0, 1], \wedge, \vee, \otimes, \Rightarrow, 0, 1 \rangle$ that is also the algebraic structure of truth-values of the respective fuzzy logic. Note that the structure has two conjunctions, the *strong* conjunction $\otimes$ and the *weak* conjunction $\wedge$.

The strong conjunction $\otimes$ is a left-continuous triangular norm (t-norm) which allows to derive a dual concept – a triangular conorm (t-conorm) $\oplus$ that serves as the strong disjunction in the lattice. Analogously to the case of classical logic and classical set theory, also here we derive the *intersection* and the

union of fuzzy sets from the above introduced logical operations. Let A, B be fuzzy sets on U. Then

$$(A \cap B)(u) = A(u) \otimes B(u) \ , \quad u \in U \ ,$$
$$(A \cup B)(u) = A(u) \oplus B(u) \ , \quad u \in U \ .$$

The left-continuity of $\otimes$ is assumed in order to meet the *adjunction property*:

$$\gamma \otimes \alpha \leq \beta \quad \text{if and only if} \quad \gamma \leq \alpha \Rightarrow \beta$$

by the *residuated implication* [34] (abbr. *residuum*) $\Rightarrow$ which enables capturing the multiple-valued modus ponens property. Furthermore, we may define additional logical connectives, for instance, the *residual negation* $\neg$, and *biresiduum* $\Leftrightarrow$ that models the multiple-valued equivalence:

$$\neg\alpha = \alpha \Rightarrow 0 \ , \quad \alpha \Leftrightarrow \beta = (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \ .$$

For the particular Łukasiewicz t-norm, the residual negation $\neg$ leads to the *involutive* negation $\neg\alpha = 1 - \alpha$ that obeys the law of double negation $\neg\neg\alpha = \alpha$. Let us denote the involutive negation by $\sim$ and recall the duality between a t-norm and a t-conorm:

$$\alpha \oplus \beta = \sim(\sim\alpha \otimes \sim\beta) \ .$$

The duality makes $\sim$ an important unary connective not only for the Łukasiewicz algebra but for all residuated lattices and we may freely extend such structures by the involutive negation for further use: $\langle [0,1], \wedge, \vee, \otimes, \Rightarrow, \sim 0, 1\rangle$. It does not mean that $\neg$ is not at disposal, it is always present via the definition recalled above and we have in general two negations that, in the case of the Łukasiewicz algebra, coincide. The fact that the weak conjunction $\wedge$ and the weak disjunction $\vee$ are the lattice operations *meet* (*infimum*) $\wedge$ and the *join* (*supremum*) $\vee$ needs no further explanation.

2.1. Gödel algebra

Based on the selected t-norm, the **lfl** package provides all the derived operations in a concise and extendable way. By calling the `algebra()` function

with the name of the underlying t-norm as an argument, an instance of the S3 `algebra` class is obtained, which is a named list of functions. The user may select from `"goedel"`, `"goguen"`, or `"lukasiewicz"` variant calling the respective Gödel, Goguen (also often called *product*), or the already above-mentioned Łukasiewicz residuated lattices of operations.

The instances of the `algebra` class serve often as a parameter to many other functions of the **lfl** package. User may extend these objects by selecting from some predefined missing value handling schemes (see Section 2.4) or by defining a custom algebra instances by themselves.

For example, the algebra based on the *Gödel* t-norm, that is the standard minimum, $\otimes = \wedge$, is obtained as follows:

```
a <- algebra("goedel")
```

The `algebra()` function returns a named list of the following functions:

- `n`: (strict) negation defined as:

$$\neg\alpha = \begin{cases} 1, & \text{if } \alpha = 0 \ , \\ 0, & \text{otherwise} \ ; \end{cases}$$

- `ni`: involutive negation defined as: $\sim\alpha = 1 - \alpha$;
- `t`, `pt`: vectorized and element-wise t-norm defined as: $\alpha \otimes \beta = \min\{\alpha, \beta\}$;
- `c`, `pc`: vectorized and element-wise t-conorm defined as: $\alpha \oplus \beta = \max\{\alpha, \beta\}$;
- `r`: residuum defined as:

$$\alpha \Rightarrow \beta = \begin{cases} 1, & \text{if } \alpha \leq \beta \ , \\ \beta, & \text{otherwise} \ ; \end{cases}$$

- `b`: biresiduum;
- `i`, `pi`: vectorized and element-wise infimum defined as: $\alpha \wedge \beta = \min\{\alpha, \beta\}$;
- `s`, `ps`: vectorized and element-wise supremum defined as: $\alpha \vee \beta = \max\{\alpha, \beta\}$.

Functions `n` and `ni` accept a vector of numeric values as a single input and return a vector of negated values. Two-argument functions `r` and `b` compute the desired operation element-wisely so that both input vectors should be of equal size and return a vector of results of the same size. Similarly, `pt`, `pc`, `pi`, and `ps` work element-wisely: they accept a vector of multiple arguments and compute the outputs of the desired operation on first elements of the input vectors, then on second elements, etc. until the end of the vectors is reached, which yields a vector of the resulting values. The vectorized variants of these functions, i.e., `t`, `c`, `i`, and `s` first concatenate all the input vector arguments into a single vector and then calculate a single resulting value from it by applying the operation recursively on all elements. See the example below for more information.

```
a$n(c(0.5, 0.8, 0, 1))
## [1] 0 0 1 0
a$ni(c(0.5, 0.8, 0, 1))
## [1] 0.5 0.2 1.0 0.0
a$t(c(0.8, 0.3), c(0.2, 1), c(1, 0))
## [1] 0
a$pt(c(0.8, 0.3), c(0.2, 1), c(1, 0))
## [1] 0.2 0.0
a$r(c(0.8, 0.3), c(0.2, 1))
## [1] 0.2 1.0
```

Note that as the strong and weak conjunction coincide in the Gödel algebra as well as the strong and weak disjunction coincide, also the following holds for the functions in the **lfl** R-package: `t` = `i`, `pt` = `pi`, `c` = `s`, and `pc` = `ps`.

2.2. Goguen algebra

Goguen algebra is also often called the *product algebra* to emphasize that its central point – the strong conjunction – is nothing else but the standard product (multiplication) operation. Therefore, $\otimes = \cdot$ is also often called the product t-norm. Goguen algebra is obtained in **lfl** as follows:

```
a <- algebra("goguen")
```

The resulting list `a` contains the following functions:

- `n`: (strict) negation defined as:

$$\neg\alpha = \begin{cases} 1, & \text{if } \alpha = 0 \ , \\ 0, & \text{otherwise} \ ; \end{cases}$$

- `ni`: involutive negation defined as: $\sim\alpha = 1 - \alpha$;
- `t`, `pt`: vectorized and element-wise t-norm defined as: $\alpha \otimes \beta = \alpha\beta$;
- `c`, `pc`: vectorized and element-wise t-conorm defined as: $\alpha \oplus \beta = \alpha + \beta - \alpha\beta$;
- `r`: residuum defined as:

$$\alpha \Rightarrow \beta = \begin{cases} 1, & \text{if } \alpha \leq \beta \ , \\ \frac{\beta}{\alpha}, & \text{otherwise} \ ; \end{cases}$$

- `b`: biresiduum;
- `i`, `pi`: vectorized and element-wise infimum defined as: $\alpha \wedge \beta = \min\{\alpha, \beta\}$;
- `s`, `ps`: vectorized and element-wise supremum defined as: $\alpha \vee \beta = \max\{\alpha, \beta\}$.

Arguments of these functions follow the same usage pattern as for Gödel algebra described in Section 2.1.

2.3. Łukasiewicz algebra

The last implemented algebra is the *Łukasiewicz algebra* that stems from the seminal work on 3-valued logic by Polish logician Jan Łukasiewicz [35]. Note, that Łukasiewicz algebra forms so-called MV algebra [36] that is the best generalization of the classical Boolean algebra. The implementation is provided as follows:

```
a <- algebra("lukasiewicz")
```

The particular functions are defined as follows:

- `n`, `ni`: both negations are equally defined as: $\neg\alpha = {\sim}\alpha = 1 - \alpha$;
- `t`, `pt`: vectorized and element-wise t-norm defined as: $\alpha \otimes \beta = \max\{0, \alpha + \beta - 1\}$;
- `c`, `pc`: vectorized and element-wise t-conorm defined as: $\alpha \oplus \beta = \min\{1, \alpha + \beta\}$;
- `r`: residuum defined as:

$$\alpha \Rightarrow \beta = \begin{cases} 1, & \text{if } \alpha \leq \beta, \\ 1 - \alpha + \beta, & \text{otherwise;} \end{cases}$$

- `b`: biresiduum;
- `i`, `pi`: vectorized and element-wise infimum defined as: $\alpha \wedge \beta = \min\{\alpha, \beta\}$;
- `s`, `ps`: vectorized and element-wise supremum defined as: $\alpha \vee \beta = \max\{\alpha, \beta\}$.

2.4. Partial fuzzy set theory – handling of undefined and missing values

Situations when some part of the information is missing are very frequent. So, naturally, it is also quite usual that we have no information about membership degrees of some elements to particular fuzzy sets. This phenomenon was employed in *partial (three-valued) logics* that, besides the truth and false, allowed to deal with the third value, say `NA`. As the missing value `NA` could have a different origin, e.g., undefinedness, irrelevancy, inconsistency, or simply an unknown truth value, the variety of available partial logics is rather rich, see [37]. Recently, three-valued partial logics have been extended to *partial fuzzy logics* and partial fuzzy set theory ([20],[21]).

Typical representatives of partial fuzzy logics, that are implemented in the **lfl** package are the following ones: Bochvar, Sobociński, Kleene, and the Nelson logic. Furthermore, as none of the referred logics was specifically designed for handling the unknown values, two recent algebras for partial fuzzy logics were

Table 1: Handling of missing values by variants of residual negation

$\neg$	default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$
`NA`	`NA`	0	`NA`	1	`NA`	0

designed, in particular, the Lower estimation algebra [38], and the Dragonfly algebra [22]. In all cases, firstly an underlying algebra, e.g., Gödel, Goguen, or Łukasiewicz, is chosen and only then the truth-value interval is extended by an additional value `NA` in order to obtain $[0, 1] \cup$ `NA`.

The implementation of the basic algebras (Gödel, Goguen, Łukasiewicz) in **lfl** treats missing values natively in such a way that if `NA` appears as a value to some operation, it is propagated to the result. That is, any operation with `NA` results in `NA`, by default. This scheme of handling missing values is equivalent to the choice of the *Bochvar logic* [39].

However, the treatment of missing values may be easily changed in **lfl**. The `sobocinski()`, `kleene()`, `nelson()`, `lowerEst()` and `dragonfly()` functions modify the algebra given as their argument to handle `NA`s in a different way than by the default choice. For example, *Sobociński algebra* simply ignores `NA` values whereas *Kleene algebra* treats `NA` similarly to the Bochvar one however, extreme points 0 and 1 have a specific position among other truth value from the interval $[0, 1]$. *Dragonfly* approach as well as the *Lower estimation algebra* combine Sobociński and Bochvar approaches with the preservation of the Kleene-style specificity of truth values 0 and 1.. The distinct algebraic incorporation of the treatment of missing values is provided in Tables 1-6.

By default, the functions in the structure that is obtained by calling the `algebra()` function simply propagate `NA` to the output. If some other handling of missing values is required, it can be done as follows. Firstly, the underlying algebra (Gödel, Goguen or Łukasiewicz) is created and then modified by applying one of the `sobocinski()`, `kleene()`, `nelson()`, `dragonfly()`, `lowerEst()`

Table 2: Handling of missing values by variants of involutive negation

$\sim$	default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$	$f(\alpha)$
NA	NA	NA	NA	NA	NA	NA

Table 3: Handling of missing values by variants of conjunctive operations

$\otimes$, $\wedge$		default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	β	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$
0	NA	NA	0	0	0	0	0
α	NA	NA	α	NA	NA	NA	NA
1	NA	NA	1	NA	NA	NA	NA
NA	0	NA	0	0	0	0	0
NA	β	NA	β	NA	NA	NA	NA
NA	1	NA	1	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	NA	NA

Table 4: Handling of missing values by variants of disjunctive operations

$\oplus$, $\vee$		default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	β	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$
0	NA	NA	0	NA	NA	NA	NA
α	NA	NA	α	NA	NA	α	α
1	NA	NA	1	1	1	1	1
NA	0	NA	0	NA	NA	NA	NA
NA	β	NA	β	NA	NA	β	β
NA	1	NA	1	1	1	1	1
NA	NA	NA	NA	NA	NA	NA	NA

Table 5: Handling of missing values by variants of residuum

$\Rightarrow$		default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	β	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$
0	NA	NA	1	1	1	1	1
α	NA	NA	$\neg\alpha$	NA	NA	NA	NA
1	NA	NA	0	NA	NA	NA	NA
NA	0	NA	0	NA	1	NA	0
NA	β	NA	β	NA	NA	β	β
NA	1	NA	1	1	1	1	1
NA	NA	NA	NA	NA	1	1	NA

Table 6: Handling of missing values by variants of biresiduum

$\Leftrightarrow$		default (Bochvar)	`sobocinski`	`kleene`	`nelson`	`dragonfly`	`lowerEst`
α	β	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$	$f(\alpha,\beta)$
0	NA	NA	0	NA	1	NA	0
α	NA	NA	$\neg\alpha \wedge \alpha$	NA	NA	NA	NA
1	NA	NA	0	NA	NA	NA	NA
NA	0	NA	0	NA	1	NA	0
NA	β	NA	$\neg\beta \wedge \beta$	NA	NA	NA	NA
NA	1	NA	0	NA	NA	NA	NA
NA	NA	NA	NA	NA	1	1	NA

functions on it – see the example:

```
a <- algebra("goedel")
a2 <- sobocinski(a)
a$t(NA, 0.3)
```

```
## [1] NA
```

```
a2$t(NA, 0.3)
```

```
## [1] 0.3
```

3. Compositions of fuzzy relations

3.1. Fundamental compositions

Compositions of fuzzy relations establish one of the fundamental blocks for mathematical fuzzy modeling, see [16, 40]. Let us consider three non-empty universes X, Y, Z and let R and S be fuzzy relations on that universes, in particular, let $R \in \mathcal{F}(X \times Y)$ and $S \in \mathcal{F}(Y \times Z)$. In general, a composition of R and S results in a fuzzy relation $R@S \in \mathcal{F}(X \times Z)$ so, it defines some appropriate relationship between elements from universes that were not connected before defining the composition. The obligatory example comes from the medical diagnosis where X denotes a set of patients, Y denotes a set of symptoms and Z denotes a set of diseases [15].

The use of the compositions may be easily demonstrated on a toy example from medical diagnosis that by purposes simplifies the situation for the sake of clarity. Consider the following numeric matrices `R` and `S` defined in R:

```
print(R)
```

```
##          tired cough fever blur.vis
## patient1   0.9   1.0   0.8      0.0
## patient2   0.0   0.9   0.8      0.1
## patient3   0.0   0.8   0.9      0.0
## patient4   0.0   0.0   1.0      0.9
```

```
print(S)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## tired         1.0        1.0     0.1      0.9       0.0
## cough         0.9        0.2     0.9      0.0       1.0
## fever         0.0        1.0     0.0      1.0       1.0
## blur.vis      1.0        0.0     0.7      0.1       0.9
```

The values in matrix `R` represent the degrees to which the patients show the given symptoms (tiredness, cough, fever, blurred vision). The values in matrix `S` indicate the degrees to which the symptoms are assigned to the given diagnoses (pulmonary hypertension, sleeping sickness, malaria, hangover, influenza).

The main four types of fuzzy relational compositions implemented in **lfl** are defined as follows:

$$(R \circ S)(x,z) = \bigvee_{y \in Y} (R(x,y) \otimes S(y,z)) \ , \tag{1}$$

$$(R \triangleleft S)(x,z) = \bigwedge_{y \in Y} (R(x,y) \Rightarrow S(y,z)) \ , \tag{2}$$

$$(R \triangleright S)(x,z) = \bigwedge_{y \in Y} (R(x,y) \Leftarrow S(y,z)) \ , \tag{3}$$

$$(R \,\square\, S)(x,z) = \bigwedge_{y \in Y} (R(x,y) \Leftrightarrow S(y,z)) \tag{4}$$

where $\circ$ denotes the *circlet* or the *basic composition* (also the *direct product*), $\triangleleft$ denotes the *Bandler-Kohout subproduct* (also the *subdirect product*), $\triangleright$ denotes the *Bandler-Kohout superproduct* (also the *supdirect product*), and finally, $\square$ denotes the *Bandler-Kohout square product*.

Note that these four compositions were studied already in late 1970's and early 1980's [15, 41], the first two of them ($\circ$ and $\triangleleft$) play an essential role in fuzzy inference mechanisms in the case of fuzzy inputs [42, 43, 44], and their impact is essential for distinct branches including the medical diagnosis, see [45].

The main compositions, as defined above, may be computed in **lfl** with the `compose()` function as follows:

```
a <- algebra("lukasiewicz")
compose(R, S, alg=a, type="basic")
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
```

```
## patient1      0.9        0.9     0.9      0.8       1.0
## patient2      0.8        0.8     0.8      0.8       0.9
## patient3      0.7        0.9     0.7      0.9       0.9
## patient4      0.9        1.0     0.6      1.0       1.0
```

The `type` argument must be equal to one of: `"basic"` ($\circ$), `"sub"` ($\triangleleft$), `"super"` ($\triangleright$) or `"square"` ($\square$). The `compose()` function is merely a wrapper around the `mult()` function, which computes a customizable inner-product of two matrices. The `mult()` function takes two matrices as the arguments as well as a two-argument function, which is called for each combination of row and column. For instance, the basic composition may be equivalently computed as follows:

```
mult(R, S, function(r, s) {
  a$s(a$pt(r, s))
})
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1      0.9        0.9     0.9      0.8       1.0
## patient2      0.8        0.8     0.8      0.8       0.9
## patient3      0.7        0.9     0.7      0.9       0.9
## patient4      0.9        1.0     0.6      1.0       1.0
```

Additional examples of more complicated compositions computed directly with the `mult()` function may be found below in Section 3.3.

The fundamental fuzzy relational compositions (1)-(4) can be directly used in the expert classification problem, and the above-mentioned medical diagnosis [45] is only its special case where the symptoms are taken as the features and the set of diseases is a special case of the set of classes. However, observing the particular formulas, it is obvious that the huge gap between the existential quantifier, represented by $\bigvee$ in (1), and the universal quantifier, represented by $\bigwedge$ in (2)-(4), may cause undesirable effect. In particular, the basic composition $\circ$ may detect too many suspicions as finding a single "connecting" feature (symptom) will be a very frequent case for many classes (diseases) while carrying all the expected features (symptoms) may be rather idealistic, in practice too eliminating, requirement. Thus, $\circ$ could nominate too many candidate classes while $\triangleleft, \triangleright$, and $\square$ may, vice-versa, eliminate all possibilities:

```
compose(R, S, alg="lukasiewicz", type="sub")
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1      0.2        0.2     0.2      0.0       0.1
## patient2      0.2        0.3     0.2      0.1       1.0
## patient3      0.1        0.4     0.1      0.2       1.0
## patient4      0.0        0.1     0.0      0.2       1.0
```

```
compose(R, S, alg="lukasiewicz", type="super")
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1        0        0.8     0.3      0.8       0.1
## patient2        0        0.0     0.4      0.1       0.2
## patient3        0        0.0     0.3      0.1       0.1
## patient4        0        0.0     0.1      0.1       0.0
```

```
compose(R, S, alg="lukasiewicz", type="square")
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1        0        0.2     0.2      0.0       0.1
## patient2        0        0.0     0.2      0.1       0.2
## patient3        0        0.0     0.1      0.1       0.1
## patient4        0        0.0     0.0      0.1       0.0
```

In order to get out of the problem, distinct extensions were defined recently. One direction of the extensions led naturally to the employment of generalized quantifiers that fill in the gap between the existential one and the universal one and offer a tool to find an appropriate balance. The other one is based on employing additional fuzzy relations containing another knowledge that may be helpful in reducing the suspicions detected by the basic composition.

3.2. Compositions of more fuzzy relations

Assume that we are given fuzzy relations $E, U \in \mathcal{F}(Y \times Z)$. The intended semantical meaning is such that $E(y, z)$ expresses the degree up to which y is a feature that excludes the class z from the possible candidates (the so-called *excluding feature*) and $U(y, z)$ expresses the degree up to which y is a feature that is unavoidable for any object to be classified into the class z (the so-called *unavoidable feature*). The approach using the first fuzzy relation has been described by [46] while the work incorporating the latter one is very recent, see [30].

The compositions employing the concepts of excluding features (E) and unavoidable features (U) are defined as follows:

$$(R \circ S \backslash E)(x, z) = (R \circ S)(x, z) \otimes \neg(R \circ E) \ , \tag{5}$$

$$(R \circ S)^{\triangleright U}(x, z) = (R \circ S)(x, z) \otimes (R \triangleright U) \ , \tag{6}$$

$$(R \circ S \backslash E)^{\triangleright U}(x, z) = (R \circ S)(x, z) \otimes \neg(R \circ E) \otimes (R \triangleright U) \tag{7}$$

where the last one $(R \circ S \backslash E)^{\triangleright U}$ combines the extra knowledge contained in both additional fuzzy relations. As we may see, these extensions are mathematically rather straightforward combinations of the fundamental blocks that only allow other fuzzy relations to enter the constructions.

For example, let us imagine that the occurrence of fever is an evidence of *not* having a pulmonary hypertension. Similarly, let cough exclude hangover from the possible diagnoses. That is, fever and cough are excluding features of pulmonary hypertension and hangover, respectively. This corresponds to the following matrix `E` of excluding features:

```
print(E)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## tired           0          0       0        0         0
## cough           0          0       0        1         0
## fever           1          0       0        0         0
## blur.vis        0          0       0        0         0
```

To compute $(R \circ S \backslash E)$ as in (5), one can proceed with **lfl** as follows:

```
a <- algebra("lukasiewicz")
RS <- compose(R, S, alg=a, type="basic")
RE <- compose(R, E, alg=a, type="basic")
a$pt(RS, a$n(RE))
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1      0.1        0.9     0.9      0.0       1.0
## patient2      0.0        0.8     0.8      0.0       0.9
## patient3      0.0        0.9     0.7      0.1       0.9
## patient4      0.0        1.0     0.6      1.0       1.0
```

Now let us assume the blurred vision to be a typical unavoidable feature of pulmonary hypertension as well as cough would be unavoidable for influenza. This corresponds to the following matrix `U` of unavoidable features:

```
print(U)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## tired           0          0       0        0         0
## cough           0          0       0        0         1
## fever           0          0       0        0         0
## blur.vis        1          0       0        0         0
```

The $(R \circ S)^{\triangleright U}$ composition (see (6)) may be evaluated by the following commands:

```
RU <- compose(R, U, alg=a, type="super")
a$pt(RS, RU)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1      0.0        0.9     0.9      0.8       1.0
## patient2      0.0        0.8     0.8      0.8       0.8
## patient3      0.0        0.9     0.7      0.9       0.7
## patient4      0.8        1.0     0.6      1.0       0.0
```

Finally, the concept of unavoidable and excluding features together, as defined in (7), is processed as follows:

```
a$pt(RS, a$n(RE), RU)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1        0        0.9     0.9      0.0       1.0
## patient2        0        0.8     0.8      0.0       0.8
## patient3        0        0.9     0.7      0.1       0.7
## patient4        0        1.0     0.6      1.0       0.0
```

3.3. Compositions based on Sugeno integrals

Another approach to avoid the undesirable effect of too many suspicions (classification candidates) provided by the basic composition and too few (or often none) suspicions provided by the Bandler-Kohout products is based on employing generalized quantifiers. Intermediate generalized quantifiers allow to deal with quantifications in between of the classical universal and existential quantifiers, for example *"most"*, *"many"*, *"at least 3"*, *"at least 25"*, or *"a few"*. Note that the use of generalized quantifiers has been found very useful, e.g., in flexible query answering systems [47, 48].

The construction of a quantifier Q of the type $\langle 1 \rangle$ [49] uses a symmetric fuzzy measure μ, i.e., a monotone measure on the potential set satisfying the boundary condition. The direct application of the quantifier to the composition, i.e., $R@^Q S$, where $@ \in \{\circ, \triangleleft\}$, has been proposed by [50, 51] and it is defined as follows:

$$(R@^Q S)(x,z) = \bigvee_{D \in \mathcal{P}(Y) \setminus \{\emptyset\}} \left(\left(\bigwedge_{y \in D} R(x,y) \circledast S(y,z) \right) \otimes \mu(D) \right) \tag{8}$$

where $\mathcal{P}(Y)$ represents the powerset of Y, $\circledast \in \{*, \rightarrow\}$ corresponds to the composition, $x \in X$, and $z \in Z$.

Due to the choice of the symmetric fuzzy measure μ, it is sufficient to consider relative cardinality and its modification by distinct non-decreasing functions f in order to use Sugeno integral [52] to calculate the composition based on Q:

$$(R@^Q S)(x,z) = \bigvee_{i=1}^{n} \big(\big(R(x, y_{\pi(i)}) \circledast S(y_{\pi(i)}, z) \big) \otimes f(i/n) \big) \tag{9}$$

where n is the cardinality of Y, π is a permutation on $\{1, \ldots, n\}$ such that $R(x, y_{\pi(i)}) \circledast S(y_{\pi(i)}, z) \geq R(x, y_{\pi(i+1)}) \circledast S(y_{\pi(i+1)}, z)$ for any $i = 1, \ldots, n-1$.

The **lfl** package provides a function for computation of Sugeno integral, which can be used for composition of fuzzy relations with the `mult()` function described in Section 3.1.

For instance, one may require the patients to show at least two symptoms of the diagnosis. To quantify the *"at least 2"* condition over a fuzzy set, Sugeno integral will be applied. For that, we need a non-decreasing `measure` function that returns a truth value from the $[0, 1]$ interval. For us, a simple conditional function will do the work:

```
qatleast <- sugeno(measure=function(x) as.numeric(x >= 2),
                   relative=FALSE,
                   strong=TRUE,
                   alg="goedel")
qRS <- mult(R, S, function(r, s) {
  qatleast(a$pt(r, s))
})
print(qRS)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1      0.9        0.8       0      0.8       0.8
## patient2      0.1        0.1       0      0.0       0.8
## patient3      0.0        0.0       0      0.0       0.8
## patient4      0.0        0.0       0      0.0       0.8
```

The `sugeno()` function requires four arguments: `measure`, `relative`, `strong` and `alg`. The `measure` argument is a non-decreasing function that assigns a truth value from the $[0, 1]$ interval to either relative or absolute quantity. The `relative` argument is a `TRUE`/`FALSE` flag indicating what is expected by the `measure` function. The `strong` argument determines whether $\otimes$ in (8) and (9) is a strong or weak conjunction. Finally, the `alg` argument is an underlying algebra, i.e., either a string `"goedel"`, `"goguen"`, or `"lukasiewicz"`, or object of type `algebra` (see Section 2).

The result of the `sugeno()` function is a function that expects a vector of membership degrees to be measured.

Remark 1. *The above proposed formula (9) for the composition based on Sugeno integrals is implemented for particular algebras for partial fuzzy logic, namely for Bochvar, Dragonfly and Lower estimation algebra. The ordering of the extended set of truth-values* $[0, 1] \cup$ `NA` *becomes of great importance. In the case of the Bochvar algebra, the ordering, defined by* `NA` $\leq \alpha$ *for any* $\alpha \in [0, 1]$*, preserves the equality of (8) and (9) and thus, is adopted. Apart from the Bochvar case, which is default, the users may choose either the Dragonfly algebra or the Lower estimation algebra. In both cases, a "lattice-like" ordering* $\leq_\ell$ *generated by the lattice operations* $\wedge$ *and* $\vee$ *ensures the same preservation of the equality of (8) and (9). It is defined as follows:* $0 \leq_\ell$ `NA` $\leq_\ell \alpha$ *for all* $\alpha \in (0, 1]$*, see [22].*

3.4. Combined cases

As the whole implementation of compositions in **lfl** is based on functions, it is very easy to follow the block structure of distinct extension and thus, to call compositions using combinations of generalized quantifiers and additional fuzzy

relations E and U. For example, the use of the following composition:

$$(R \circ^Q S^{\backslash} E)^{\triangleright U}(x, z) = (R \circ^Q S)(x, z) \otimes \neg(R \circ E) \otimes (R \triangleright U)$$

turned to be very efficient in classifying dragonfly species as well as amphibian species, for appropriately chosen quantifiers, see [30].

In the **lfl** package, such complex formula is evaluated by the following code:

```
a$pt(qRS, a$n(RE), RU)
```

```
##          pulm.hyp sleep.sick malaria hangover influenza
## patient1        0        0.8       0        0       0.8
## patient2        0        0.1       0        0       0.7
## patient3        0        0.0       0        0       0.6
## patient4        0        0.0       0        0       0.0
```

4. Evaluative linguistic expressions

Evaluative linguistic expressions [3] are expressions vaguely describing a position on a quantitative axis no matter that not necessarily the position can be numerically expressed, for example, we may consider expressions such as *"very nice"*, *"not very intelligent"*, *"extremely friendly"*. Such expressions have either the form

$$\langle \text{linguistic hedge} \rangle \langle \text{atomic expression} \rangle$$

or they are expressed as vague quantities also called fuzzy numbers [53].

The latter case of fuzzy numbers is in the **lfl** package modelled with help of triangular or raised cosine that reach normality fuzzy set. The earlier case is based on a small set of *atomic expressions*. Originally, the trichotomy *"small"*, *"medium"*, *"big"* was used however, later on, an extension to a pentachotomy by adding *"lower medium"*, *"upper medium"* was proposed as an alternative for better distinction of medium-close values. The set of *hedges* serve as linguistic modifiers making the meaning of an atomic expression wider or narrower. If we consider an empty hedge as a special case between the hedges with narrowing effect and the hedges with widening effect, we obtain a linear order of hedges:

$$\mathrm{Ex} \leq_{\mathrm{H}} \mathrm{Si} \leq_{\mathrm{H}} \mathrm{Ve} \leq_{\mathrm{H}} \langle \text{empty} \rangle \leq_{\mathrm{H}} \mathrm{ML} \leq_{\mathrm{H}} \mathrm{Ro} \leq_{\mathrm{H}} \mathrm{QR} \leq_{\mathrm{H}} \mathrm{VR}$$

where the abbreviations stand for *"extremely"*, *"significantly"*, *"very"*, *"more or less"*, *"roughly"*, *"quite roughly"*, and *"very roughly"*, respectively. Note that not all hedges have to be necessarily used and some redundancy analysis results are valid only under assumptions that, e.g., require to omit the hedge VR.

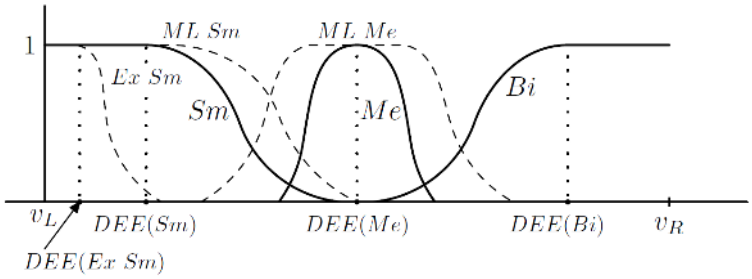


Figure 1: Visual sketch of fuzzy sets representing some particular evaluative linguistic expressions. Specific defuzzification DEE is charted too.

The ordering of hedges has an important role in the inference mechanism tailored to the linguistic fuzzy rules with the above mentioned evaluative linguistic expressions. It is based on specificity that is directly determined by the used hedge, assuming the same atomic expression is used. The narrower the hedge, the more specific the expression is. And if an expression is more specific that another one, the respective fuzzy set that models its meaning is a fuzzy subset of the fuzzy set that models the less specific expression. This inclusion is visualized on Fig. 1.

4.1. Linguistic context

The *linguistic context* is a sort of extended notion to the notion of the universe in a sense that the universe is also accompanied with some fundamental points. For the case of modeling expressions on numerical axis, we may dare to restrict our focus to universes that are closed real intervals so, the universe would be $U = [v_L, v_R]$. In order to talk about linguistic expression, we need to add at least the third "middle" point that does not represent the center of the interval however, it denotes the most typical value for middle size objects in

the given domain. Usually, as humans are more sensitive to smaller values than to the big ones, the middle point is closer to v_L than to v_R. For example, we may consider the universe of pine trees $U = [3, 80]$ (in meters) while the middle size pine tree would be rather around 15-20 meters than in the middle of the interval U. So, extending the context to $[v_L, v_C, v_R]$ is not just a redundant adding of the central point v_C but a desirable specification of the prototypical middle point v_C.

As the context has to reflect *unilateral/bilateral* nature (if it respects the positive and negative values) or the decision whether we deal with *trichotomy* or a *pentachotomy* (pentachotomy would require two more such emphasized points), the order triplet $[v_L, v_C, v_R]$ is not the only choice but the most fundamental and the simplest form of a linguistic context. In particular, four different contexts are supported in **lfl**, and the above-mentioned simplest context is the *unilateral trichotomy* that is chosen by calling the function `ctx3()`. The higher density of atomic expressions can be obtained by adding expressions *"lower middle"* and *"upper middle"*, which requires to consider *unilateral pentachotomy* by calling the function `ctx5()`. The bilateral versions of the trichotomy and the pentachotomy allow to deal with expressions such as *"negative small"*, *"positive medium"*, or *"negative very small"* and can be called by functions `ctx3bilat()` and `ctx5bilat()`, respectively. The summary of functions responsible for creation of the linguistic context is provided below:

- `ctx3(low, center, high)`: the unilateral trichotomy that enables the atomic expressions: *"small", "medium", "big"*;
- `ctx5(low, lowerCenter, center, upperCenter, high)` – the unilateral pentachotomy that enables the atomic expressions: *"small", "lower medium", "medium", "upper medium", "big"*;
- `ctx3bilat(negMax, negCenter, origin, center, max)` – the bilateral trichotomy that enables the atomic expressions: *"negative big", "negative medium", "negative small", "zero", "small", "medium", "big"*;

- `ctx5bilat(negMax, negUpperCenter, negCenter, negLowerCenter, origin, lowerCenter, center, upperCenter, max)` – the bilateral pentachotomy that enables the atomic expressions: *"negative big", "negative upper medium", "negative medium", "negative lower medium", "negative small", "zero", "small", "lower medium", "medium", "upper medium", "big"*.

The arguments of context creator functions have sensible defaults and need not be therefore explicitly stated in all cases:

```
ctx3(5, 100, 1000)
```

```
## Linguistic context: unilateral trichotomy (ctx3)
##    low center   high
##      5    100   1000
```

```
ctx3()
```

```
## Linguistic context: unilateral trichotomy (ctx3)
##    low center   high
##    0.0    0.5    1.0
```

```
ctx3(high=100)
```

```
## Linguistic context: unilateral trichotomy (ctx3)
##    low center   high
##      0     50    100
```

Alternatively, the context may be automatically determined from data by calling the `minmax()` function, which creates the selected type of the context based on the minimum and maximum value found in data:

```
data <- runif(n=100, min=20, max=5000)
summary(data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   118.6  1306.3  2539.8  2580.3  3794.6  4997.8
```

```
minmax(data, type="ctx3")
```

```
## Linguistic context: unilateral trichotomy (ctx3)
##      low   center     high
##  118.573 2558.195 4997.816
```

The `minmax()` function may be forced not to guess some values by specifying them explicitly as additional arguments:

```
minmax(data, type="ctx3", center=1000)
```

```
## Linguistic context: unilateral trichotomy (ctx3)
##      low   center     high
##  118.573 1000.000 4997.816
```

4.2. Evaluative linguistic expressions

The atomic expressions, e.g., *"small"*, *"medium"* or *"big"* in the case of the unilateral trichotomy, are according to the theory of evaluative linguistic expressions [3] modelled with help of the `horizon()` function. Horizon of the atomic expression is a function that represents basic limits of what humans treat as small, medium or big, see Fig. 2.

```
ctx <- ctx3()
smHoriz <- horizon(ctx, atomic="sm")
smHoriz(seq(from=0, to=1, by=0.2))
```

```
## [1] 1.0 0.6 0.2 0.0 0.0 0.0
```

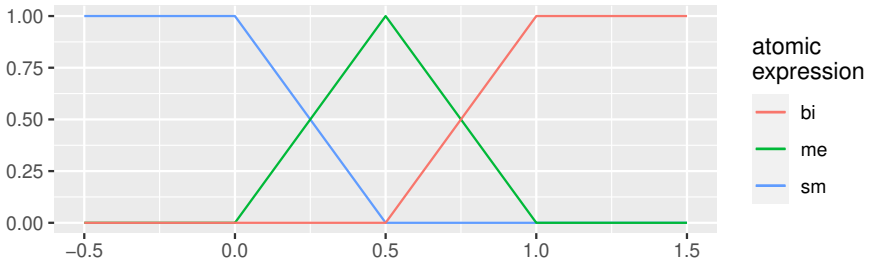


Figure 2: Horizons for the atomic expressions *small*, *medium* and *big* in the unilateral trichotomy (`ctx3(0, 0.5, 1)`)

A particular linguistic expression is obtained after the application of the linguistic hedge. So, in **lfl**, the `hedge()` function works as a modifier function, which is applied to a particular horizon. For instance, the function that represents the "very small" expression can be obtained as follows:

```
veHedge <- hedge("ve")
ve.sm <- function(x) veHedge(smHoriz(x))
ve.sm(seq(from=0, to=0.5, by=0.1))
```

```
## [1] 1.000000 0.585098 0.000000 0.000000 0.000000 0.000000
```

Such an approach gives the user a detailed control of the creation of a linguistic expression, which may be useful for experimenting with novel expressions. However, it may be tedious to manually create the functions for a routine use. Therefore, the `lingexpr()` function provides a shortcut for creation of pre-defined expressions:

```
ve.sm2 <- lingexpr(ctx, atomic="sm", hedge="ve")
ve.sm2(seq(from=0, to=0.5, by=0.1))
```

```
## [1] 1.000000 0.585098 0.000000 0.000000 0.000000 0.000000
```

An expression, consisting of an atomic expression only, is constructed using an empty hedge:

```
emptyHedge <- hedge("-")
sm <- function(x) emptyHedge(smHoriz(x))
sm(seq(from=0, to=0.5, by=0.1))
```

```
## [1] 1.0000000 0.9620685 0.2439553 0.0000000 0.0000000 0.0000000
```

or equivalently:

```
sm2 <- lingexpr(ctx, atomic="sm", hedge="-")
sm2(seq(from=0, to=0.5, by=0.1))
```

```
## [1] 1.0000000 0.9620685 0.2439553 0.0000000 0.0000000 0.0000000
```

Figure 3 shows all linguistic expressions of the unilateral trichotomy context (`ctx3`).

4.3. Other functions

For the sake of completeness, the **lfl** package provides tools for the creation of triangular or raised-cosine functions. Both `triangular()` and `raisedcosine()` functions take three arguments, `lo`, `center`, `hi`, which fully parameterize the shape of the resulting function. See the example below as well as Figure 4 for more detail. Note also that the `lo` and `hi` parameters may be set to an infinite value (`-Inf` resp. `Inf`), which causes the particular tail to be constantly equal to 1.

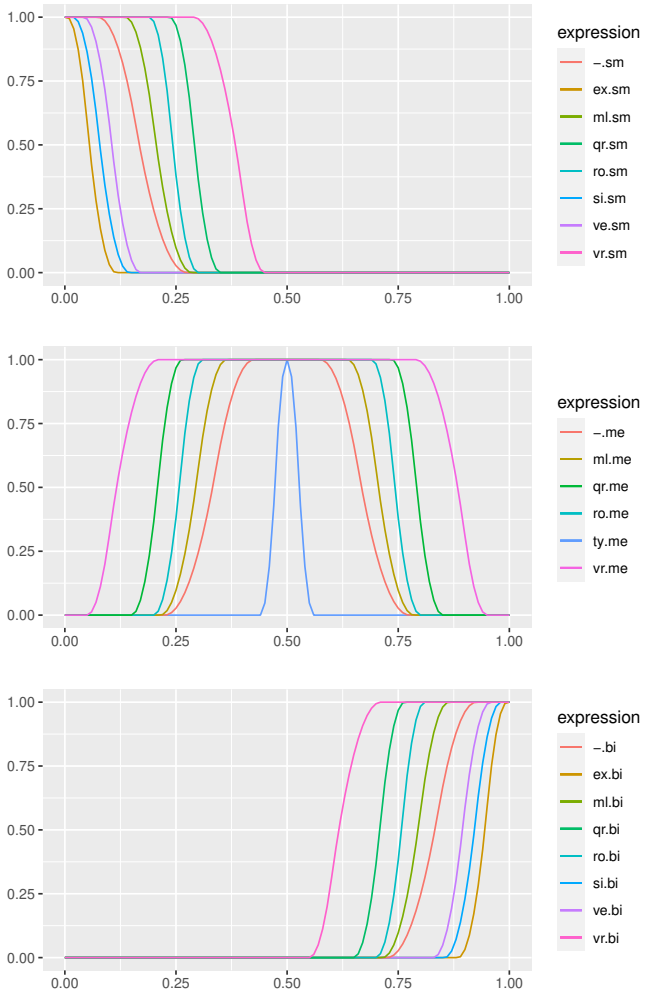


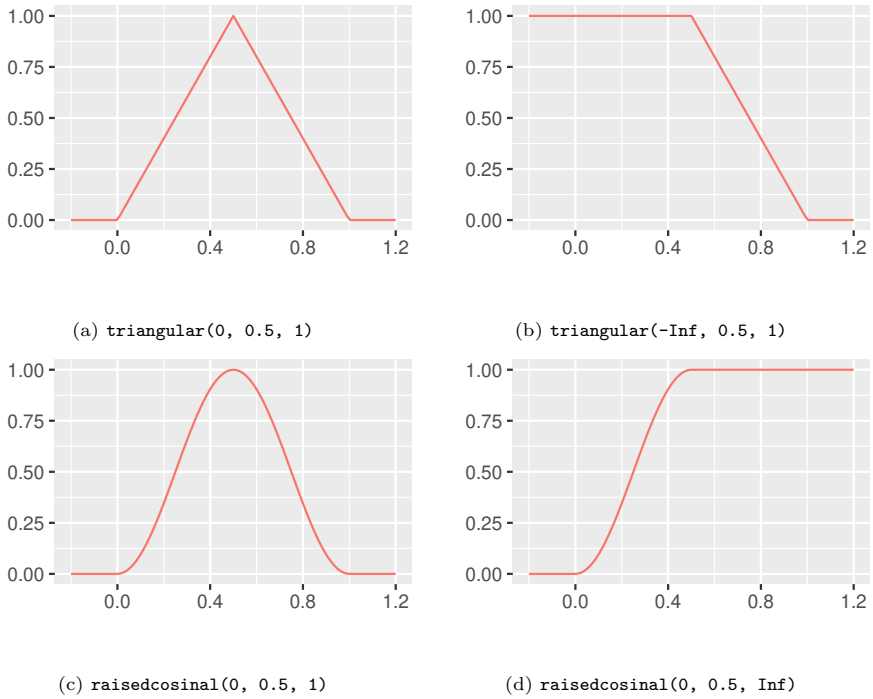
Figure 3: All pre-defined linguistic expressions for the unilateral trichotomy context `ctx3`

```
tri <- triangular(0, 0.5, 1)
tri(seq(from = 0, to = 1, by = 0.2))

## [1] 0.0 0.4 0.8 0.8 0.4 0.0

rcos <- raisedcosine(0, 0.5, 1)
rcos(seq(from = 0, to = 1, by = 0.2))

## [1] 0.0000000 0.3454915 0.9045085 0.9045085 0.3454915 0.0000000
```



(a) `triangular(0, 0.5, 1)` (b) `triangular(-Inf, 0.5, 1)`

(c) `raisedcosinal(0, 0.5, 1)` (d) `raisedcosinal(0, 0.5, Inf)`

Figure 4: Triangular and raised-cosinal functions

4.4. Batch transformations of data to membership degrees of fuzzy sets

Practical applications often require to transform data into multiple fuzzy sets. The **lfl** package provides the `lcut()` and `fcut()` functions to perform such transformations. Both functions transform vectors (numeric, logical or factors), matrices or data frames into an `fsets` object. Such an object is a data frame with each column representing a single fuzzy set. The values are from the

$[0, 1]$ interval and they are equal to the membership degrees of the element of the universe to the particular fuzzy sets. These functions are the fuzzy-counterparts of the well known `cut()` operation of the base R.

For logical input, the `lcut()` function returns two columns of 0s and 1s: these columns represent (crisp) truth degrees equivalent to the original input and its negation, respectively. The name of the column is either specified by the user in the `name` argument, or derived from the given data argument automatically:

```
logvec <- c(TRUE, FALSE, TRUE, TRUE)
lcut(logvec)
```

```
##      logvec not.logvec
## [1,]      1          0
## [2,]      0          1
## [3,]      1          0
## [4,]      1          0
```

```
lcut(logvec, name="employed")
```

```
##      employed not.employed
## [1,]        1            0
## [2,]        0            1
## [3,]        1            0
## [4,]        1            0
```

The factor input is dichotomized in the result:

```
position <- factor(c("worker", "manager", "worker", "accountant"))
lcut(position)
```

```
##      position=accountant position=manager position=worker
## [1,]                   0                0               1
## [2,]                   0                1               0
## [3,]                   0                0               1
## [4,]                   1                0               0
```

For numeric input, the `lcut()` function performs transformation to linguistic expressions similarly as described in Section 4.2. For this step, a linguistic context must be provided (see Section 4.1). If the context is not provided, it is determined automatically using the `minmax()` function described in Section 4.1. The required atomic expressions or hedges may be specified manually or leaved empty to let the system use all relevant combinations:

```
age <- round(runif(n=4, min=18, max=65))
print(age)
```

```
## [1] 55 32 41 52
```

```
lcut(age,
     context=ctx3(low=0, high=100))
```

```
##      ex.sm.age si.sm.age ve.sm.age sm.age ml.sm.age ro.sm.age qr.sm.age
## [1,]         0         0         0      0         0         0 0.0000000
## [2,]         0         0         0      0         0         0 0.1315789
## [3,]         0         0         0      0         0         0 0.0000000
## [4,]         0         0         0      0         0         0 0.0000000
##      vr.sm.age  ty.me.age    me.age ml.me.age ro.me.age qr.me.age
## [1,] 0.0000000 0.04761905 1.0000000 1.0000000         1         1
## [2,] 0.9475480 0.00000000 0.3914128 0.7993319         1         1
## [3,] 0.1993769 0.00000000 0.9859853 1.0000000         1         1
## [4,] 0.0000000 0.73333333 1.0000000 1.0000000         1         1
##      vr.me.age ex.bi.age si.bi.age ve.bi.age bi.age ml.bi.age ro.bi.age
## [1,]         1         0         0         0      0         0         0
## [2,]         1         0         0         0      0         0         0
## [3,]         1         0         0         0      0         0         0
## [4,]         1         0         0         0      0         0         0
##      qr.bi.age vr.bi.age
## [1,]         0         0
## [2,]         0         0
## [3,]         0         0
## [4,]         0         0
```

If data frame is to be processed with the `lcut()` function, the result is created per column. Also note that the names of the resulting variables are derived from the column names of the input data frame. For the sake of brevity, the result's column names are listed only:

```
data <- data.frame(position=position,
                   age=age,
                   employed=logvec)
print(data)
```

```
##     position age employed
## 1     worker  55     TRUE
## 2    manager  32    FALSE
## 3     worker  41     TRUE
## 4 accountant  52     TRUE
```

```
employees <- lcut(data,
     context=ctx3(low=0, high=100),
     atomic=c("sm", "me", "bi"),
     hedges=c("ve", "-", "ro"))
print(colnames(employees))
```

```
##  [1] "position=accountant" "position=manager"    "position=worker"
##  [4] "ve.sm.age"           "sm.age"              "ro.sm.age"
##  [7] "me.age"              "ro.me.age"           "ve.bi.age"
## [10] "bi.age"              "ro.bi.age"           "employed"
## [13] "not.employed"
```

The given contexts, atomic expressions and hedges are recycled for each input numeric column. If a different setting is needed for each column, the arguments should be given as named lists as follows:

```
data$salary <- round(runif(n=4, min=1000, max=20000))
print(data)
```

```
##     position age employed salary
## 1     worker  55     TRUE  14191
## 2    manager  32    FALSE   7728
## 3     worker  41     TRUE  16279
## 4 accountant  52     TRUE  15044
```

```
employees <- lcut(data,
                  context=list(age=ctx3(low=0, high=100),
                               salary=ctx3(low=500, high=50000)),
                  atomic=list(salary=c("sm", "bi")),
                  hedges=list(age=c("ve", "-", "ro"),
                              salary=c("ex", "ve", "-", "ro")))
print(colnames(employees))
```

```
##  [1] "position=accountant" "position=manager"    "position=worker"
##  [4] "ve.sm.age"           "sm.age"              "ro.sm.age"
##  [7] "me.age"              "ro.me.age"           "ve.bi.age"
## [10] "bi.age"              "ro.bi.age"           "employed"
## [13] "not.employed"        "ex.sm.salary"        "ve.sm.salary"
## [16] "sm.salary"           "ro.sm.salary"        "ex.bi.salary"
## [19] "ve.bi.salary"        "bi.salary"           "ro.bi.salary"
```

The `fsets` object returned by the `lcut()` function (and the `fcut()` function as well, see below) handles an additional information, the `vars` and `specs` attributes. In particular, `vars` is a character vector that assigns the original

data name to each of the resulting column of membership degrees. In other words, the `vars` vector specifies the equivalence classes of fuzzy sets that were originated from the same data:

```
vars(employees)
```

```
##  [1] "position" "position" "position" "age"      "age"      "age"
##  [7] "age"      "age"      "age"      "age"      "age"      "employed"
## [13] "employed" "salary"   "salary"   "salary"   "salary"   "salary"
## [19] "salary"   "salary"   "salary"
```

The `specs` attribute returns a matrix that encodes a sort of *specificity* relation (see Section 6) between the columns of the `fsets` object. (In the following example, some columns and rows are omitted for brevity.

```
specs(employees)[1:5, 1:5]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    0    0    0
## [2,]    0    0    0    0    0
## [3,]    0    0    0    0    0
## [4,]    0    0    0    0    1
## [5,]    0    0    0    0    0
```

As can be seen, the 4th fuzzy set (`ve.sm.age`) is more specific than 5th (`sm.age`), which is more specific than the 6th fuzzy set (`ro.sm.age`).

The `fcut()` function works identically to `lcut()` for logical and factor input. However, numerical values are transformed with the `fcut()` function to triangular or raised-cosine membership degrees (depending on the `type` argument which has to be either `"triangle"` or `"raisedcos"`):

```
numvec <- 1:9
fcut(numvec,
     breaks=c(1, 5, 9),
     type="triangle")
```

```
##       numvec=1
##  [1,]     0.00
##  [2,]     0.25
##  [3,]     0.50
##  [4,]     0.75
##  [5,]     1.00
```

```
##  [6,]     0.75
##  [7,]     0.50
##  [8,]     0.25
##  [9,]     0.00
```

This is identical to the call of the `triangular()` function:

```
triangular(1, 5, 9)(numvec)
```

```
## [1] 0.00 0.25 0.50 0.75 1.00 0.75 0.50 0.25 0.00
```

However, the `fcut()` function is mainly useful for creation of multiple fuzzy sets. A mandatory `breaks` argument determines the break-points of the positions of the fuzzy sets. It should be an ordered vector of numbers such that the i-th index specifies the infimum of the support (left-hand side corner), $(i+1)$-th the center, and $(i+2)$-th the supremum of the support of the i-th fuzzy set. The minimum number of break-points is 3. $n-2$ elementary fuzzy sets would be created for n break-points.

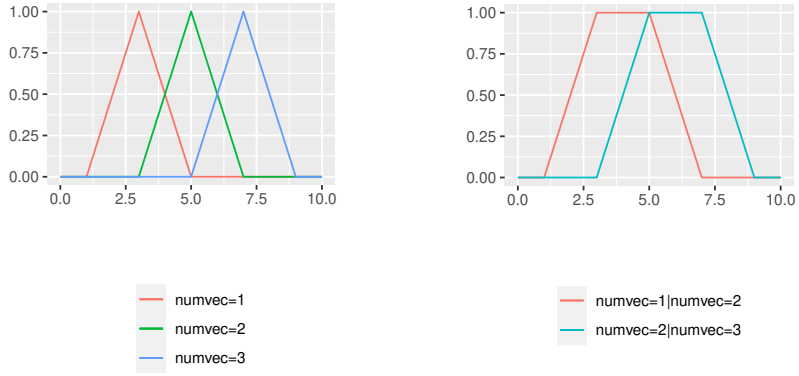
For instance, the following command produces three triangular fuzzy sets with parameters $(1, 3, 5)$, $(3, 5, 7)$ and $(5, 7, 9)$:

```
fcut(numvec,
     breaks=c(1, 3, 5, 7, 9),
     type="triangle")
```

```
##       numvec=1 numvec=2 numvec=3
##  [1,]      0.0      0.0      0.0
##  [2,]      0.5      0.0      0.0
##  [3,]      1.0      0.0      0.0
##  [4,]      0.5      0.5      0.0
##  [5,]      0.0      1.0      0.0
##  [6,]      0.0      0.5      0.5
##  [7,]      0.0      0.0      1.0
##  [8,]      0.0      0.0      0.5
##  [9,]      0.0      0.0      0.0
```

Let us consider the i-th fuzzy set. The values smaller than the i-th break and greater than $(i+2)$-th break result in the zero membership degree, values equal to $(i+1)$-th break result in the membership degree equal 1, and values between them result in a membership degree between 0 and 1 accordingly to

the specified `type` (`"triangle"` or `"raisedcos"`). Names of resulting fuzzy sets are created from the name of the original data variable (`numvec` in this case), the symbol of equality (`=`) and a number i.



(a) `merge=1` (b) `merge=2`

Figure 5: Results of the `fcut` call with `breaks=c(1, 3, 5, 7, 9)` and different settings of `merge`.

Additionally, combined fuzzy sets may be created by using the argument `merge`. If `merge=1` (the default), only the elementary fuzzy sets discussed above are produced. Setting `merge=2` means that each two consecutive elementary fuzzy sets should be combined with the Łukasiewicz t-conorm into a single fuzzy set, `merge=3` causes combining three consecutive elementary fuzzy sets etc. See Fig. 5 and also the following example.

The `merge` argument determines whether to derive additional fuzzy sets by merging the elementary fuzzy sets (defined with the `breaks` argument) into super-sets. The `merge` may contain any integer number from `1` to `length(breaks) - 2`. Value `1` means that the elementary fuzzy sets have to be created only, as described above (the default case).

```
fcut(numvec,
     breaks=c(1, 3, 5, 7, 9),
     merge=2,
     type="triangle")
```

```
##       numvec=1|numvec=2 numvec=2|numvec=3
```

```
##  [1,]               0.0               0.0
##  [2,]               0.5               0.0
##  [3,]               1.0               0.0
##  [4,]               1.0               0.5
##  [5,]               1.0               1.0
##  [6,]               0.5               1.0
##  [7,]               0.0               1.0
##  [8,]               0.0               0.5
##  [9,]               0.0               0.0
```

```
fcut(numvec,
     breaks=c(1, 3, 5, 7, 9),
     merge=3,
     type="triangle")
```

```
##       numvec=1|numvec=2|numvec=3
##  [1,]                        0.0
##  [2,]                        0.5
##  [3,]                        1.0
##  [4,]                        1.0
##  [5,]                        1.0
##  [6,]                        1.0
##  [7,]                        1.0
##  [8,]                        0.5
##  [9,]                        0.0
```

The `merge` argument may contain multiple values. In that case, all types of merged fuzzy sets are provided. In the following example, the `merge` argument is a numeric vector containing `1, 2, 3`, which means that the elementary fuzzy sets are created (`1`), the combinations of two consecutive elementary fuzzy sets are provided too (`2`), as also a single fuzzy set that combines all the three elementary fuzzy sets is returned (`3`):

```
fd <- fcut(numvec,
           breaks=c(1, 3, 5, 7, 9),
           merge=c(1, 2, 3),
           type="triangle")
print(colnames(fd))
```

```
## [1] "numvec=1"                  "numvec=2"
## [3] "numvec=3"                  "numvec=1|numvec=2"
## [5] "numvec=2|numvec=3"         "numvec=1|numvec=2|numvec=3"
```

As can be seen, the names of the derived (merged) fuzzy sets are created from the names of the original elementary fuzzy sets by concatenating them with the pipe (`|`) separator.

Similarly as for `lcut()`, the result of the `fcut()` function is an instance of the `fsets` object. Hence the additional information, the `vars` and `specs` attributes discussed above, is also available:

```
vars(fd)
```

```
## [1] "numvec" "numvec" "numvec" "numvec" "numvec" "numvec"
```

```
specs(fd)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    0    0    1    0    1
## [2,]    0    0    0    1    1    1
## [3,]    0    0    0    0    1    1
## [4,]    0    0    0    0    0    1
## [5,]    0    0    0    0    0    1
## [6,]    0    0    0    0    0    0
```

5. Fuzzy association rules

5.1. Theoretical background

The association rules [54] need not be introduced in detail, we only recall the fact that firstly the method appeared under the name GUHA [55, 56] and furthermore, we recall basic principles of how this method finds distinct statistically approved associations between attributes of given objects. With help of **lfl** the method can be used also in the fuzzy setting, i.e., for graded properties.

The crisp version of association rules deals with Table 7 where $o_1, \ldots, o_n$ denote objects, $X_1, \ldots, X_m$ denote independent boolean attributes, $Y_1, \ldots, Y_q$ denote the dependent boolean attributes, and finally, symbols a_{ij} and b_{ij} are values from $\{0, 1\}$ that denote whether the i-th object o_i carries attribute X_j or Y_j, respectively. Each object can be represented as a boolean vector with $m + q$ components.

	X_1	$\dots$	X_m	Y_1	$\dots$	Y_q
o_1	a_{11}	$\dots$	a_{1m}	b_{11}	$\dots$	b_{1q}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_n	a_{n1}	$\dots$	a_{nm}	b_{n1}	$\dots$	b_{nq}

Table 7: The table objects and attributes.

As the GUHA method deals with boolean attributes and features are often taking values from a real universe, the attributes are usually partitioned to intervals. Then the method seeks for associations:

$$\mathtt{A}(X_{i1}, \dots, X_{ip}) \simeq \mathtt{B}(Y_k) \tag{10}$$

where A is a predicate with conjunctively connected variables $X_{i1}, \dots, X_{ip}$ (for $p \leq m$), and B is a predicate with variables Y_k, for k taking values from 1 to q. In order to mine such associations, a four-fold Table 8 is constructed.

	B	not B
A	a	b
not A	c	d

Table 8: Four-fold table for mining linguistic associations.

The number of synchronous occurrences of A and B is denoted by a in Table 8, b denotes the number of occurrences of A while B does not hold, numbers c and d are determined analogously. Often, only numbers a and b are important for distinct qualitative measures. For instance, the relationship between A and B may be obtained with help of the *binary multitudinal quantifier* $\simeq := \sqsubset_r^\gamma$ that confirms the association if:

$$\frac{a}{a+b} > \gamma \qquad \text{and} \qquad \frac{a}{n} > r,$$

where $\gamma \in [0,1]$ is a *degree of confidence* and $r \in [0,1]$ is a *degree of support*.

In order to soften the binary character of the associations and the sensitivity to the partitioning of the universes into intervals, distinct approaches to fuzzy

associations were employed [57, 58, 59]. The **lfl** package adopts the approach published in [6] because it directly uses the theory of evaluative linguistic expressions. The attributes are not boolean anymore. Recalling the example from [6], we may consider independent variables BMI (Body Mass Index) and Chol (cholesterol level), and the dependent variable BP (blood pressure) and the attributes such as BMI^{VeBi}, BP^{Me}, $\text{Chol}^{\text{SiBi}}$ etc. that are defined by the fuzzy sets modeling the particular evaluative expressions. In such a way, the authors obtained Table 9 with membership degrees $a_{ij} \in [0, 1]$.

	BMI^{ExSm}	...	$\text{Chol}^{\text{ExBi}}$	BP^{ExSm}	...	BP^{ExBi}
o_1	0.5	...	0	1	...	0
o_2	0.8	...	0	0.4	...	0
o_3	0	...	0.1	0	...	0.4
o_4	0	...	0.4	0	...	0.3
o_5	0.6	...	0	1	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
o_n	0	...	0	0.5	...	0

Table 9: An example of the table with fuzzy attributes for fuzzy associations mining, ref. [6].

Table 8 is constructed for the fuzzy case in the same way using the cardinality, i.e., values a, b, c, d are summations of membership degrees. The conjunctive aggregations of the truth-values of `A` and `B` were modeled by the minimum operation however, other t-norms [60] may be considered as well. So, if the membership degree of o_i to `A` is 0.4 and its membership degree to `B` is 0.7, the value that enters the summation equals to $\min\{0.4, 0.7\} = 0.4$. If we sum up such values over all objects, we obtain the number a in Table 8. The other numbers are determined in an analogous way.

The advantage of associations (10) is that each of them can be interpreted as the fuzzy rule:

$$\mathcal{R} := \mathsf{IF}\ X_{i1}\ \text{is}\ \mathcal{A}_{i1}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_{ip}\ \text{is}\ \mathcal{A}_{ip}\ \mathsf{THEN}\ Y_k\ \text{is}\ \mathcal{B}_{ik}$$

that may be used in the inference systems for approximate reasoning.

5.2. Searching for fuzzy association rules in ***lfl***

The **lfl** package provides the `searchrules()` function that implements an OPUS-inspired algorithm [61] for searching for fuzzy association rules in data. The `searchrules()` function traverses through the data set transformed to fuzzy sets (see `lcut()` and `fcut()` functions in Section 4.4) and searches for all rules that satisfy certain restrictive conditions specified by the user:

```
rb <- searchrules(employees,
                  lhs=seq_len(ncol(employees)),
                  rhs=seq_len(ncol(employees)),
                  minSupport=0.5,
                  minConfidence=0.8,
                  maxLength=3)
print(rb)
```

```
##                                support lhsSupport rhsSupport confidence
## position=worker => employed 0.5000000  0.5000000  0.7500000  1.0000000
##  => ro.me.age               1.0000000  1.0000000  1.0000000  1.0000000
## employed => me.age          0.7464963  0.7500000  0.8443495  0.9953284
## me.age => employed          0.7464963  0.8443495  0.7500000  0.8841082
##  => me.age                  0.8443495  1.0000000  0.8443495  0.8443495
```

Besides the `fsets` data object (see Section 4.4), the `searchrules()` function accepts the following arguments:

- `lhs` – indices of data columns that may appear on the left-hand side of the rule (i.e., in the antecedent);
- `rhs` – indices of data columns that may appear on the right-hand side of the rule (i.e., in the consequent);
- `tnorm` – the t-norm that represents the conjunction of predicates in the antecedent (defaults to Gödel minimum);
- `n` – the maximum number of rules with greatest confidence to be found;
- `minSupport` – the minimum support degree of a rule. Rules with support below that number are filtered out;

- `minConfidence` – the minimum confidence degree of a rule. Rules with confidence below that number are filtered out;
- `maxConfidence` – the maximum confidence threshold. After finding a rule that has confidence degree above the maxConfidence threshold, no other rule is resulted based on adding some additional attribute to its antecedent part. So, if `a => c` has confidence above the `maxConfidence` threshold, no more rules containing `a` in the antecedent and `c` in the consequent will be produced regardless of their interest measures;
- `maxLength` – the maximum allowed length of the antecedent, i.e. maximal number of predicates that are allowed to appear on both sides of the rule. If negative, the maximum length of rules is unlimited.
- `numThreads` – the number of computing threads to start in parallel multi-threaded computation.

Internally, the rule base produced by the `searchrules()` function is an instance of the `farules` class. It is a list of two elements, `rules` and `statistics`, where `rules` is a list of character vectors of rule predicate names. The first element of these vectors is a consequent of the rule, the rest is the antecedent. The `statistics` element is a matrix that assigns some quality measures to each rule. The `as.data.frame()` function converts the `farules` object to a usual data frame. The `antecedent()` or `consequent()` functions create a list of antecedents or consequents from the `farules` object.

5.3. Reduction of rule bases

If the generated association rules are intended for use in some inference mechanisms (such as PbLD described in Section 6), it is sometimes useful to employ a `reduce()` function to decrease the amount of rules obtained by the `searchrules()` function.

The *rule base coverage of data* expresses the amount of data entries, for which there exists a rule with an antecedent that models (that is, "covers")

the original data. The reduction algorithm performed in the `reduce()` function selects a minimal rule base that covers at least the specified ratio of data. The algorithm described by [62] turns out to be very efficient in reduction while retaining the output of the PbLD inference.

For example, a set `rb` of rules obtained from the `employees` fuzzy sets may be reduced to the 90 % coverage ratio as follows:

```
reduce(employees, rb, 0.9)
```

```
##                         support lhsSupport rhsSupport confidence
##  => me.age            0.8443495  1.0000000  0.8443495  0.8443495
##  => ro.me.age         1.0000000  1.0000000  1.0000000  1.0000000
## me.age => employed    0.7464963  0.8443495  0.7500000  0.8841082
```

6. Perception-based Logical Deduction

Unlike the more frequent fuzzy inference methods, i.e., mainly the Mamdani-Assilian [25] (and other derived conjunctive models) and the Takagi-Sugeno [63], the *perception-based logical deduction* (abbr. PbLD) uses the genuine Łukasiewicz implication. However, it is even different to the standard fuzzy relational approach to implicative rules [64] that aggregates them to a single fuzzy relation by the minimum. Let us note that the first two above-mentioned approaches as well as many others are accessible in a very rich **frbs** R package while the latter approach implementing conjunction of implications is up to the best knowledge of the authors not implemented in any R package.

The difference between inferring with standard implicative fuzzy rules and inferring with linguistic rules (containing evaluative linguistic expressions) lies in the specific inference method PbLD that applies the so-called perception – a certain algorithm that fires only particular rules based firing degree and the specificity of the antecedents, see [65].

6.1. Formal background and implementation

In this Section, we present PbLD that was introduced in [4] and in the selected fuzzy relational formal environment studied in [24]. However, note that

there is also another variant, see e.g. [5]. The first one can be called *global PbLD* and the latter one as *local PbLD*. The names, in the R-package implementation mirrored in the values `"global"` and `"local"` of the `type` argument, point to the application of the specificity ordering in the perception function that chooses particular rules to be fired. Unless explicitly stated, all what is described below has a general validity for both variants.

We briefly recall that the specificity ordering of linguistic expressions is based on the ordering of hedges under the assumption that we consider two expressions based on the same atomic expression, i.e.,$\mathcal{A}_i \leq_{\mathrm{LE}} \mathcal{A}_k$ for $\mathcal{A}_i := \langle \mathrm{hedge} \rangle_i \mathcal{A}$ and $\mathcal{A}_k := \langle \mathrm{hedge} \rangle_k \mathcal{A}$ with $\mathcal{A}$ being an atomic expression and $\langle \mathrm{hedge} \rangle_i \leq_{\mathrm{H}} \langle \mathrm{hedge} \rangle_k$ where

$$\mathrm{Ex} \leq_{\mathrm{H}} \mathrm{Si} \leq_{\mathrm{H}} \mathrm{Ve} \leq_{\mathrm{H}} \langle \mathrm{empty} \rangle \leq_{\mathrm{H}} \mathrm{ML} \leq_{\mathrm{H}} \mathrm{Ro} \leq_{\mathrm{H}} \mathrm{QR} \leq_{\mathrm{H}} \mathrm{VR} \ .$$

The specificity ordering principle, i.e., evaluative expressions of the same type are ordered according to their hedges and expressions with different atomic expressions are incomparable), is preserved also for the multiple-variable case where the same atomic expression needs to be on each axis and the ordering $\leq_{\mathrm{H}}$ has to be preserved also for all variables (hedges), otherwise again, the expressions are incomparable. In **lfl**, the specificity relation of two rules may be questioned with the `is.specific()` function. See also the `specs()` function discussed in Section 4.4.

Fuzzy rules with evaluative expressions are gathered to a fuzzy rules base that is here called the *linguistic description*, $LD = \{\mathcal{R}_1, \ldots, \mathcal{R}_K\}$:

$$\begin{aligned} \mathcal{R}_1 &:= \mathsf{IF}\ x \text{ is } \mathcal{A}_1\ \mathsf{THEN}\ y \text{ is } \mathcal{B}_1, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \mathcal{R}_K &:= \mathsf{IF}\ x \text{ is } \mathcal{A}_K\ \mathsf{THEN}\ y \text{ is } \mathcal{B}_K \end{aligned} \tag{11}$$

with x, y taking values from universes X and Y, respectively. For further purposes, we will need to define the *topic of linguistic description LD* as the set of antecedent fuzzy sets $T^{LD} = \{A_j \mid j = 1, \ldots, K\}$ where A_j models the expression $\mathcal{A}_j$.

Given a linguistic description and an input $x_0 \in X$, one may order the elements of the topic w.r.t. the input: $A_i \leq_{x_0} A_k$ for $A_i, A_k \in T^{LD}$ if

$$\text{either} \quad A_i(x_0) > A_k(x_0); \quad \text{or} \quad A_i(x_0) = A_k(x_0) \text{ and } \mathcal{A}_i \leq_{\text{LE}} \mathcal{A}_k \ .$$

The definition of $\leq_{x_0}$ above relates only to the "original" PbLD that is called *global* in this work as well as in **lfl** R-package. For the *local PbLD*, there is an additional assumption that A_i and A_k need to be of the same type (constructed with the same atomic expression), in order to be ordered $A_i \leq_{x_0} A_k$. This slight change in the definition has a strong impact on the functionality of the inference [5] which is discussed in Remark 2 below.

Again, the extension to more variables is straightforward and componentwise with the use of the minimum t-norm to aggregate the membership degrees on the individual axes:

$$A_i(x_0) = \bigwedge_{j=1}^{m} A_{ij}(x_{0j}), \quad \text{with} \quad X = X_1 \times \cdots \times X_m, \quad x_0 = (x_{01}, \ldots, x_{0m}) \ .$$

The *perception* function maps an input x to the topic of a given linguistic expression. In particular, it assigns to each input $x_0 \in X$ a subset of antecedent fuzzy sets that are minimal wrt. the ordering $\leq_{x_0}$:

$$P^{LD}(x_0) = \{A_i \in T^{LD} \mid A_i(x_0) > 0 \,\&\, \forall A_j \in T^{LD} : (A_j \leq_{x_0} A_i) \Rightarrow (\mathcal{A}_j = \mathcal{A}_i)\}.$$

Remark 2. *Only the rules with antecedents chosen by the perception function are fired that can be interpreted as firing rules with the highest firing degree and, firing the rules with the most specific antecedents, in the case of more rules fired to the degree 1. The motivation came from the situations when some extremely small objects are observed on the input and we have rules with antecedents "small" and "extremely small", see Figure 1. Indeed, extremely small objects are also small however, we may wish to result different conclusions for those small objects that are even extremely small. The above described principle for the example with antecedents "small" and "very small" is preserved even when the* local PbLD *is applied. However, in such a case, the perception also*

fires the rules with antecedents of the different type no matter that they are fired in distinct degrees. If the input has a non-zero membership degree to antecedents with expressions derived from atomic expressions "small" and "medium", the ordering $\leq_{x_0}$ determines the most fired antecedents separately for both subsets and both such rules are then chosen by perception to be fired.

The conclusion $C \in \mathcal{F}(Y)$ is then determined analogously to the standard implicative approach, i.e.,

$$C = \bigcap \{C_{i_\ell} \mid C_{i_\ell}(y) = A_{i_\ell}(x_0) \Rightarrow B_{i_\ell}(y) \ \ \& \ \ A_{i_\ell} \in P^{LD}(x_0)\}, \quad y \in Y, \quad (12)$$

where $\Rightarrow$ is the Łukasiewicz residuum and $\cap$ is the Gödel intersection.

After deriving the output C, it often needs to be defuzzified that is, we often need to find an element $y \in Y$ that represents the conclusion C in the best way. This necessity appears whenever particular numerical output is required, e.g., in automatic control, robotics, etc. This step is provided by the function `defuzz()` that is, as a function standing outside of the PbLD inference, described in Section 6.2.

The **lfl** package implements PbLD in the `pbld()` function. The inference is performed for each row of the input data object (which must be the instance of the `fsets` class, see Section 2). For instance, let us consider the R's standard dataset, `CO2`:

```
head(CO2, n=4)
```

```
## Grouped Data: uptake ~ conc | Plant
##   Plant   Type  Treatment conc uptake
## 1   Qn1 Quebec nonchilled   95   16.0
## 2   Qn1 Quebec nonchilled  175   30.4
## 3   Qn1 Quebec nonchilled  250   34.8
## 4   Qn1 Quebec nonchilled  350   37.2
```

We are going to create a rule base for predicting the `uptake` variable from the other columns of the dataset. First of all, let us convert the original data into the set of fuzzy sets. We define a custom trichotomous context for the predicted variable and leave defaults for all the other columns:

```
uptakeContext <- ctx3(7, 28.3, 46)
d <- lcut(CO2, context=list(uptake=uptakeContext))
```

Now we split the data into the `training` and `testing` part. There are 10 data rows randomly selected for the testing dataset (in real application, perhaps more sophisticated method of training/testing split may be performed e.g. by using the `createDataPartition()` function of the **caret** package [66]):

```
testingIndices <- sort(sample(seq_len(nrow(d)), 10))
print(testingIndices)
```

```
##  [1]  8 44 46 51 61 67 73 78 80 83
```

```
training <- d[-testingIndices, ]
testing <- d[testingIndices, ]
```

On the training part, the rule base is created with the `uptake` fuzzy sets as consequents (columns 39–58) and the rest as antecedents (columns 1–38):

```
rb <- searchrules(training,
                  lhs=which(vars(d) != "uptake"),
                  rhs=which(vars(d) == "uptake"),
                  minConfidence=0.5)
```

Before performing the inference with the `pbld()` function, a sampled consequent values have to be provided. The following commands produce a vector `v` of 1000 evenly distributed values from the `uptake` context and a `fsets` object `p` of fuzzy sets that appear in the consequent with membership degrees corresponding to the values from `v`. This information is needed for defuzzification, which enables to obtain a crisp value of the `uptake` predicted by the PbLD inference:

```
v <- seq(uptakeContext[1], uptakeContext[3], length.out=1000)
p <- lcut(v, name="uptake", context=uptakeContext)
```

After that, everything is prepared to run the inference on the `testing` dataset:

```
pbld(testing, rb, p, v, type="global")
```

```
##  [1] 27.26126 22.22523  7.00000 22.22523  7.00000 21.79580 19.29730
##  [8] 17.63814 27.26126 27.26126
```

Each value of the resulting vector corresponds to the result of the inference performed on a row of the `testing` input. As indicated by the `type` argument, the *global* variant of PbLD has been computed. The *local* PbLD is obtained for `type="local"`.

If the user wants to create the rule base by herself or himself, a list of character strings has to be provided that represent the rules, with first element standing for the consequent. For instance, rules

if `Plant=Mc1` **and** `conc` *is small* **then** `uptake` *is medium*

if `Type=Mississippi` **then** `uptake` *is small*

if `Treatment=nonchilled` **and** `conc` *is roughly medium* **then** `uptake` *is big*

may be evaluated with PbLD using the following code:

```
rules <- list(c("me.uptake", "Plant=Mc1", "sm.conc"),
              c("sm.uptake", "Type=Mississippi"),
              c("bi.uptake", "Treatment=nonchilled", "ro.me.conc"))
pbld(testing, rules, p, v)
```

```
##  [1]  7.00000 10.16216 10.16216 10.16216  7.00000 10.16216 10.16216
##  [8] 10.16216 10.16216 10.16216
```

6.2. Other functions related to the PbLD inference

The **lfl** package provides some additional functions related to the logical inference. These functions act primarily as building blocks for the PbLD, however, they may be sometimes useful even separately.

The `fire()` function evaluates a list of rules on a given input. The result is a vector of truth value degrees.

The `perceive()` function handles the perception, which is a central notion of the PbLD inference. From a set of rules, `perceive()` removes each rule for which another rule exists that is more specific. The specificity is determined by

calling the `is.specific()` function. In other words, for each rule $\mathcal{R}_i$ in the list of rules, it searches for another rule $\mathcal{R}_j$ such that `is.specific(Rj, Ri, ...)` returns `TRUE`. If the answer is positive then $\mathcal{R}_i$ is removed from the list. The specificity is based on the `specs()` matrix defined as an attribute of the `fsets` object, which can be created by the `lcut()` or `fcut()` function.

The `defuzz()` function performs the defuzzification. As the flavor of the output obtained by (12) leads to expressions that look like modified evaluative linguistic expressions, the use of the *Defuzzification of Evaluative Expressions* (DEE) that has been designed for these purposes seems reasonable. It is a hierarchical two-step defuzzification that firstly classifies the type of the output into three types: S^- (*"small"*), Π (*"medium"*), and S^+ (*"big"*). This step is done based on the monotonicity of the output fuzzy set membership functions, e.g., non-increasing is classified as S^- (*"small"*). In its second step, DEE calls one of the standard defuzzifications for implicative rules, *First-Of-Maxima* (FOM) is used for the S^- type, *Mean-Of-Maxima* (MOM) is used for the Π type, and *Last-Of-Maxima* (LOM) is used for the S^+ type, see Figure 1.

Function `defuzz()` takes a numeric vector of membership degrees, a numeric vector of values corresponding to the given membership degrees and a type of defuzzification (`"mom"` for *Mean-Of-Maxima*, `"fom"` for *First-Of-Maxima*, `"lom"` for *Last-Of-Maxima*, or `"dee"` for *Defuzzification of Evaluative Expressions*) and returns a defuzzified value.

As the **lfl** package is mainly focused on modeling implicative rules, it does not contain defuzzifications designed for conjuntive rules (rules of the Mamdani-Assilian type), such as the *Center-Of-Gravity* (COG) defuzzification, however, it is at disposal in **frbs** R-package.

6.3. Application – fuzzy rule-based ensemble for time series prediction

The PbLD inference methods together with linguistic fuzzy rules have been used for distinct purposes, such as classification of geological layers [67]. We will emphasize one of them that significantly favored the use of the R-package implementation compared to the LFLC commercial package (see [32]), namely,

the *Fuzzy Rule Based Ensemble* (FRBE) for time series predictions [68]. It is an ensemble of particular time series forecasting methods that is strongly motivated by distinct automatic ensembles for these purposes [69], often dealing on the meta-learning level with characteristic features of time series [70] and stimulated by the interpretable form of rules as in the case of the well-known *rule based forecasting*, see [71].

In this particular application, the authors used the **forecast** R-package [72] in order to use particular forecasting methods with automated parameter tuning, and the above-described functionalities of the **lfl** package in order to create ensemble that, based on characteristic features of the given time series, flexibly adapts the weights of the particular time series forecasting methods.

There were four methods called from **forecast** package, namely the seasonal ARIMA, exponential smoothing, Theta, and Random Walk. Using the M3 competition dataset of 2829 time series [73], it was possible to determine the accuracy of the four particular methods and to determine characteristic features, such as frequency (yearly, monthly, weekly, etc.), skewness, kurtosis, seasonality etc., for more details, see [68]. The weights were set up proportional to the determined accuracy and employed into a table similar to Table 7 where the objects were the particular time series used for the learning, features $X1, X_m$ where the determined time series features, and Y was set up to be equal to the weight determined based on the accuracy of the particular method for the particular time series. Such a table led to the generation of the linguistic description for a single time series forecasting method and the procedure was repeated for the remaining three forecasting methods. This resulted into four linguistic descriptions that, after the redundancy analysis [24] and the size reduction by the function `reduce()` were ready to determine particular method weights. Note that the DEE defuzzification was used at the end of the inference process followed by the normalization of the obtained weights.

The whole procedure was not a single experiment procedure but a creation of a stable forecasting tool that is at disposal in the **lfl** package under the function `frbe()`.

To perform the FRBE forecast, a proper time series object has to be created with assigned frequency attribute. This can be done as in the following example. The `frbe()` function may then be called with the `h` argument, which is the forecast horizon, i.e. the number of future values to be predicted:

```
myts <- ts(1:100 + rnorm(100), frequency=24)   # hourly frequency
fit <- frbe(myts, h=10)
fit$mean
```

```
##  [1] 100.5002 101.4108 102.3139 103.2133 104.0460 104.9274 105.7708
##  [8] 106.6504 107.5235 108.3899
```

The returned list contains a lot of values related to the forecast such as forecasts of the particular methods and weights. The `mean` element is a list with predicted values.

7. Applicability

This section places the **lfl** package into the perspective of the others from the practical applicability point of view. Firstly, we briefly recall the other tools and packages, secondly, we present a real case-study problem.

7.1. Other related packages

As mentioned above, **lfl** is by far not the only SW or even R-package related to the fuzzy methods. An exhaustive study on SW related to fuzzy techniques can be found in [12].

Apart from the above-mentioned LFLC [32] that served as the starting point for **lfl** package, we should name at least some of the most often used ones. For instance, **JFML** (Java Fuzzy Markup Language) [74] is a library that enables to design fuzzy rule-based systems. It allows to implement fuzzy systems for embedded systems such as Arduino or Raspberry Pi [75]. **Juzzy** [76, 77] is another SW implementation that attracted an attention, it is a Java-based toolkit that enables to deal also with Type-2 fuzzy systems.

Staying in the environment of R-packages, the above-mentioned **frbs** [14, 78] is, according to the subjective opinion of the authors, topically the closest one and it seems to be also developed to the widest application potential.

Other packages that deal with fuzzy rules are, for instance, **FisPro** [79, 80], **FuzzyToolkitUoN** [81] or **FuzzyR** [82, 83] where the first two ones focuse on Mamdani-Assilian and Takagi-Sugeno systems which makes them closer to **frbs** while the latter one focuses mainly on the well-known neurofuzzy system ANFIS.

The portfolio of problems possibly solvable by existing R-packages with fuzzy techniques is, indeed, much wider. We may find, for example, packages for fuzzy linear regression, AHP, fuzzy statistics, or fuzzy decision trees and random forests [84, 85].

If we intend to compare the **lfl** package with the most related ones from the area/technique coverage point of view, we find the following differences and similarities.

Due to the specific character of the PbLd inference and the related necessity of the use of evaluative linguistic expressions [86] based on the basic trichotomy, none of the existing packages contains these structures. Neither the predefined fuzzy sets modeling expressions created by the application of linguistic hedges to the basic triplet, nor the related implication-based inference based on a sort of "pre-selection" of rules. This fact that is not that surprising, however, up to the best knowledge of the authors, none of the existing packages contains fuzzy relational model of implicative rules. Having in mind its importance, this contribution of **lfl** to the existing tools is essential.

For the sake of the completeness of the **lfl** package and for the sake of the consistency of the implementation with the implicative rules, **lfl** contains also Mamdani-Assilian model of fuzzy rules. However, here we have to note that this approach is already employed in several packages, e.g., in **FisPro**, **FuzzyToolkitUoN**, and mainly in **frbs**. We also emphasize distinct efficient modifications and learning techniques, e.g., neural-network based fuzzy inference system ANFIS employed in **FuzzyR** and in **frbs**. In the latter package, ANFIS

is supplemented by DENFIS, hybrid neural approach HYFIS, or evolutionary approaches, e.g. by genetic fuzzy systems based on Thrift and MOGUL methods. This, jointly with other inferences (Takagi-Sugeno-Kang or Ishibuchi's method), makes **frbs** a powerfull and rich package for distinct purposes, such as regression, data-driven classification, control, or decision-making.

Owing to mention other differences, **lfl** contains complete and very recent group of fuzzy relational composition calculus including the most recent techniques [46, 30]. These structures can be used in the inference processing fuzzy inputs as well as in distinct, e.g., expert-driven, classification or decision-making tasks. We also emphasize that the implementation allows to compose partial fuzzy relations [22].

Generally, the incorporation of algebraic structures is different in **lfl**. It contains the main three residuated structures (Gödel, Goguen, and Łukasiewicz) that are used as parameters to some algorithms. Moreover, these main underlying residuated structures may be combined with six partial algebraic structures that determine the way handling of undefined or missing values.

The **lfl** package also contains fuzzy association rules mining algorithm that generates linguistic rules to which the PbLD inference is tailored. The mining algorithm works with a genuine approach that reflects membership degrees and that was used for the generation of the fuzzy rule base ensemble technique for time series forecasting [6]. This ensemble is also employed in **lfl**.

Association rules have a rich history. Before they got their name in [54], they were propoed as so-called GUHA method already in [55]. However, we are not aware of fuzzy associations rule mining algorithm implemented in any existing R-package that would be any reflecting membership degrees. We may recall, e.g., **arules** package, however, it deals with crisp association rules only.

The survey provided above brings a comparison of the R-packages that are topic-closest to **lfl** however, its goal is not to compare them from the superiority point of view. This is probably even impossible. The goal is to compare them from the area and technique coverage point of view – to show the similarities and differences. The fact is that the packages have rather complementary positions

not duplicating each other. Out of the existing packages, we would like to once more emphasize the **frbs** package that provides the richest set of methods for distinct purposes (control, regression, classification) – many of them not even attempted in **lfl**. The **lfl** package is closer to more mathematically oriented foundations in providing algebraic backgrounds and tools from the fuzzy relational calculus that may be useful for distinct purposes. In the area of inference systems, it complements existing packages with fundamental implicative rule bases and specific PbLD method, that is even followed by associations mining and FRBE for time series forecasts.

7.2. Case study I – expert classification of amphibians

This section briefly describes the application of fuzzy relational compositions to the expert classification of animal species, namely of amphibians. It nicely demonstrates the power of chaining distinct composition blocks to their extensions. We avoid the details and the experimental comparison with classical data-driven classifiers as this can be found int he very recent source [30] and we concentrate on the use of the R-package.

The problem setting is as follows. We are given 79427 training data samples (animal records) which constitutes the set X. Each data sample is a vector containing 29 values – the features constituting the set Y. These are, e.g., Boolean descriptors or distinct colors from the top or from the bottom of an animal. The task is to classify each animal into one of the 21 species – which constitute the set of classes Z.

Expert knowledge is then encoded in the matrix S declaring the "supportive" features for given species; in the matrix E declaring the excluding features for particular classes; and in matrix U declaring the unavoidable features for given classes. All these matrices (mathematically fuzzy relations on $Y \times Z$) are naturally of the dimension 29×23 and were determined by a biologist. Note, that for the testing reasons, other 79418 data samples were used in [30].

The results of chosen compositions for a fixed row from matrix R, i.e., for a particular animal record, are vectors of 23 values – each value expressing a

membership degree to a particular species. However, it is important to note that it is not a membership degree of the given animal to the given species as each animal belongs to a single species only.

For example, if we use the command:

```
a <- algebra("lukasiewicz")
compose(R, S, alg = a, type = "basic")
```

we get $(R \circ S)(x, z)$ which is the truth degree of the statement *"there exists at least a single feature that is carried by animal record x and it belongs to the features related supporting the suspicion for species z"*. As we can see from [30], this composition would lead to 100% accuracy (sensitivity) in a sense of true positive cases however, very lows accuracy (specificity) in a sense of false positive cases. Indeed, on average over the whole testing set, there were 18 vector values equal to 1 and so, 18 species were found suspicious on average.

Right for the reasons of increasing the true negative accuracy and thus, narrowing the set of suspicious species, the other extensions may be used. For example, adding the commands:

```
RE <- compose(R, E, alg = a, type = "basic")
a$pt(RS, a$n(RE))
```

leads to $(R \circ S \backslash E)(x, z)$ which is the truth degree of the statement *"there exists at least a single feature that is carried by animal record x and it belongs to the features related supporting the suspicion for species z and at the same time, x carries no feature that would be excluding for the species z"*. Such an extension directly reduces the number of suspicious species to 3.41 on average while it keeps the 100% accuracy in the true positive cases point of view, see [30].

Furthermore, we can involve also the unavoidable features:

```
RU <- compose(R, U, alg = a, type = "super")
a$pt(RS, a$n(RE), RU)
```

which gives back values $(R \circ S \backslash E)^{\triangleright U}(x, z)$ with the meaning *"there exists at least a single feature that is carried by animal record x and it belongs to the features related supporting the suspicion for species z and at the same time, x carries no feature that would be excluding for the species z, and at the same time all unavoidable features for species z are present in the record x"*. This operations reduces the number of suspicious species to 1.36 on average while still keeping the 100% accuracy in the true positive cases point of view, see [30].

Of course, using distinct quantifiers may lead to even narrower set of suspicion species but, it can easily lower the accuracy. For this particular case but also for the case of Dragonfly classification, we again refer to [30] where the readers can find also comparison with data-driven techniques applied to the same data and other accuracy measures. The details can be found again in the referred source but it is probably worth mentioning that some techniques, nnet and rpart in particular, were more efficient in narrowing the set of suspicion species to values 1.13 and 1.12., respectively. However, the price for that was that the true positive cases accuracy dropped to 90.17% and to 90.19%, respectively.

7.3. Case study II – fuzzy associations rules from the Iris data

The second case study will work with the famous Iris dataset [87, 88], which is also available in R as a built-in data frame:

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
```

```
##
##
```

We are going to search for association rules in that database: first by using the **lfl** package and then with the **arules** package. After that, we will compare the results.

First of all, we transform the `iris` data frame into fuzzy sets:

```
ldata <- lcut(iris,
              context=function(x) minmax(x, type="ctx5"),
              hedges="-")
```

This creates a data frame with each column representing linguistic expressions on original Iris data attributes. We have opted for pentachotomical contexts, which cause each numeric variable to be transformed using 5 atomic linguistic expressions "small" (`sm`), "lower medium" (`lm`), "medium" (`me`), "upper medium" (`um`), and "big" (`bi`). The `hedges="-"` argument disables all linguistic hedges. Note that such a setting is comparable to categorization of numeric variables needed in order to run **arules**, see below.

The resulting `fsets` object contains 23 fuzzy sets: five for each of four original numeric variables (sepal/petal length/width) and three dichotomic fuzzy sets for species categories.

All association rules with species in consequent and the rest of variables in antecedent can be found by executing the following command:

```
lrules <- searchrules(ldata,
                      lhs=grep("^Species=", colnames(ldata), invert=TRUE),
                      rhs=grep("^Species=", colnames(ldata)),
                      minSupport=0.05,
                      minConfidence=0.8,
                      n=0,                          # search for all rules
                      maxLength=5)
```

The minimum support threshold of a rule was set to 0.05 while the minimum confidence is required to be at least 0.8 . Such a setting resulted in 50 fuzzy association rules in total. Let us now print 8 rules with the greatest confidence:

rule	support	confidence
bi.Petal.Length ⇒ Species=virginica	0.088	1.000
bi.Petal.Width ⇒ Species=virginica	0.139	1.000
sm.Petal.Width ⇒ Species=setosa	0.317	1.000
sm.Petal.Length ⇒ Species=setosa	0.323	1.000
lm.Sepal.Length & um.Sepal.Width ⇒ Species=setosa	0.059	1.000
sm.Sepal.Length & me.Sepal.Width ⇒ Species=setosa	0.119	0.998
lm.Sepal.Width & um.Petal.Width ⇒ Species=virginica	0.067	0.997
bi.Sepal.Length ⇒ Species=virginica	0.054	0.996

Table 10: First 8 rules with the greatest confidence as found by the **lfl** package.

```
ldf <- as.data.frame(lrules)
ldf <- ldf[order(ldf$confidence, decreasing=TRUE), ]
head(ldf[, c('support', 'confidence')], n=8)
```

The resulting rules are listed in Table 10.

In **arules**, the process is as follows. First of all, the numeric data have to be transformed into categories. The standard R's `cut()` function can be used for that. The following command transforms the numeric variables into five equidistant interval categories:

```
library(arules)

adata <- data.frame(
  Sepal.Length=cut(iris$Sepal.Length, 5),
  Sepal.Width=cut(iris$Sepal.Width, 5),
  Petal.Length=cut(iris$Petal.Length, 5),
  Petal.Width=cut(iris$Petal.Width, 5),
  Species=iris$Species
)
```

Now the data may be transformed into transactions and the rules may be searched with antecedents, consequents, minimum support and confidence set as above:

```
tdata <- transactions(adata)
```

rule	support	confidence
Sepal.Length=(7.18,7.9] ⇒ Species=virginica	0.073	1.000
Petal.Length=(5.72,6.91] ⇒ Species=virginica	0.107	1.000
Petal.Width=(2.02,2.5] ⇒ Species=virginica	0.153	1.000
Petal.Width=(0.0976,0.58] ⇒ Species=setosa	0.327	1.000
Petal.Length=(0.994,2.18] ⇒ Species=setosa	0.333	1.000
Sepal.Length=(7.18,7.9] & Petal.Length=(5.72,6.91] ⇒ Species=virginica	0.073	1.000
Petal.Length=(5.72,6.91] & Petal.Width=(2.02,2.5] ⇒ Species=virginica	0.060	1.000
Sepal.Width=(2.96,3.44] & Petal.Length=(5.72,6.91] ⇒ Species=virginica	0.053	1.000

Table 11: First 8 rules with the greatest confidence as found by the **arules** package.

```
arules <- apriori(tdata,
                  parameter=list(support=0.05, confidence=0.8, target='rules'),
                  appearance=list(lhs=grep('Species=', itemLabels(tdata),
                                           invert=TRUE, value=TRUE),
                                  rhs=grep('Species=', itemLabels(tdata),
                                           value=TRUE)))
```

The **arules** package has found 64 rules, from which the 8 rules with the greatest confidence are as follows:

```
adf <- as(arules, 'data.frame')
adf <- adf[order(adf$confidence, decreasing=TRUE), ]
head(adf[, c('rules', 'support', 'confidence')], n=8)
```

See Table 11 for a list of the resulting rules.

As can be seen, both packages provide comparable results. For **arules**, the numeric data have to be categorized into crisp intervals. The **lfl** package allows to work with linguistic expressions modelled with fuzzy sets, which enables non-sharp borders between categories. The interpretability as well as the lower number of rules play in favor of **lfl** however, we do not dare to generalize too much from the latter observation as each case study could lead to a different result and the impact of the a proper categorization is essential.

8. Conclusion

This article presents the R-package **lfl** that presents tools and functions for fuzzy relational calculus and fuzzy natural logic. It provides the implementation of the most usual algebras of operations on fuzzy sets (Gödel, Goguen, and Łukasiewicz), rich variety of compositions of binary fuzzy relations, namely, the basic (circlet) one, all three Bandler-Kohout products, extensions with more binary fuzzy relations and generalized quantifiers. Of course, their combinations are allowed as well. The chosen algebra always holds the position of an argument so, all the compositions may be calculated in any of the chosen algebra. Furthermore, in order to cover various situations when the membership degree is only partially defined (inconsistency, undefinedness, missing values), distinct algebras for partial fuzzy logics, namely Bochvar, Sobociński, Kleene, Nelson, Deagonfly, and Lower estimation algebras are implemented as well.

The second very important part of the **lfl** is formed by the concepts of the fuzzy natural logic allowing to deal with linguistic fuzzy models. It allows to define fuzzy sets, especially those modeling the evaluative linguistic expressions. The particular shapes of fuzzy sets as well as their universes (contexts) may be determined from data automatically. Theses concepts are later on used in linguistic fuzzy rules that jointly with specific inference method PbLD that is also implemented in **lfl** for the approximate reasoning. An appropriate defuzzification DEE is implemented as well.

The fuzzy rules can be defined expertly as well as learned from data – using the associations mining. The **lfl** package contains original method of mining associations rules with the evaluative expressions in antecedents as well as consequents. Automatic redundancy analysis is supplied with a size reduction algorithms that additionally reduces the size in cases when the deletion of redundant rules is not sufficient. The provided algorithm is optimized to run fast in multiple threads.

The general tools are supplied by a particular yet very powerful tool for time series forecasting based on an ensemble of four forecasting methods. These

are combined into a weighted average ensemble with weights determined by linguistic fuzzy rules based on distinct characteristic features of a given time series.

Acknowledgments

Partial support of Czech Science Foundation through the grant 20-07851S is gratefully announced.

The authors would like to express their thanks to anonymous reviewers for their constructive comments.

References

[1] R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2020).
URL `https://www.R-project.org/`

[2] An introduction to R, `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`, accessed: 2021-06-16 (2021).

[3] V. Novák, A comprehensive theory of trichotomous evaluative linguistic expressions, Fuzzy Sets and Systems 159 (22) (2008) 2939–2969.

[4] V. Novák, Perception-based logical deduction, in: B. Reusch (Ed.), Computational Intelligence, Theory and Applications, Advances in Soft Computing, Springer, Berlin, 2005, pp. 237–250.

[5] A. Dvořák, M. Štěpnička, On perception-based logical deduction and its variants, in: Proceedings of the 16th World Congress of the International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijón, 2015, pp. 341–350.

[6] M. Štěpnička, M. Burda, L. Štěpničková, Fuzzy rule base ensemble generated from data by linguistic associations mining, Fuzzy Sets and Systems 285 (2016) 141–160.

[7] D. Meyer, K. Hornik, Generalized and customizable sets in R, Journal of Statistical Software 31 (2) (2009) 1–27.
URL https://www.jstatsoft.org/v31/i02/

[8] M. Gagolewski, **FuzzyNumbers** Package: Tools to Deal with Fuzzy Numbers in R (2014).
URL https://CRAN.R-project.org/package=FuzzyNumbers

[9] W. Trutschnig, A. Lubiano, **SAFD**: Statistical Analysis of Fuzzy Data, r package version 0.4 (2012).
URL https://CRAN.R-project.org/package=SAFD

[10] P. Giordani, M. B. Ferraro, **fclust**: Fuzzy Clustering, r package version 1.0.1 (2014).
URL https://CRAN.R-project.org/package=fclust

[11] M. B. Ferraro, P. Giordani, A toolbox for fuzzy clustering using the r programming language, Fuzzy Sets and Systems 279 (2015) 1–16. doi:https://doi.org/10.1016/j.fss.2015.05.001.

[12] J. Alcalá-Fdez, J. M. Alonso, A survey of fuzzy systems software: Taxonomy, current research trends, and prospects, IEEE Transactions on Fuzzy Systems 24 (1) (2016) 40–56.

[13] A. Bujard, **fugeR**: FUzzy GEnetic, A Machine Learning Algorithm to Construct Prediction Model Based on Fuzzy Logic., r package version 0.1.2 (2012).
URL https://CRAN.R-project.org/package=fugeR

[14] L. S. Riza, C. Bergmeir, F. Herrera, J. M. Benitez, **frbs**: Fuzzy Rule-based Systems for Classification and Regression Tasks, r package version 3.0-0

(2015).
URL https://CRAN.R-project.org/package=frbs

[15] W. Bandler, L. J. Kohout, Fuzzy relational products and fuzzy implication operators, in: Proceedings of the International Workshop on Fuzzy Reasoning Theory and Applications, Queen Mary College, London, 1978.

[16] R. Bělohlávek, Fuzzy Relational Systems: Foundations and Principles, Kluwer Academic, Plenum Press, Dordrecht, New York, 2002.

[17] L. Běhounek, M. Daňková, Relational compositions in fuzzy class theory, Fuzzy Sets and Systems 160 (8) (2009) 1005–1036.

[18] N. Cao, M. Štěpnička, M. Burda, A. Dolný, Excluding features in fuzzy relational compositions, Expert Syst. Appl. 81 (C) (2017) 1–11. doi:10.1016/j.eswa.2017.03.033.
URL https://doi.org/10.1016/j.eswa.2017.03.033

[19] N. Cao, M. Holčapek, M. Štěpnička, Extensions of fuzzy relational compositions based on generalized quantifiers, Fuzzy Sets and Systems 339 (2017) 73–98.

[20] L. Běhounek, V. Novák, Towards fuzzy partial logic, in: 2015 IEEE International Symposium on Multiple-Valued Logic (ISMVL), IEEE, 2015, pp. 139–144.

[21] L. Běhounek, M. Daňková, Towards fuzzy partial set theory, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Vol. 611 of Communications in Computer and Information Science, Springer, Eindhoven, The Netherlands, 2016, pp. 482–494.

[22] M. Štěpnička, N. Cao, L. Běhounek, M. Burda, A. Dolný, Missing values and dragonfly operations in fuzzy relational compositions, International Journal of Approximate Reasoning 113 (2019) 149 – 170. doi:https://doi.org/10.1016/j.ijar.2019.07.004.

[23] A. Dvořák, V. Novák, Formal theories and linguistic descriptions, Fuzzy Sets and Systems 143 (1) (2004) 169–188.

[24] A. Dvořák, M. Štěpnička, L. Štěpničková, On redundancies in systems of fuzzy/linguistic IF-THEN rules under perception-based logical deduction inference, Fuzzy Sets and Systems 277 (2015) 22 – 43. doi:http://dx.doi.org/10.1016/j.fss.2014.10.002.

[25] E. H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies 7 (1975) 1–13.

[26] H. Jones, B. Charnomordic, D. Dubois, S. Guillaume, Practical inference with systems of gradual implicative rules, IEEE Transactions on Fuzzy Systems 17 (1) (2009) 61–78.

[27] M. Štěpnička, U. Bodenhofer, M. Daňková, V. Novák, Continuity issues of the implicational interpretation of fuzzy rules, Fuzzy Sets and Systems 161 (2010) 1959–1972.

[28] R. Srikant, R. Agrawal, Mining generalized association rules, in: U. Dayal, P. M. D. Gray, S. Nishio (Eds.), VLDB, Morgan Kaufmann, 1995, pp. 407–419.

[29] J. Kupka, P. Rusnok, Regression analysis based on linguistic associations and perception-based logical deduction, Expert Systems with Applications 67 (2017) 107 – 114. doi:https://doi.org/10.1016/j.eswa.2016.08.053.

[30] M. Štěpnička, N. Cao, M. Burda, A. Dolný, S. Ožana, The concept of unavoidable features in fuzzy relational compositions, Knowledge-Based Systems 196 (2020) 105785. doi:https://doi.org/10.1016/j.knosys.2020.105785.

[31] M. Burda, Linguistic fuzzy logic in R, in: Proc. IEEE International Conference on Fuzzy Systems, Istanbul, Turkey, 2015.

[32] A. Dvořák, H. Habiballa, V. Novák, V. Pavliska, The concept of **LFLC 2000** – its specificity, realization and power of applications, Computers in Industry 51 (2003) 269–280.

[33] V. Novák, I. Perfilieva, J. Močkoř, Mathematical Principles of Fuzzy Logic, Kluwer, Massachusetts, USA, 1999.

[34] M. Baczyński, B. Jayaram, Fuzzy Implications, Springer-Verlag, 2008.

[35] J. Łukasiewicz, On 3-valued logic (1920), McCall, S.(ed.) Polish Logic (1967) 16–18.

[36] C. C. Chang, Algebraic analysis of many valued logics, Transactions American Mathematical Society 88 (1958) 476–490.

[37] D. Ciucci, D. Dubois, A map of dependencies among three-valued logics, Information Sciences 250 (2013) 162–177.

[38] N. Cao, M. Štěpnička, Compositions of partial fuzzy relations employing the lower estimation approach, in: The 10th International Conference on Knowledge and Systems Engineering (KSE 2018), IEEE, HCM, Vietnam, 2018, pp. 146–151.

[39] G. Malinowski, The Many Valued and Nonmonotonic Turn in Logic, North-Holand, Amsterdam, 2007.

[40] B. De Baets, E. Kerre, Fuzzy relational compositions, Fuzzy Sets and Systems 60 (1993) 109–120.

[41] W. Bandler, L. J. Kohout, Fuzzy power sets and fuzzy implication operators, Fuzzy Sets and Systems 4 (1980) 183–190.

[42] W. Pedrycz, Applications of fuzzy relational equations for methods of reasoning in presence of fuzzy data, Fuzzy Sets and Systems 16 (1985) 163–175.

[43] A. Di Nola, S. Sessa, W. Pedrycz, E. Sanchez, Fuzzy Relation Equations and Their Applications to Knowledge Engineering, Kluwer, Boston, 1989.

[44] M. Štěpnička, B. Jayaram, On the suitability of the Bandler-Kohout subproduct as an inference mechanism, IEEE Transactions on Fuzzy Systems 18 (2) (2010) 285–298.

[45] D. J. Mak, A fuzzy probabilistic method for medical diagnosis, IEEE Transactions on Fuzzy Systems 29 (26) (2015).

[46] N. Cao, M. Štěpnička, M. Burda, A. Dolný, Excluding features in fuzzy relational compositions, Expert Systems with Applications 81 (2017) 1–11.

[47] O. Pivert, P. Bosc, Fuzzy preference Queries to Relational Databases, Imperial College Press, London, 2012.

[48] P. Bosc, L. Liétard, O. Pivert, Sugeno fuzzy integral as a basis for the interpretation of flexible queries involving monotonic aggregates, Information Processing & Management 39 (2) (2003) 287–306.

[49] A. Dvořák, M. Holčapek, L-fuzzy quantifiers of type $\langle 1 \rangle$ determined by fuzzy measures, Fuzzy Sets and Systems 160 (23) (2009) 3425 – 3452. doi:https://doi.org/10.1016/j.fss.2009.05.010.

[50] M. Štěpnička, M. Holčapek, Fuzzy relational compositions based on generalized quantifiers, in: A. Laurent, O. Strauss, B. Bouchon-Meunier, R. R. Yager (Eds.), Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, Cham, 2014, pp. 224–233.

[51] N. Cao, M. Štěpnička, M. Holčapek, An extension of fuzzy relational compositions using generalized quantifiers, in: Proceedings of the 16th World Congress of the International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijón, 2015, pp. 49–58.

[52] E. P. Klement, R. Mesiar, E. Pap, A universal integral as common frame for choquet and sugeno integral, IEEE Transactions on Fuzzy Systems 18 (2010) 178–187.

[53] M. Mareš, Computation over Fuzzy Quantities, CRC Press, Boca Raton, Florida, 1994.

[54] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the 20th International Conference on Very Large Databases, AAAI Press, Chile, 1994, pp. 487–499.

[55] P. Hájek, The question of a general concept of the GUHA method, Kybernetika 4 (1968) 505–515.

[56] P. Hájek, T. Havránek, Mechanizing Hypothesis Formation: Mathematical Foundations for a General Theory, Springer-Verlag, Berlin/Heidelberg/New York, 1978.

[57] T. Sudkamp, Examples, counterexamples, and measuring fuzzy associations, Fuzzy Sets Systems 149 (1) (2005) 57–71.

[58] J. Kupka, I. Tomanová, Some extensions of mining of linguistic associations, Neural Network World 20 (2010) 27–44.

[59] D. H. Glass, Fuzzy confirmation measures, Fuzzy Sets Systems 159 (4) (2008) 475–490.

[60] E. P. Klement, R. Mesiar, E. Pap, Triangular Norms, Vol. 8 of Trends in Logic, Kluwer Academic Publishers, Dordrecht, 2000.

[61] G. I. Webb, Opus: An efficient admissible algorithm for unordered search, Journal of Artificial intelligence Research 3 (1995) 431–465.

[62] M. Burda, M. Štěpnička, Reduction of fuzzy rule bases driven by the coverage of training data, in: Proceedings of the 16th World Congress of the

International Fuzzy Systems Association and 9th Conference of the European Society for Fuzzy Logic and Technology (IFSA-EUSFLAT 2015), Advances in Intelligent Systems Research, Atlantic Press, Gijón, 2015.

[63] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man and Cybernetics 15 (1985) 116–132.

[64] U. Bodenhofer, M. Daňková, M. Štěpnička, V. Novák, A plea for the usefulness of the deductive interpretation of fuzzy rules in engineering applications, in: Proceedings of the 16th IEEE International Conference on Fuzzy Systems, London, 2007, pp. 1567–1572.

[65] V. Novák, I. Perfilieva, On the semantics of perception-based fuzzy logic deduction, International Journal of Intelligent Systems 19 (2004) 1007–1031.

[66] M. Kuhn, **caret**: Classification and Regression Training, r package version 6.0-86 (2020).
URL `https://CRAN.R-project.org/package=caret`

[67] V. Novák, Ancient sea level estimation, in: R. V. Demicco, G. J. Klir (Eds.), Fuzzy Logic in Geology, Academic Press, Burlington, 2004, pp. 301 – 336.

[68] M. Burda, M. Štěpnička, L. Štěpničková, Fuzzy rule-based ensemble for time series prediction: Progresses with associations mining, in: Strengthening Links between Data Analysis and Soft Computing, Vol. 315 of Advances in Intelligent Systems and Computing, Springer, Heidelberg, 2014, pp. 261–271.

[69] D. K. Barrow, S. Crone, N. Kourentzes, An evaluation of neural network ensembles and model selection for time series prediction, in: Proceedings of the 16th IEEE International Conference on Neural Networks, Barcelona, 2010, pp. 752–759.

[70] C. Lemke, B. Gabrys, Meta-learning for time series forecasting in the nn gc1 competition, in: Proceedings of the 16th IEEE International Conference on Fuzzy Systems, Barcelona, 2010, pp. 2258–2262.

[71] J. S. Armstrong, M. Adya, F. Collopy, Rule-based forecasting using judgment in time series extrapolation, in: J. S. Armstrong (Ed.), Principles of Forecasting: A Handbook for Reasearchers and Practitioners, Kluwer Academic Publishers, Boston/Dordrecht/London, 2001.

[72] R. J. Hyndman, G. Athanasopoulos, S. Razbash, D. Schmidt, Z. Zhou, Y. Khan, C. Bergmeir, **forecast**: Forecasting Functions for Time Series and Linear Models, r package version 4.06 (2013).
URL `https://CRAN.R-project.org/package=forecast`

[73] K. Ord, K. Hibon, S. Makridakis, The m3–competition, International Journal of Forecasting 16 (2000) 433–436.

[74] J. Soto-Hidalgo, J. M. Alonso, G. Acampora, J. Alcala-Fdez, JFML: A java library to design fuzzy logic systems according to the IEEE std 1855-2016, IEEE Access 6 (2018) 54952–54964.

[75] J. Soto-Hidalgo, A. Vitiello, J. M. Alonso, G. Acampora, J. Alcala-Fdez, Design of fuzzy controllers for embedded systems with jfml, International Journal of Computational Intelligence Systems 12 (2019) 204–214.

[76] P. D'Alterio, J. Garibaldi, R. John, C. Wagner, Juzzy constrained: Software for constrained interval type-2 fuzzy sets and systems in java, in: Proc. IEEE International Conference on Fuzzy Systems, Glasgow, UK, 2020.

[77] C. Wagner, Juzzy - a java based toolkit for type-2 fuzzy logic, in: Proc. IEEE Symposium Series on Computational Intelligence, Singapore, 2013, pp. 45–52.

[78] L. S. Riza, C. Bergmeir, F. Herrera, J. M. Benitez, **frbs**: Fuzzy rule-based systems for classification and regression in R, Journal of Statistical Software 65 (2015).

[79] S. Guillaume, B. Charnomordic, J.-L. Lablée, H. Jones, L. Desperben, **FisPro**: Fuzzy Inference System Design and Optimization, r package version 1.1 (2021).
URL https://CRAN.R-project.org/package=FisPro

[80] S. Guillaume, B. Charnomordic, Learning interpretable fuzzy inference systems with FisPro, Information Sciences 20 (2011) 4409–4427.

[81] C. Knott, L. Hovell, N. Karimian, **FuzzyToolkitUoN**: Type 1 Fuzzy Logic Toolkit, r package version 1.0 (2013).
URL https://CRAN.R-project.org/package=FuzzyToolkitUoN

[82] C. Chen, J. Garibaldi, T. Razak, **FuzzyR**: Fuzzy Logic Toolkit for R, r package version 2.3.1 (2021).
URL https://CRAN.R-project.org/package=FuzzyR

[83] C. Chen, T. R. Razak, J. Garibaldi, FuzzyR: An extended fuzzy logic toolbox for the r programming language, in: Proc. IEEE International Conference on Fuzzy Systems, Glasgow, UK, 2020.

[84] D. Conn, T. Ngun, C. M. Ramirez, **fuzzyforest**: Fuzzy Forests, r package version 1.0.8 (2020).
URL https://CRAN.R-project.org/package=fuzzyforest

[85] D. Conn, T. Ngun, G. Li, C. M. Ramirez, Fuzzy Forests: Extending random forest feature selection for correlated, high-dimensional data, Journal of Statistical Software 91 (9) (2019).
URL https://www.jstatsoft.org/article/view/v091i09

[86] V. Novák, A comprehensive theory of trichotomous evaluative linguistic expressions, Fuzzy Sets and Systems 159 (22) (2008) 2939–2969.

[87] R. A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (2) (1936) 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x.
URL http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x

[88] E. Anderson, The species problem in iris, Annals of the Missouri Botanical Garden 23 (3) (1936) 457–509.
URL http://www.jstor.org/stable/2394164