Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

niarules: Advancing interpretable machine learning through numerical association rule mining and 3D coral plot visualization

Iztok Fister Jr[b,a], Gerlinde Emsenhuber[d,e], Jan Hendrik Plümer[d,e], Iztok Fister[b], Andreas Holzinger[a,c,*]

[a] Human-Centered AI Lab, Institute of Forest Engineering, Department of Ecosystem Management, Climate and Biodiversity, BOKU University, Vienna, Austria
[b] Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia
[c] Institute of Human-Centered Computing, Faculty of Computer Science and Biomedical Engineering, Graz University of Technology, Graz, Austria
[d] Institute of Visual Computing, Faculty of Computer Science and Biomedical Engineering, Graz University of Technology, Graz, Austria
[e] Department of Creative Technologies, Salzburg University of Applied Sciences, Salzburg, Austria

ARTICLE INFO

Keywords:
Association rule mining
Interpretable machine learning
Numerical association rule mining
Coral plot visualization

ABSTRACT

Numerical association rule mining remains comparatively underexplored in interpretable machine learning, largely due to the challenges of handling continuous variables and the limited availability of effective visualization techniques. We introduce `niarules`, an open-source R package that provides a complete and extensible pipeline for numerical association rule mining, complemented by advanced post-processing and interactive 3D visualization. The package integrates bio-inspired optimization-based rule mining methods within a modular architecture that encompasses data preprocessing, rule mining, and visualization. A novel radial layout engine, implemented in C++, generates Coral Plots, which depict rules sharing a common consequent as radial trees. This design facilitates intuitive exploration of antecedent specificity, alongside key quality measures such as support, confidence, and lift. By combining methodological innovation with user-friendly visualization, `niarules` lowers the entry barrier to numerical association rule mining and supports the development of explainable AI systems for numerical datasets.

Code metadata

Code metadata.

Metadata	Details
Current code version	0.3.1
Permanent link to code/repository Used for this code version	https://github.com/firefly-cpp/niarules/releases/tag/0.3.1
Code ocean compute capsule	None
Legal Code License	MIT License
Code versioning system used	Git
Software code languages, tools, and services used	R, C++
Compilation requirements, Operating environments & Dependencies	R >= 4.0.0
If available, link to developer documentation/manual	https://github.com/firefly-cpp/niarules
Support email for questions	iztok@iztok-jr-fister.eu

* Corresponding author at: Human-Centered AI Lab, Institute of Forest Engineering, Department of Ecosystem Management, Climate and Biodiversity, BOKU University, Vienna, Austria.
Email address: andreas.holzinger@boku.ac.at (A. Holzinger).

https://doi.org/10.1016/j.softx.2025.102470
Received 23 September 2025; Received in revised form 18 November 2025; Accepted 28 November 2025
Available online 8 December 2025

Notation

The symbols and notations used throughout this paper are summarized in Table 1.

Table 1
List of symbols and their definitions.

Symbol	Definition
$A_{1:k}$	Ordered prefix sequence of antecedent items $(a_1, \ldots, a_k)$.
a_k	The k-th item in the ordered antecedent.
k	Prefix depth/tree level (root $k = 0$).
L	Length of the (ordered) antecedent of a rule.
K	Number of distinct RHS in a ruleset.
D	Maximum length of a rule's (ordered) antecedent for this RHS.
$r(k)$	Radial distance assigned to nodes at depth k.
(x, z)	2D layout coordinates.

The abbreviations used in this paper are summarized in Table 2.

Table 2
List of abbreviations and their definitions.

Symbol	Definition
ARM	Association rule mining
DE	Differential evolution
NARM	Numerical association rule mining
PSO	Particle swarm optimization
LHS	Left-hand size
RHS	Right-hand size
XAI	eXplainable artificial intelligence

1. Motivation and significance

Association Rule Mining (ARM) is a fundamental data mining method for uncovering interesting patterns, co-occurrences, and dependencies within structured datasets and has been studied for a long time [1]. Although extensively applied in categorical domains, especially in market basket analysis [2,3], its extension to numerical data remains non-trivial due to the continuous nature of attributes, the challenges of interval discretization and the interpretability of the rules derived [4,5].

Numerical Association Rule Mining (NARM) extends the canonical idea of ARM by supporting the mining of both categorical and numerical datasets concurrently, without the need for discretization. Most NARM approaches are based on population-based bio-inspired algorithms such as Differential Evolution (DE) [6], Particle Swarm Optimization (PSO) [7], or other metaphor-based nature-inspired algorithms [8]. Currently, the only Python-based frameworks covering the full NARM pipeline are NiaARM [9] and NiaARMTS [10].

However, there are also several GitHub repositories that support isolated parts of the NARM pipeline but lack integration. Thus, existing tools and libraries often provide only limited support for end-to-end processing of numerical association rules, and even fewer offer integrated visualization capabilities that enhance human understanding and expert-in-the-loop analytics. Incorporating the human-in-the-loop principle is particularly important, as domain experts are indispensable for validating, contextualizing, and refining the mined rules to ensure their reliability and practical utility [11–13].

To address this gap, we introduce `niarules`, a complete and extensible R pipeline for NARM with a strong focus on explainability and visual transparency. The pipeline provides a structured workflow that supports the main stages of NARM, including data preprocessing, data mining with arbitrary bio-inspired algorithms [14], and postprocessing. Beyond presenting identified association rules in tabular form, it also offers various visualization approaches. The choice of implementing the software in R is motivated by its strong ecosystem in biomedicine, biology, and other life sciences, where such approaches are widely applied [15]. Consequently, `niarules` can be easily integrated into existing R-based workflows. It is also important to mention other frameworks for pure association rule mining that are based on classical approaches, such as Apriori and Eclat, and additionally provide support for visualization. Table 3 presents an overview of various methods encompassing multiple variants of association rule mining (ARM). Some of the packages provide support for the Apriori algorithm, while others also implement additional ARM variants and support for visualization of association rules. A significant portion of the methods listed in Table 3 are designed for numerical association rule mining and are primarily based on swarm intelligence or evolutionary algorithms.

Table 3
A list of related frameworks (organized alphabetically).

Method	About	Reference
arules	State-of-the-art framework covering several association-rule mining methods	[16]
arulespy	Python package that provides the R package **arules** to the Python community	[17]
arulesViz	Support for visualization of association rules	[18]
CleverMiner	Package for enhanced association-rule mining (eARM)	[19,20]
MLxtend	Library of modules for python's data analysis and machine learning ecosystem	[21]
NiaARM	Implementation of evolutionary approaches for numerical association rule mining	[9]
NiaARMTS	Extension of **NiaARM** tailored to mining time series datasets	[10]
NiaAutoARM	AutoML-based framework for constructing and evaluating numerical ARM pipelines	[22]
LISp-Miner	Academic project supporting research and teaching of knowledge discovery in databases	[23]
PyAerial	Novel neuro-symbolic ARM algorithm for tabular datasets	[24]
SPML	Data-mining library written in Java, specialized in pattern mining	[25]
uARMSolver	Framework for numerical association rule mining in C++	[26]

In a nutshell, the developed `niarules` pipeline is presented in Fig. 1.

According to the Fig. 1, `niarules` consists of the steps, such as data preprocessing, association rule mining, and postprocessing and visualization. Each of the mentioned steps includes corresponding predefined tools that are selected when the ARM pipeline is defined. As an input, transaction datasets are provided as acquired in the particular problem domain, while the output is represented by the results of some kind of eXplainable Artificial Intelligence (XAI) methods or a tabular view of results.

The significance of `niarules` lies in its ability to support human-centered model inspection at scale, which emphasizes that inspection is not only a technical process (e.g., generating feature importance or visualizing a decision tree) but also a cognitive and interactive process shaped around human expert needs, abilities, and limitations. In this sense, it is based on the principles of human-centered software development [27].

The pipeline allows domain experts to visually explore and assess rule-based models, trace the specificity of rule antecedents, and examine how interestingness metrics such as support, confidence, and lift evolve across rules. Its modular architecture facilitates integration with external libraries and frameworks, and supports extension of the layout engine and rendering components, thus ensuring long-term adaptability and reuse in various application domains.

In application domains where human oversight, trust, and transparency are indispensable, e.g. in smart agriculture [28], smart forestry [29] or clinical decision support [30], explainability becomes a prerequisite for the adoption of machine learning systems [31–33]. Within this context, `niarules` extends the scope of numerical association rule mining beyond algorithmic rule extraction. It integrates advanced visualization techniques that allow for the clear presentation of mined

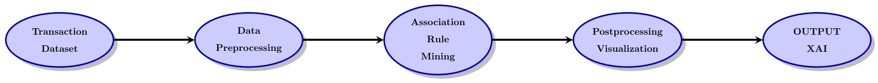

Fig. 1. Association rule mining pipeline in a nutshell.

rules and the underlying quality measures, thereby fostering user trust in the derived models. Through interactive exploration and intuitive visual metaphors such as Coral Plots, the software enables domain experts to inspect, compare, and validate rules in ways that align with their cognitive processes and decision-making needs. This human-centered design not only enhances interpretability but also supports critical reflection and responsible use of AI models in high-stakes environments. By being open-source, well-documented, and reproducible, `niarules` contributes to the broader development of explainable AI (XAI) and advances the practical application of interpretable methods for numerical datasets [34,35].

2. Software description

`niarules` is an R package for NARM built on population-based bio-inspired algorithms. The following subsections outline its software architecture in detail. The package requires several R dependencies, including `Rcpp`, `dplyr`, `rlang`, and `rgl`; for visualizations, it additionally relies on `ggplot2` and `gridExtra`. This section concludes with a detailed presentation of Coral plots, a newly proposed visualization technique for the visualization of association rules.

2.1. Software architecture

The `niarules` package is organized around the components of the NARM pipeline. Each R source file acts as a self-contained module, and together these modules implement the three major pipeline components, i.e.,:

- preprocessing: raw transaction datasets enter the pipeline and are transformed into a transaction database;
- rule mining: candidate solutions, optimized by different bio-inspired algorithms, are decoded into the association rules, and evaluated by the fitness function consisting of various interestingness measures;
- output & visualization: mined rules are formatted, exported, and visualized through visualization methods.

Table 4 summarizes the main modules, their implementation files, and their roles in the NARM pipeline. This organization reflects the sequential flow of data and rules in the NARM process, while preserving modularity, transparency, and reproducibility.

2.2. Coral plots

Inspired by the striking visualization concept of Coral Cities [37], we introduce a novel visualization technique, termed *Coral Plots*, which draws inspiration from the organic and branching morphology of coral formations [38]. This biological metaphor is adapted to the representation of association rules, where items and their relationships are depicted as interconnected branches that resemble coral growth patterns. In this layout, rules sharing a common consequent are naturally clustered, while antecedents expand outward in branching structures that visually convey their hierarchical specificity and interdependencies. By merging metaphorical design with rigorous quantitative encoding, Coral Plots provide not only an aesthetically engaging visualization but also an intuitive medium for exploring the structure of complex rule sets. The branching arrangement enhances pattern recognition, highlights rule quality measures such as support, confidence, and lift, and allows domain experts to navigate large collections of rules without being overwhelmed by traditional tabular or list-based presentations. In this way, Coral Plots bridge interpretability and usability, making numerical association rule mining results more accessible, explainable, and actionable in practical decision-making contexts.

We propose a graph-based visualization in which all rules sharing the same **RHS** are arranged as a radial "coral". Each coral aggregates rules into *rule-trees* constructed from ordered antecedents, where the order is induced by ranking items by the quality of their single-item rule $item \rightarrow RHS$. In this construction, each non-root node corresponds to a unique antecedent prefix $A_{1:k}$ but is labeled by a_k for readability; each edge represents adding one item to the current prefix $A_{1:k-1}$. As one moves outward from the center, k increases and rules become more specific. Visual encodings on edges (and optionally on nodes) summarize how support, confidence, and lift change as items are added to the antecedent.

Rule-trees and prefix construction. Within each coral, we order **LHS** items by *significance*. By default this is derived automatically from single-item rules: we collect all rules with an antecedent length of 1 for the given consequent, extract their metrics, and rank items in descending order by a user-selected metric, i.e., confidence, support, or lift. This ranking is used to sort the items inside every **LHS** before we construct the prefixes, ensuring that stronger items appear earlier in the prefix and that prefixes are comparable across rules. The sorting is a user-supplied parameter. If no single-item rules are present, the ranking falls back to alphabetic ordering. For each rule, we generate all prefixes $A_{1:1}, A_{1:2}, \ldots, A_{1:L}$ using the ranked order and *aggregate* metrics over identical prefixes by taking the mean (support, confidence, lift) averaged over all rules whose ordered antecedent begins with that prefix. Prefixes serve as the nodes of the rule-tree, with edges connecting $A_{1:k-1} \rightarrow A_{1:k}$.

Node identity and merging. A node is identified by its entire ordered prefix $A_{1:k}$, even though we display only the last item a_k. Two

Table 4
Module overview of `niarules`, grouped by pipeline components.

Pipeline component	Module (file)	Role (key features)
Preprocessing	Dataset (`dataset.R`)	Reads CSV/frames; optional timestamp parsing; infers feature metadata (type, bounds, categories); computes dimension of the problem for optimizer.
Rule mining	Rule construction (`rule.R`)	Decodes real-valued candidates into concrete rules; assembles antecedent/consequent; optional time-window mapping.
	Evaluation (`evaluate.R`)	Computes support & confidence; aggregates fitness; validates feasibility; evaluates on full data or time intervals.
	Optimization: DE/PSO (`de.R`, `pso.R`)	Search over rule space via DE or PSO [36]; uniform interface to `evaluate()`.
Output & visualization	Output (`output.R`)	Formats human-readable rule summaries; exports CSV; bridges mining and visualization with stable textual outputs.
	Visualization (`narmviz.R`, `coral_plot_visualization.R`)	*narmviz*: implementation of NarmViz method; *coral*: builds coral layouts from parsed rules;

rules therefore contribute to the same node only if, after sorting their antecedents, their ordered prefixes are identical. e.g., if two rules differ in their first item after ordering, e.g., (item1, item2, item3) vs. (item2, item1, item4) they will produce two distinct branches in the resulting plot. As a result, the same item label may appear multiple times at the same depth when it is reached through different parent prefixes. We deliberately do not apply a further unification step that would merge nodes solely on the basis of sharing the same item label at a given depth, as doing so would mix branches originating from different antecedent contexts and would break the one-to-one correspondence between prefixes and nodes.

Multi-item consequents. If a rule's **RHS** contains multiple items, the algorithm assigns a *composite* root ID and records its components. The layout then uses one logical root prefix for the coral and emits one *visible* center node per atomic consequent at the same coordinates. This disambiguates multi-item consequents without changing the tree geometry.

Layout algorithm. Given all prefixes for an **RHS**, the layout algorithm first assigns a polar angle θ and a radial distance $r(k)$ to each one and then places nodes in (x, z) on a grid of coral centers. The y coordinate is set to 0 and can be configured separately during the styling/rendering step.

Radial distance. Let $D = max\, L$ be the maximum prefix length in the coral. Nodes at depth k (root $k = 0$) are placed on concentric circles with distance $r(k) = R_{max} * \frac{k}{D}$, i.e., growing approximately linearly with depth up to a configurable maximum radius R_{max}.

Polar angles are distributed top-down in proportion to leaf counts, in order to prevent dense subtrees from being visually cramped. Concretely, the algorithm (1) builds the set of unique prefixes with aggregated metrics, (2) computes the *children* relation, and (3) counts *leaves* under every node via a depth-first search from the *root prefix*. Sibling lists are sorted deterministically to stabilize the layout. At step $k > 0$, each child's angular span is a fraction of its parent's span equal to $\frac{\text{leaves under child}}{\sum \text{leaves under siblings}}$. A node's drawing angle is the midpoint of its span.

Node coordinates are expressed as:

$$(x, z) = (x_{center}, z_{center}) + r(k) * (\cos\theta, \sin\theta), \quad (1)$$

where (x_{center}, z_{center}) is the coral's cell center on the global grid, θ is the node's midpoint angle, and $r(k)$ is the step-to-radial-distance mapping. The builder emits one center node per **RHS** item in the case of a multi-item **RHS**, or a single center when the **RHS** is atomic and one node per non-root path.

For every prefix $A_{1:k}$ with $k \geq 1$ an edge connects $A_{1:k-1} \to A_{1:k}$. Each edge carries the *aggregated* metrics for that prefix (mean support, confidence, and lift across contributing rules), enabling direct mapping to styling attributes during rendering. Edge coordinates are taken from the parent/child nodes' (x, z); vertical positions y are set during rendering.

Multiple corals on a grid. To display many coral plots at once, we place them on a near-square grid: With K distinct (non-empty) RHS combinations, the grid side length is $\lceil\sqrt(K)\rceil$. Coral centers are located at half-integer coordinates inside that grid.

Layout determinism. The layout is deterministic given the input rules and the selected **LHS**-sorting metric: The sibling lists are sorted, and centers are placed in a fixed scan order across grid cells.

Rendering. The renderer takes the layout's `nodes` and `edges`, draws an optional ground grid, renders edges as 3D segments and nodes as spheres, and provides interactive camera controls. Edge width, color, and optional alpha are mapped from a chosen metric after normalization and a selectable transform (linear/sqrt/log). Node color can be keyed by base feature, item ID, or edge-derived summaries. Thus, a node elevation y can be set proportional to normalized radial distance for gentle relief. To reduce clutter, labels are optional (interval, item, or short-interval labels), and only a subset of non-root labels is drawn by default. In the case of multi-item **RHS**, when multiple center nodes coincide, small vertical offsets are added to separate them.

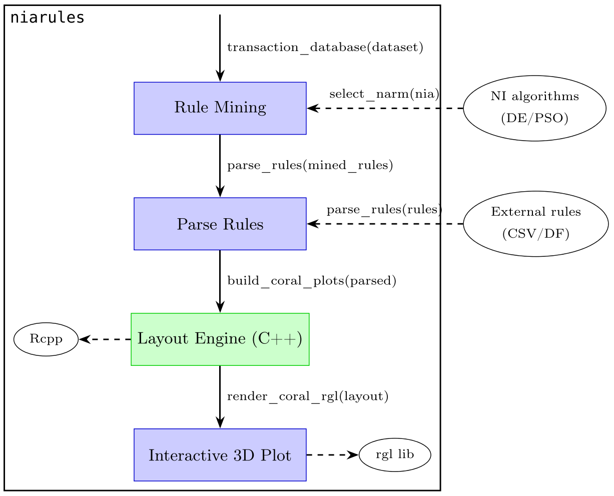

Fig. 2. Architecture of the Coral Plots visualization: (1) rule mining using NI algorithm as selected by `select_narm()`, 2) `parse_rules()` normalizes either rules mined by bio-inspired algorithms or imported from external sources (like Comma-Separated Values (CSV) or data frame (DF)), (3) `build_coral_plots()` drives the C++ layout engine via Rcpp, and (4) `render_coral_rgl()` displays an interactive 3D plot.

2.2.1. The use of coral plots in niarules

The results of the `niarules` ARM pipeline can be presented using the Coral Plot visualization obtained in four loosely coupled steps, as follows (Fig. 2): (1) rule mining, (2) rule parsing and normalization, (3) layout construction, and (4) interactive 3D rendering. Clear data contracts between the steps keep the system modular and extensible.

Inputs and overall data flow. Rules can be supplied either from the package's mining interface (e.g., `select_narm()`) or from external sources via CSV or data frame. Both inputs are funneled through `parse_rules()`, which normalizes identifiers and metadata and outputs a layout-agnostic representation consumed by the C++ layout engine. The C++ step returns a compact layout – nodes and edges with geometry and attached metrics – which the rgl renderer turns into an interactive plot. The main user-facing functions are:

- `parse_rules(rules)`
- `build_coral_plots(parsed, …)`
- `render_coral_rgl(layout, …)`

Parsing and normalization. `parse_rules()` constructs a consistent internal model, consisting of:

1. Items: Identified by 0-based integer IDs and per-item metadata such as base feature, type (numeric/categorical), numeric bounds, categorical value, and preformatted interval labels.
2. Rules: For each rule, the following data are stored: the split, antecedent and consequent items, and the interestingness measures, i.e., support, confidence, and lift.
3. Validation: Referring to the splitting of LHS/RHS tokens, trimming, type checking for numeric intervals and relations.

This stage is intentionally layout-agnostic and retains only what the downstream stages need. **Metric naming.** Rules mined by `niarules` often report a metric called `Fitness`; in our visualization pipeline we interpret it as a lift-like ratio and normalize the column name to `lift`. External sources that already provide `lift` are accepted unchanged.

R → C++ bridging. `build_coral_plots()` prepares parameters and hands the normalized structures to the C++ layout engine via `Rcpp`. It also handles two R-side responsibilities, in other words:

- Multi-item consequents: Builds a composite root identifier for each distinct consequent item and retains the component IDs so that the renderer can draw one visible center node per unique item at the same location.
- Global plot placement: Computes a near-square grid for placing multiple corals, i.e., with K distinct (non-empty) consequents, the grid size is $\lceil\sqrt{(K)}\rceil$.

C++: layout engine (core). Given a fixed consequent, the engine constructs *rule-trees* whose non-root nodes are antecedent prefixes $A_{1:k}$. It orders LHS items per consequent using single-item metrics, generates all prefixes, and aggregates metrics for identical prefixes. Then, nodes representing the prefixes and edges representing their connections are built, as:

- Nodes: The fields include coordinates, depth, angular span, item ID/labels, and fixed geometric sphere size per prefix and one center node per consequent item.
- Edges: To each edge, the aggregated metrics (mean support, confidence, lift) and start/end coordinates are attached.

The layout is deterministic given the rules and the chosen ordering metric.

R: interactive rendering (`rgl`). `render_coral_rgl()` consumes the layout and handles all visual encodings and interactivity:

- Encodings: Maps edge metrics to width/color/alpha, with normalization and optional linear/sqrt/log transforms. Colors nodes by base feature or item; optionally elevates nodes in y proportional to normalized radial distance.
- Labels and clutter control: Interval and item labels are optional and throttled; overlapping center nodes are vertically offset by small stems.
- Interaction: The effects, like zoom, pan, and rotate are applied for interaction. In addition a “return data” mode used for plotting is included that exposes the per-edge/node aesthetics.

Extensibility. The architecture isolates concerns, such as parsing that can accommodate new item syntax, swapping in alternative angular weightings (e.g., by summed support rather than leaf counts) or node-level metrics that can apply in the C++ step, rendering that can add new encoding or export formats without changing the layout engine.

3. Illustrative examples

3.1. Canonical rule mining

The first example presenting a basic workflow for mining numerical association rules is shown in the code snippet Listing 1. A typical workflow starts with reading a dataset and extracting feature metadata, which defines the search space for candidate rules. Then, bio-inspired optimization algorithms, such as DE or PSO, explore this space to generate and evaluate candidate rules using a fitness function consisting of interestingness measures and weights. Finally, mined association rules are collected, printed in human-readable form, and exported to CSV for further analysis or visualization (e.g., with `narmviz` or Coral plots).

```r
library("niarules")

dataset  <- "Abalone.csv"
data     <- read_dataset(dataset)
features <- extract_feature_info(data)
dim      <- problem_dimension(features, is_time_series = FALSE)

de <- differential_evolution(
  d = dim, np = 30, f = 0.5, cr = 0.9, nfes = 1000,
  features = features, data = data, is_time_series = FALSE
)

print_association_rules(de$arules, is_time_series = FALSE)
write_association_rules_to_csv(de$arules, "Rules.csv", is_time_series = FALSE)
```

Listing 1. `niarules` example.

```r
library(niarules)
library(readr)

df <- readr::read_csv("rules.csv", show_col_types = FALSE)

parsed <- niarules::parse_rules(df)
layout <- niarules::build_coral_plots(parsed)

feat_levels <- sort(unique(na.omit(layout$nodes$feature)))
pal_nodes <- grDevices::hcl.colors(length(feat_levels), "Purple-Yellow")
names(pal_nodes) <- feat_levels

pal_edges <- c("#2166ac", "#f0f0f0", "#b2182b")

out <- niarules::render_coral_rgl(
  nodes = layout$nodes, edges = layout$edges,
  grid_size = layout$grid_size,
  legend = TRUE, y_scale = 0.2,
  node_color_by = "type", node_gradient = pal_nodes,
  node_gradient_map = "even",
  edge_width_metric = "support", edge_width_transform = "sqrt",
  edge_width_range = c(2, 8),
  edge_color_metric = "lift", edge_color_transform = "log",
  edge_gradient = pal_edges,
  edge_alpha_metric = "confidence", edge_alpha_transform = "linear",
  edge_alpha_range = c(0.35, 1)
)
```

Listing 2. Rendering a coral plot from a prepared rule table.

3.2. Coral plots visualization

Having introduced coral plots and the layout in Section 2.2, we now use them as a compact summary of mined association rules. After mining we convert rules to the plotting model with `niarules::parse_rules()`, build the layout with `build_coral_plots()`, and render with `render_coral_rgl()`. An example of this rendering pipeline is shown in Listing 2. Fig. 3 presents two illustrative panels for reference and discussion: rules mined with `niarules::differential_evolution` (left) and with `arules::apriori` (right). Both use a common legend discretization and the same RHS target (`Rings = 9`); in the DE panel we snap only Length, Diameter, and Height to this grid, whereas Apriori is fully discretized before mining.

Visual encodings are held constant across panels: node hue denotes the feature; edge width and color encode support and lift, respectively. Edge color uses the same log-centered diverging scale in both panels (blue < 1, gray ≈ 1, red > 1), so the neutral point corresponds to lift $= 1$. Additionally, edge opacity could be used to encode confidence by making low-confidence edges fade into the background; However, we limited the plots to two edge encodings to avoid overloading them with information. The legend lists numeric bin intervals per feature.

How to read the plot.

- Each path from the center to a leaf spells a rule’s LHS in a fixed, deterministic order. When single-item scores are available the order follows the chosen metric (e.g., decreasing confidence with the RHS); otherwise an alphabetical fallback by feature name is used. In Fig. 3 the LHS are ordered alphabetically by feature.
- Numbers inside nodes are bin indices for numeric features; categorical nodes (e.g., `Sex`) show the level.

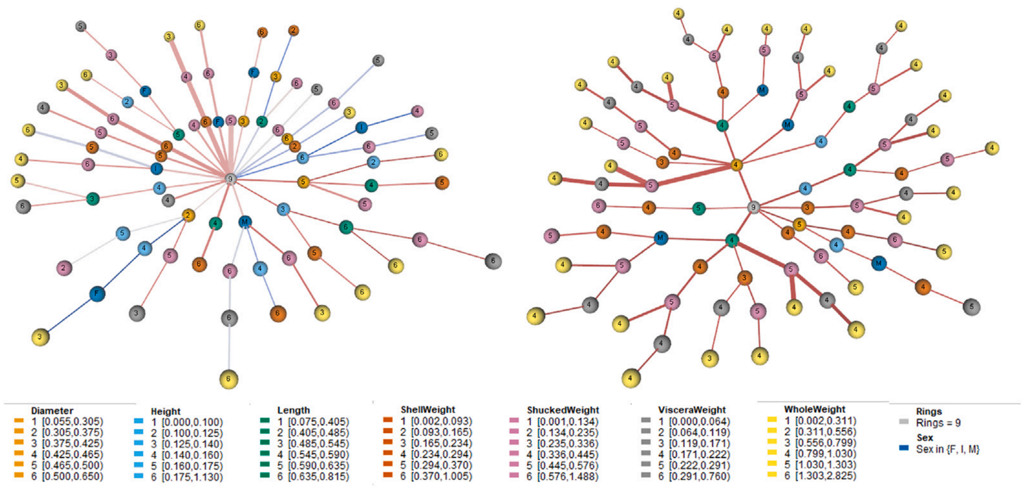

Fig. 3. Coral visualizations of association-rule sets for the same RHS target (Abalone dataset; `Rings = 9`). Left: rules mined with `niarules::differential_evolution`; after mining, we snapped length, diameter, and height to the legend's discretization grid to preserve branch structure (other numeric features retain their mined cut points). Right: rules mined with `arules::apriori` on a frequency-discretized dataset. Both panels use identical visual encodings and a shared log-centered lift color scale (gray edges show a lift of ≈ 1, blue of < 1, and red of > 1).

- Thicker edges indicate higher support for that partial conjunction; color reflects departure from baseline via (log-)lift (neutral at 1).
- Repeated hues near the center indicate features that frequently appear first under the chosen ordering; longer branches show commonly co-occurring conditions.

In the DE panel, non-geometric features retain their mined thresholds (only Length/Diameter/Height are snapped), which preserves branch structure while keeping visual encodings comparable to Apriori. The example is illustrative: it shows that the same pipeline can render rule sets from different miners without API changes and provides a compact overview of which feature bins appear and how they combine.

Coral plot interpretation. Fig. 3 compares two coral visualizations of association-rule sets targeting Rings = 9 in the Abalone dataset. The panels illustrate how the same visualization algorithm summarizes rule structures from different mining algorithms. In both cases, features are ordered alphabetically because no single-feature metrics were available; thus, the inner nodes (Length, Diameter, Height) reflect layout convention rather than statistical priority.

The differential-evolution panel (Fig. 3, Left) shows a relatively compact coral with few branches. This mirrors the algorithm's search strategy: DE optimizes numeric thresholds in continuous space and retains only a limited set of high-fitness rules. Since we discretized only some item categories after the mining, few rules share identical antecedent prefixes, so the graph forms a small number of strong branches. Edge colors and widths show a mixture of high-lift, high-support paths (thick, red) as well as weaker, low-support ones (thin, gray or blue). Lift often decreases with rule length, reflecting that more specific antecedents describe smaller, less generalizable subgroups.

In comparison, because Apriori (Fig. 3, Right) operates on a fully discretized dataset, it enumerates many overlapping categorical combinations, producing a branched out structure with numerous shared prefixes. Most of its edges are red, indicating uniformly high lift, which is a consequence of frequent-itemset mining on coarse bins. Support varies across branches.

Taken together, the two corals demonstrate how the visualization exposes algorithmic and structural differences in mined rule sets. Even at a reduced scale, users can infer which features often combine, how rule strength varies, and how the search behavior of different miners shapes the resulting rule space.

4. Impact

Association rule mining is a method proposed more than 30 years ago, yet it remains highly relevant today [39]. Although many methods have been developed for this task, important challenges still persist. With `niarules`, we aim to broaden the applicability of NARM within the R ecosystem, which is widely used in biomedicine and other life sciences, thereby enabling researchers to easily integrate the software into their workflows. As much of today's real-world data comes in the form of time series, `niarules` efficiently handles such datasets and supports rule mining with recently proposed methods.

The impact of `niarules` lies not only in providing the first extensible R framework for NARM, but also in fostering explainability through integrated visualization tools. This contributes to greater transparency, trust, and usability of mined rules in sensitive domains such as healthcare, smart agriculture, and environmental sciences. Furthermore, the modular design encourages community-driven extensions, ensuring that the software can evolve alongside emerging methods. As such, `niarules` has the potential to become a cornerstone tool for advancing both methodological research and practical applications in numerical data mining.

We believe that niarules can also be used in conjunction with other frameworks. In general, this is possible provided that the external package's output can be converted into a compatible format (e.g., standardized rule representation or CSV structure). The level of integration effort depends mainly on the data format used by the external package, but in most cases, only minimal preprocessing would be required.

5. Conclusion and future outlook

In this paper, we present `niarules`, an R package for mining numerical association rules with strong support for visualization in a context of XAI. The software follows a modular design, where the pipeline is organized into distinct components, and can be easily extended with additional modules in the future. In `niarules`, numerical association rule mining is formulated as an optimization problem and solved using population-based bio-inspired algorithms. Beyond mining classical datasets, the package also supports mining the time series data.

We further introduce a novel visualization technique for association rules, termed coral plots, which are inspired by the structure of corals. Two illustrative examples demonstrate the use of the framework, while more comprehensive information is provided in the accompanying documentation.

For the future work, `niarules` should undergo rigorous testing to prevent potential bugs and expand its reliability. Additional visualization methods will be incorporated to further enhance explainability, and new bio-inspired algorithms will be integrated into the pipeline. Finally, the ability to efficiently handle large-scale datasets and evaluate computational performance represents an important direction for further development.

CRediT authorship contribution statement

Iztok Fister Jr: Writing – review & editing, Writing – original draft, Visualization, Supervision, Software, Project administration, Methodology, Investigation, Conceptualization. **Gerlinde Emsenhuber:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation. **Jan Hendrik Plümer:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis. **Iztok Fister:** Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis. **Andreas Holzinger:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used language tools such as Grammarly and ChatGPT in order to improve the article's readability. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was funded in part by the Austrian Science Fund (FWF) https://doi.org/10.55776/P32554. For open access purposes, the authors have applied a CC BY public copyright license to any author accepted manuscript version arising from this submission. This publication reflects only the authors' view and the funder is not responsible for any use that may be made of the information it contains.

References

[1] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. ACM SIGMOD Record 1993;22(2):207–16. https://doi.org/10.1145/170036.170072
[2] Kaur M, Kang S. Market basket analysis: identify the changing trends of market data using association rule mining. Procedia Comput Sci 2016;85:78–85. https://doi.org/10.1016/j.procs.2016.05.180
[3] Ünvan YA. Market basket analysis with association rules. Commun Stat Theory Methods 2021;50(7):1615–28. https://doi.org/10.1080/03610926.2020.1716255
[4] Fister Jr I, Fister I, Fister D, Podgorelec V, Salcedo-Sanz S. A comprehensive review of visualization methods for association rule mining: taxonomy, challenges, open problems and future ideas. Expert Syst Appl 2023;233:120901. https://doi.org/10.1016/j.eswa.2023.120901
[5] Varol Altay E, Alatas B. Performance analysis of multi-objective artificial intelligence optimization algorithms in numerical association rule mining. J Ambient Intell Human Comput 2020;11(8):3449–69. https://doi.org/10.1007/s12652-019-01540-7
[6] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 1997;11(4):341–59. https://doi.org/10.1023/A:1008202821328
[7] Kennedy J, Eberhart R. Particle swarm optimization. In: Neural Net, 1995. Proceedings., IEEE International Conference on, 4. IEEE; 1995. p. 1942–8.
[8] Yang X-S. A new metaheuristic bat-inspired algorithm. In González JR, Pelta DA, Cruz C, Terrazas G, Krasnogor N, editors. Nature inspired cooperative strategies for optimization (NICSO 2010). Berlin, Heidelberg: Springer Berlin Heidelberg; 2010. pp. 65–74. https://doi.org/10.1007/978-3-642-12538-6_6
[9] Stupan Ž, Fister I. Niaarm: a minimalistic framework for numerical association rule mining. J Open Source Softw 2022;7(77):4448.
[10] Fister Jr I, Salcedo-Sanz S, Alexandre-Cortizo E, Novak D, Fister I, Podgorelec V, Gorenjak M. Toward explainable time-series numerical association rule mining: a case study in smart-agriculture. Mathematics 2025;13(13):2122.
[11] Mosqueira-Rey E, Hernández-Pereira E, Alonso-Ríos D, Bobes-Bascaran J, Fernandez-Leal A. Human-in-the-loop machine learning: a state of the art. Artif Intell Rev 2023;56(4):3005–54. https://doi.org/10.1007/s10462-022-10246-w
[12] Mosqueira-Rey E, Hernández-Pereira E, Bobes-Bascarán J, Alonso-Ríos D, Pérez-Sánchez A, Fernández-Leal A, Moret-Bonillo V, Vidal-Ínsua Y, Vázquez-Rivera F. Addressing the data bottleneck in medical deep learning models using a human-in-the-loop machine learning approach. Neural Comput Appl 2024;36(5):2597–616. https://doi.org/10.1007/s00521-023-09197-2
[13] Hausleitner C, Mueller H, Holzinger A, Pfeifer B. Collaborative weighting in federated graph neural networks for disease classification with the human-in-the-loop. Nat Sci Rep 2024;14(21839):1–10. https://doi.org/10.1038/s41598-024-72748-7
[14] Freitas AA. A survey of evolutionary algorithms for data mining and knowledge discovery. In Ghosh A, Tsutsui S, editors. Advances in evolutionary computing. Springer; 2003. pp. 819–45. https://doi.org/10.1007/978-3-642-18965-4_33
[15] Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, et al. Bioconductor: open software development for computational biology and bioinformatics. Genome Biol 2004;5(10):R80.
[16] Hahsler M, Grün B, Hornik K. Arules-a computational environment for mining association rules and frequent item sets. J Stat Softw 2005;14:1–25.
[17] Hahsler M. Arulespy: exploring association rules and frequent itemsets in python. arXiv:2305.15263. 2023, [preprint].
[18] Hahsler M, Chelluboina S. Visualizing association rules: introduction to the r-extension package Arulesviz. R Project Module 2011;6:223–38.
[19] Little Big Company, s.r.o, Masa P. Cleverminer: enhanced association rule mining library, python package; license GNU GPLv3 2025. https://pypi.org/project/cleverminer/.
[20] Máša P, Rauch J. A novel algorithm weighting different importance of classes in enhanced association rules. Knowl Based Syst 2024;294:111741.
[21] Raschka S. Mlxtend: providing machine learning and data science utilities and extensions to python's scientific computing stack. J Open Source Softw 2018;3(24):638.
[22] Mlakar U, Fister Jr I, Fister I. Niaautoarm: automated framework for constructing and evaluating association rule mining pipelines. Mathematics 2025;13(12):1957.
[23] Simunek M. Academic KDD project Lisp-Miner. In: Intelligent systems design and applications. Springer; 2003. p. 263–72.
[24] Karabulut E, Groth P, Degeler V. Pyaerial: scalable association rule mining from tabular data. SoftwareX 2025;31:102341.
[25] Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu C-W, Tseng VS, et al. Spmf: a Java open-source pattern mining library. J Mach Learn Res 2014;15(1):3389–93.
[26] Fister I, Fister Jr I. uarmsolver: a framework for association rule mining. arXiv:2010.10884 [preprint]. 2020.
[27] Peischl B, Ferk M, Holzinger A. The fine art of user-centered software development. Softw Qual J 2015;23(3):509–36. https://doi.org/10.1007/s11219-014-9239-1
[28] Holzinger A, Fister Jr. I, Fister I, Kaul H-P, Asseng S. Human-centered AI in smart farming: towards agriculture 5.0. IEEE Access 2024;12:62199–214. https://doi.org/10.1109/ACCESS.2024.3395532
[29] Holzinger A, Lukac N, Rozajac D, Johnston E, Kocic V, Hörl B, Gollob C, Nothdurft A, Stampfer K, Schweng S, Del-Ser J. Enhancing trust in automated 3D point cloud data interpretation through explainable counterfactuals. Inf Fusion 2025;119:103032. https://doi.org/10.1016/j.inffus.2025.103032
[30] Plass M, Kargl M, Nitsche P, Jungwirth E, Holzinger A, Müller H. Understanding and explaining diagnostic paths: toward augmented decision making. IEEE Comput Graph Appl 2022;42(6):47–57. https://doi.org/10.1109/MCG.2022.3197957
[31] Afroogh S, Akbari A, Malone E, Kargar M, Alambeigi H. Trust in AI: progress, challenges, and future directions. Humanit Soc Sci Commun 2024;11(1):1–30.
[32] de Brito Duarte R, Correia F, Arriaga P, Paiva A. AI trust: can explainable AI enhance warranted trust? Hum Behav Emerg Technol 2023;(1):4637678.
[33] Ma W, Zhang M, Cao Y, Jin W, Wang C, Liu Y, Ma S, Ren X. Jointly learning explainable rules for recommendation with knowledge graph. In: Liu L, White R, editors. The world wide web conference (19). Association for Computing Machinery (ACM); 2019. p. 1210–21. https://doi.org/10.1145/3308558.3313607
[34] Linardatos P, Papastefanopoulos V, Kotsiantis S. Explainable AI: a review of machine learning interpretability methods. Entropy 2020;23(1):18. https://doi.org/10.3390/e23010018
[35] Cabitza F, Campagner A, Malgieri G, Natali C, Schneeberger D, Stoeger K, Holzinger A. Quod erat demonstrandum-towards a typology of the concept of explanation for the design of explainable AI. Expert Syst Appl 2023;213(3):118888. https://doi.org/10.1016/j.eswa.2022.118888
[36] Anderson RL. Recent advances in finding best operating conditions. J Am Stat Assoc 1953;48(264):789–98. http://www.jstor.org/stable/2281072.
[37] Taylor C. Coral cities: an ITO Design Lab concept, in: medium Sep 28; 2018. https://medium.com/data-science/coral-cities-an-ito-design-lab-concept-c01a3f4a2722 Accessed: [23 Sept 2025].
[38] Zawada KJA, Dornelas M, Madin JS. Quantifying coral morphology. Coral Reefs 2019;38(6):1281–92. https://doi.org/10.1007/s00338-019-01842-4
[39] Wu X, Kumar V, Ross Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, et al. Top 10 algorithms in data mining. Knowl Inf Syst 2008;14(1):1–37.