Black Sea Journal of Engineering and Science
doi: 10.34248/bsengineering.1951627

Open Access Journal
e-ISSN: 2619 – 8991

Research Article
Volume 9 - Issue 4: 1803-1818 / July 2026

MCDABENCH: AN R PACKAGE FOR BENCHMARKING MULTI-CRITERIA DECISION MAKING METHODS AND SCENARIOS

Cagatay CEBECI[1]*

[1]*Osmaniye Korkutata University, Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, 80000, Osmaniye, Türkiye*

Abstract: This paper introduces mcdabench, an R package for Multi-Criteria Decision Making (MCDM), which contains several useful tools, including data normalization, objective weighting, implementation of various well-known methods and their comparisons, sensitivity and stability analyses, benchmarking, aggregation, and visualization. The methods amount to 38 widely used MCDM methods from the literature. Sensitivity and stability analyses utilize measures such as the rank stability index and rank volatility. These features are enhanced by two novel contributions: Generalized gradual-weighting and random weight perturbations, both of which are important for evaluating the robustness of MCDM methods. The benchmark function applies the MCDM methods and returns a combined ranking matrix, which the comparison function then uses to calculate pairwise comparison metrics and perform statistical tests. The statistical tests include Spearman's rank correlation, Sałabun-Urbaniak similarities, and novel contributions such as Shannon's entropy-based permutation tests and Jensen-Shannon Divergence-based bootstrap tests. The aggregation function provides a variety of methods, such as Borda Count, Kemeny-Young, Copeland, and the Markov method, to generate a final ranking. In addition, a useful "top-k" technique is used to focus on the top-k alternatives. The mcdabench package also provides built-in visualization functions to present analysis results in clean, detailed demonstrations.

Keywords: Multi-criteria decision making, MCDM analysis, Benchmarking, R package, Industrial robots, Robotic manipulators

***Corresponding author:** Osmaniye Korkutata University, Faculty of Engineering and Natural Sciences, Department of Electrical and Electronics Engineering, 80000, Osmaniye, Türkiye
E mail: cagataycebeci@osmaniye.edu.tr
Cagatay CEBECI https://orcid.org/0000-0003-2644-1261

Received: May 15, 2026
Accepted: June 23, 2026
Published: July 15, 2026

Cite as: Cebeci, C. (2026). Mcdabench: An R package for benchmarking multi-criteria decision making methods and scenarios. *Black Sea Journal of Engineering and Science*, *9*(4), 1803-1818.

1. Introduction

Multi-Criteria Decision Making (MCDM), also known as Multi-Criteria Decision Analysis (MCDA), is an applied discipline that solves decision-making problems by evaluating alternatives using multiple, often conflicting, criteria (Kumar and Pamučar, 2025). MCDM is used in many domains, including engineering, operations research, economics, business, medicine, healthcare, agriculture, and other life sciences (Zavadskas and Turskis, 2011; Stojčić et al., 2019; Kumar, 2025; Taherdoost and Madanchian, 2023; Cebeci, 2026). Following the introduction of the Analytic Hierarchy Process (AHP) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), numerous MCDM methods have been developed. Surveys indicate that there are now over 100 distinct MCDM approaches, each characterized by specific advantages and disadvantages. A common way to distinguish these approaches is to classify them as either function-based or function-free (Taherdoost and Madanchian, 2023).

Although a large number of MCDM methods have been proposed in the literature, software tools to support their implementation and analysis remain relatively few. Nevertheless, a number of libraries and packages developed in Python, R, and Julia are available for MCDM research and applications, useful to researchers and practitioners alike. Among these, pyMCDM is a universal Python library that offers a wide range of MCDM methods, criteria weighting techniques, and normalization approaches (Kizielewicz et al., 2023; Shekhovtsov et al., 2025). Based on pyMCDM, the pymcdm-reidentify tool has been developed by utilizing the pyMCDM and Mealpy libraries (Kizielewicz and Sałabun, 2024). An important feature of pymcdm-reidentify is its capability to reconstruct MCDM models when rankings are known, but the original parameters are unavailable. It offers fuzzy normalization to enhance accuracy and adaptability, as well as advanced visualizations. In another Python package, PySensMCDA, Paradowski et al. (2024) apply a variety of sensitivity analyses within eight modules that include alternative modifications, weight adjustments, ranking alterations, compromise solutions, preference calculations, visualizations, and validation. The pyDecision library (Pereira et al., 2024) provides implementations of 70 MCDA methods and visualization functions. It also integrates the well-known Large Language Model (LLM),

ChatGPT, to enable an interactive analysis of MCDA methods. Such an LLM integration distinguishes pyDecision from other applications. However, no external studies have yet evaluated its practical usability. Although this approach seems very innovative, further research is required to validate its effectiveness, accuracy, and impact on user experience in real-world applications. More specifically, comprehensive evaluation studies should be conducted to determine whether it enhances decision-makers' intuitive understanding or increases the risk of misguidance, as LLMs can cause biases and AI hallucinations.

The decision support platform DecideXpert (Saoud et al., 2025) integrates both conventional and fuzzy versions of AHP and TOPSIS into a user-friendly, single-page application that requires no programming skills. The authors state that the platform is scalable and cross-platform compatible as it has been built with Java, Spring Boot, Angular, and MySQL.

Clearly, a significant portion of MCDM analysis solutions have been offered as Python libraries (e.g., pyMCDM). Whereas only a limited number of MCDM tools are available in the R programming environment, most of which include a small variety of MCDM methods. Among these R packages, IFMCDM tackles the MCDM problems with uncertainty (Roszkowska et al., 2024). It has been particularly designed to work with intuitionistic fuzzy sets for representing imprecise information, utilizing methods such as Intuitionistic Fuzzy TOPSIS and the Intuitionistic Fuzzy Synthetic Measure. Another example is the MCDA (Multi-criteria Decision Aiding) package introduced by Bigaret et al. (2017), which similarly includes only a few methods. However, a very recent R package called RMCDA provides a broad spectrum of methods that comprise traditional, probabilistic, and stratified decision-making methods (Najafi and Mirzei, 2025). This package also provides enhanced accessibility through its ShinyRMCDA web application.

In addition to packages developed in R and Python, there is a contribution in Julia, a relatively new language that is gaining popularity. In this regard, the JMcDM package provides implementations of 22 widely used MCDM methods, as well as Data Envelopment Analysis (DEA) (Satman et al., 2021).

The field of multi-criteria decision making encompasses a wide range of methods. Each of these methods has its own strengths and weaknesses relative to others, differing in terms of reliability, efficiency, and robustness. At this point, the critical question arises: Which MCDM method is the most suitable for the decision-making problem at hand? To answer this question precisely, benchmarking tools that enable comparisons of multiple methods are needed. Such tools should not only facilitate comparisons but also provide systematic evaluations through performance analysis, incorporating metrics such as sensitivity and stability. While the packages discussed here have been highly useful for various MCDM analyses and comparisons, there remains a lack of tools specialized in benchmarking. A benchmarking framework that successfully addresses these gaps can eliminate the burden of manual computations, allowing users to concentrate on interpreting results and deriving meaningful inferences by automating method selection, sensitivity testing, and ranking evaluations.

To address the aforementioned needs, this study introduces the mcdabench, an R package designed for benchmarking in MCDM analyses. The main contributions and novelties of this study can be summarized as follows:

- Comprehensive Unified Framework: Development of mcdabench, an all-in-one R package that integrates all phases of the MCDM workflow, including normalization, weighting, method execution, robustness analysis, pairwise comparison, and visualization.
- Extensive Method Encapsulation: Incorporation of 38 widely used traditional and modern MCDM methods into a single library to ensure cross-method consistency.
- Methodological Novelties in Sensitivity Analysis: Introduction of two original algorithmic contributions: a bi-directional generalized gradual weighting method (gengradwei) and random weight perturbations (genrandwei) for advanced robustness evaluations.
- Advanced Statistical Benchmarking: Integration of robust information-theoretic statistical tests—specifically Shannon’s entropy-based permutation tests and Jensen-Shannon Divergence-based bootstrap tests—offering significant comparative advantages over conventional rank correlations.
- Consensus and Focus Tools: Provision of diverse rank aggregation strategies (Borda, Kemeny-Young, Copeland, Markov) alongside a "Top-k" focus feature to isolate superior alternatives.
- Built-in Visualization: Inclusion of custom graphical functions to seamlessly demonstrate decision structures, weight fluctuations, and statistical comparison matrices.

The remainder of this paper is organized as follows. Section 2 presents the structure, content and functions of the package. Section 3 describes how to utilize the package over an industrial robot selection example. It is shown how to implement different MCDM methods and analyse the results in great detail. Section 4 concludes the paper and discusses future research directions.

2. Materials and Methods

The proposed package, mcdabench, establishes a benchmarking environment for comparing the ranking results of several MCDM methods. By integrating the six key steps of MCDM analysis, the package provides in-depth evaluations of case-specific scenarios. These key steps are structured as illustrated in Figure 1 and can be defined as follows: Normalization, weight calculation,

MCDM analyses, sensitivity/stability analyses, comparison, and aggregation. As shown in the figure, mcdabench utilizes these segments and thus offers more than a single decision-support tool.

The "normalize" function within the package implements several normalization methods frequently applied in MCDM analyses. During data preprocessing stage, normalization adjusts values with varying scales and magnitudes in decision matrices to ensure comparability. The current version of mcdabench supports MaxMin, Vector, Ratio (max), Sum, Enhanced accuracy, Nonlinear, Logarithmic, and Zavadskas-Turskis normalization methods.

In addition to normalization, the selection of an objective weighting method significantly influences the assigned importance of each criterion. The diversity of available weighting methods underscores the necessity of choosing an approach that aligns with the specific characteristics and objectives of the multi-criteria decision analysis. The "calcweights" function in the package currently offers eight objective weighting methods: Equal, Merec, Geometric, Gini, Critic, Standard Deviation, Rank sum, Rank order centroid, and Entropy. In addition to these objective methods, users may input subjective weights based on expert evaluations. This feature enables greater customization through user-defined weights.

The mcdabench package currently offers thirty-eight MCDM methods for decision-making. These are AHP, ARAS, AROMAN, COCOSO, CODAS, COPRAS, EDAS, ELECTRE family (I-IV), FUCA, Fuzzy TOPSIS, GRA, MABAC, MACONT, MAIRCA, MARCOS, MAUT, MAVT, MOORA, MEGAN (Cebeci, 2026), ORCA, ORETES, PROMETHEE family (I-VI), RAM, ROV, SMART, TOPSIS, VIKOR, WASPAS, WPM, and WSM. All of these methods share the common arguments "dmatrix", "bcvec", "weights", and "normethod", which represent the decision matrix, benefit-cost vector, criteria weights, and normalization methods, respectively. Alternatives are arranged in the rows and criteria in the columns of the decision matrix. In this package, benefit or profit criteria are coded as 1, while cost criteria are coded as -1 in the "bcvec" vector. The "weights" argument is provided as a numeric vector, or the type of weights can be specified using options such as "equal", "entropy", or "gini". The "normethod" argument initiates any method with a normalized matrix, and the normalization type can be defined as "maxmin", "vector", or "roc". This argument is optional, as most multi-criteria decision-making (MCDM) methods apply their own intrinsic normalization internally. In addition to these general arguments, certain methods require additional, method-specific parameters. For example, the VIKOR method uses the parameter "v", while the PROMETHEE family of methods requires preference functions ("prefuncs") and parameters such as "p", "q", and "s". If these parameters are not specified, the methods use their default values.

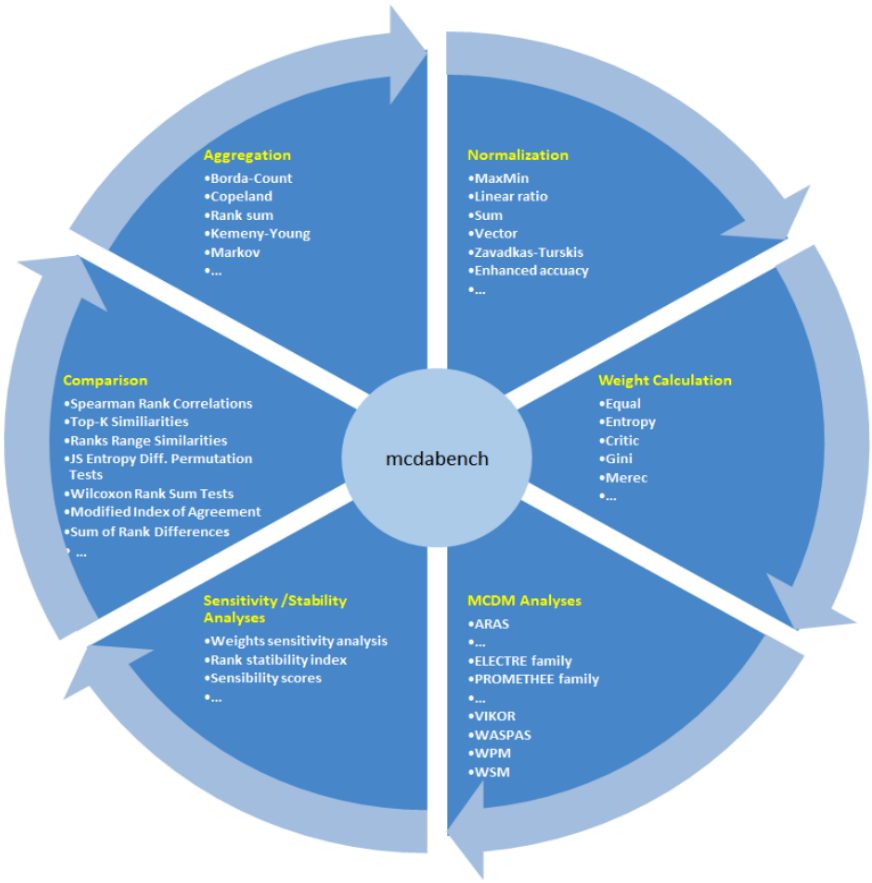

Figure 1. Components of the R package "mcdabench".

The "sensana" function within the package conducts sensitivity and stability analyses. This function calculates sensitivity scores and stability indexes to compare predefined conditions or assumptions, thereby evaluating how results vary across different scenarios. To enhance the robustness and reliability of decision-making processes, several analytical techniques are applied. Parameter sensitivity analysis examines the impact of changes in criteria weights or methodological parameters on final rankings or decisions. Stability analysis for alternatives evaluates the consistency of alternative rankings when minor adjustments are made to input data or preferences. The current version of the package includes measures such as the standard deviation of ranks for each alternative across all scenarios, which reflects the dispersion of ranks. The Rank Stability Index for each alternative is calculated as one minus the proportion of rank changes across scenarios; higher scores indicate greater rank stability. Rank Volatility scores for each alternative represent the proportion of rank changes between consecutive scenarios, with lower scores indicating reduced rank volatility. Weight Sensitivity scores for each scenario denote the average absolute rank change of all alternatives compared to the first scenario; lower scores suggest greater stability of the overall ranking relative to the initial scenario.

For sensitivity analyses, two novel contributions not yet implemented in other packages have been incorporated into the mcdabench package. The first, "gengradwei", is based on the gradual weighting method proposed by Stević et al. (2020). This approach has been examined by Rachman et al. (2024) using the formulas provided in the review by Demir, Chatterjee, and Pamučar (2024). The core operational mechanism of this function works as follows: it systematically isolates a single criterion and adjusts its initial weight across a series of user-defined percentage steps. In the mcdabench implementation, this framework is generalized to execute bi-directional sensitivity testing: both systematic reductions (decrements) and incremental changes (increases) of weights are supported. During each iteration, as the target criterion's weight changes, the remaining balance is proportionally redistributed among the other criteria to ensure the total weight sum strictly equals 1.0, while a lower bound is automatically enforced to prevent negative values. The resulting numeric matrix presents multiple weighting scenarios, which serve as a valuable tool for sensitivity analysis in MCDM models. Each row in the matrix represents a distinct weight scenario, with the first row containing the initial weights and subsequent rows reflecting modifications based on predefined reduction percentages. The second function, genrandwei, operates similarly but iteratively modifies the initial weights by introducing random perturbations within a defined step size. The adjusted weights are normalized to ensure their sum equals one, and negative weights are also prevented. The matrices generated by these functions are then used to evaluate the sensitivity or robustness of MCDM methods to weight fluctuations.

The "methodbench" function, central to the mcdabench package, evaluates and compares the results of MCDM methods across various datasets and input scenarios. As shown on the flowchart in Figure 2, methodbench accepts a decision matrix, a benefit/cost vector, optional weights (defaulting to equal), and a normalization method. It applies the selected MCDM methods to rank alternatives according to the specified criteria. The function produces a ranking matrix by aggregating individual rank vectors from each MCDM method. This ranking matrix is subsequently used to compute matrices for several pairwise comparison metrics and statistical tests, including Spearman rank correlation coefficients (SCC), Sałabun-Urbaniak Similarity (SUS or WS), Rank Range Similarity (RRS), Sum of Ranking Differences (SRD), Modified Index of Agreement (MIA), Wilcoxon Rank Sum Tests (WRST), Jensen-Shannon Divergence with permutation tests (JSDEPT), and entropy-based pairwise difference tests with bootstrap (ENTBST). Each metric is implemented as an individual function, accessible via the rankcompare interface, to calculate these measures and perform the corresponding statistical tests.

As a non-parametric statistical measure, SCC is widely in the MCDM world to calculate similarities between pairs of methods in comparison. When two methods yield similar rankings, the SCC approaches +1. As the similarity between rankings decreases, the SCC approaches zero. In contrast, the SCC approaches -1 when the methods produce completely opposite rankings. Besides SCC, SUS can provide a more nuanced or specific way of comparing rankings, particularly in the context of MCDM where the absolute positions and the distances between ranks are important. The SUS gives higher priority to the similarity of positions at the top of the ranking.

The Modified Index of Agreement (MIA) is an enhanced version of the original Index of Agreement developed by Willmott (1981). MIA serves as a standardized metric for evaluating the accuracy of model predictions relative to observed data. Modifications to the original Index of Agreement were introduced to address its limitations and to improve sensitivity and reliability (Willmott, 1984). The "rankmia" function within the package extends the use of MIA to the comparison of rank orderings, which is particularly relevant in MCDM problems. MIA values range from 0 to 1, with 1 representing perfect agreement and 0 indicating complete disagreement. The package also introduces the RRS metric, which quantifies the similarity between ranking methods by comparing selected alternatives across different ranking approaches.

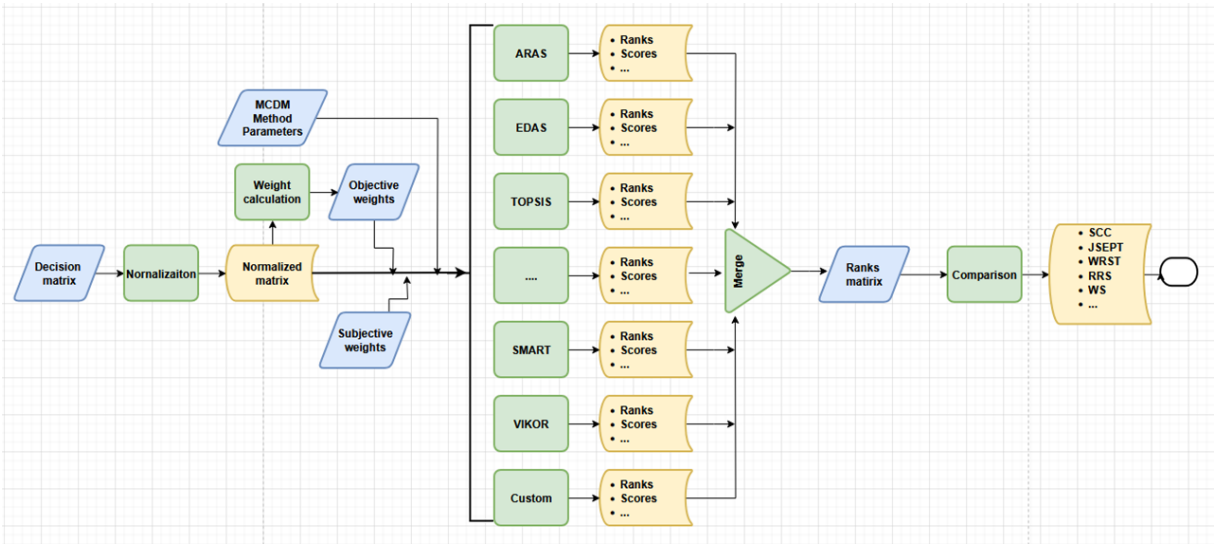

Figure 2. Flowchart for comparison of the MCDM methods.

To assess differences between compared methods, the Wilcoxon Rank-Sum Test (WRST) (Wilcoxon, 1992), also known as the Mann-Whitney U Test, is employed as a suitable non-parametric test. This test is applied alongside a p-value adjustment function to account for multiple comparisons. The "rankwilcox" function is utilized to perform WRSTs. In addition to WRST, the package implements the Permutation Test using Jensen-Shannon Divergences of the rankings (JSDEPT) and the Bootstrap Test for Entropy Differences (ENTDBST) to support MCDM comparison studies. The comprehensive theoretical background, mathematical formulations, and validation protocols of these information-theoretic tests are fundamentally based on the recent methodology established by Cebeci (2026). Unlike traditional non-parametric metrics such as Spearman's rank correlation, which are limited to evaluating linear or monotonic relationships, these entropy and divergence-based approaches evaluate rankings as probability distributions of information. Consequently, their primary comparative advantage lies in their unique capacity to robustly capture the structural uncertainty, information loss, and precise distance between diverse MCDM rankings using robust resampling techniques, making them highly optimized for rigorous benchmarking applications. The "rankentperm" function applies the JSDEPT and adjusts p-values using correction methods such as Bonferroni, Holm, and FDR. Furthermore, the "rankentboot" function conducts permutation tests based on Shannon entropy differences between rankings from the compared methods. Héberger (2010) introduced the SRD method to address ranking problems among chromatographic columns, predictive models, and analytical methods. The SRD method sums the absolute differences between the ideal and actual rankings for each compared method and then orders the computed SRD values. The "ranksrd" function within the package calculates relationships between pairs of compared methods.

To support decision-making by aggregating results from multiple MCDM methods, the "aggregate" function processes a rank matrix and generates a final rank vector using aggregation techniques such as Borda-Count, Copeland (Copeland, 1951), Kemeny-Young (Ali and Meilă, 2012; Kemeny, 1962; Young, 1995), Markov chains, Median method, and Rank sum. In this matrix, the rows correspond to evaluation methods (e.g., MCDA methods), and the columns represent alternatives. The function computes aggregated ranks and produces outranking and preference strings. Additionally, the package introduces the top-k technique for final ranking as a major contribution, recognizing that decision-makers frequently focus on identifying superior or inferior alternatives. This approach enables the determination of a final ranking based on the frequency with which alternatives appear in the top k positions across rankings generated by different methods.

The mcdabench package also offers advanced visualization tools. These tools employ graphical and diagrammatic methods to display decision matrix data, analysis results, and decision-maker preferences, thereby improving comprehension and interpretability in MCDM processes. In R, packages such as ggplot2, gplots, corrplot, and GGally are widely recognized for facilitating the visualization of input data, output rankings, and results from pairwise similarity and statistical tests. Additionally, visualization functions incorporated into the mcdabench package enable the generation of customized graphics for MCDM data and results. As detailed in the following section, functions including boxplotmcda, corplot, flowplot, parcorplot, pcaplot, rankheatmap, and sensplot support the visualization of boxplots for criteria in normalized matrices, correlation and similarity matrices, dominance flows, heat maps, and parallel coordinate plots. These tools assist in comparing ranking results and examining matrix structures.

3. Results and Discussion

Robotic manipulators, often colloquially termed robot arms, are autonomous devices designed to manipulate, modify, or transport objects without direct human intervention. Since the 1980s, they have become widely used in industry for welding, laser cutting, painting or coating, handling materials, assembling parts, packaging, etc. Over the years, demands for larger production volumes, higher-quality products, and lower manufacturing costs resulted in an increased amount of automation. Researchers, innovators, and companies came up with several novel designs and improvements to offer more advanced robotics solutions. However, from the invertors/manufacturers' point of view, it has become more complex to choose the right robotic solution as there are many of them available now. Fortunately, it is possible to make more objective investment decisions by using MCDM techniques in the evaluation of robotic systems. For this purpose, numerous studies have been conducted on this subject, often using fuzzy and hybrid approaches. (Shanmugasundar et al., 2023; Gamal and Mohamed, 2023; Goswami et al., 2021; Tran et al., 2025; Rashid et al., 2021; Garg and Garg, 2021; Khan et al., 2024). Among these studies, an important difference appears in terms of the alternatives and criteria considered. For example, Shanmugasundar, et al. (2023) listed the attributes of industrial robots as shown in Table 1.

For this paper, a synthetic decision matrix, representing six alternative robotic manipulators, has been constructed based on the ten criteria in Table 1 and is shown in Table 2.

Table 1. Criteria for selection industrial robots

Code	Criterion	Type
C1	Load Capacity (kg)	Benefit
C2	Repeatability Error (mm)	Cost
C3	Top Speed (mm/s)	Benefit
C4	Kinematic Structure (dof)	Benefit
C5	Vertical Reach (mm)	Benefit
C6	Horizontal Reach (mm)	Benefit
C7	Memory Capacity (steps)	Benefit
C8	Weight (kg)	Cost
C9	Power Consumption (kWh)	Cost
C10	Initial Cost (x1000$)	Cost

Table 2. Decision matrix and benefit-cost vector

Robot	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
R1	8.0	0.05	1800	6	1100	1500	1024	130	1.5	48
R2	6.8	0.07	1600	7	1200	1350	2048	115	1.2	32
R3	12.2	0.03	2000	6	1250	1600	1536	150	1.8	38
R4	15.1	0.04	2500	7	1400	1750	3000	165	2.2	45
R5	4.7	0.09	1200	5	950	1250	768	98	1.0	29
R6	3.5	0.10	900	4	800	1000	512	80	0.8	24
BC	1	-1	1	1	1	1	1	-1	-1	-1

In the last row, the Benefit-Cost (BC) vector, where 1 represents “benefit” or “max” criterion, and -1 shows “cost” or “min”.

3.1 Installation and Loading the Package

The mcdabench package can be downloaded from CRAN (https://cran.r-project.org/package=mcdabench) and installed by using the install_packages function of R. After the installation, the package can be called by using the library or require function. The following lines show how to implement these steps:

```
> install_packages("mcdabench", repo=https://cloud.r-project.org, dep=TRUE)
> library (mcdabench)
```

3.2 Data Input

The code snippet below creates the decision matrix and benefit-cost vector using the data presented in Table 2. Optionally, the data may be stored in comma-separated or tab-separated text files or other formats on disk. In such case, the relevant read functions (e.g. read.table, read.csv, read_excel) may be used to read the data and convert it into a decision matrix.

```
> df <- data.frame(
C1 = c(8.0, 6.8, 12.2, 15.1, 4.7, 3.5),
C2 = c(0.05, 0.07, 0.03, 0.04, 0.09, 0.10),
C3 = c(1800, 1600, 2000, 2500, 1200, 900),
C4 = c(6, 7, 6, 7, 5, 4),
C5 = c(1100, 1200, 1250, 1400, 950, 800),
C6 = c(1500, 1350, 1600, 1750, 1250, 1000),
C7 = c(1024, 2048, 1536, 3000, 768, 512),
C8 = c(130, 115, 150, 165, 98, 80),
C9 = c(1.5, 1.2, 1.8, 2.2, 1.0, 0.8),
C10 = c(48, 32, 38, 45, 29, 24))

> rownames(df) <- c("R1", "R2", "R3",
```

"R4", "R5", "R6")

```
> dmatrix <- as.matrix(df)
> print(dmatrix)
   C1   C2   C3   C4   C5   C6   C7   C8   C9  C10
R1  8.0 0.05 1800  6 1100 1500 1024 130 1.5  48
R2  6.8 0.07 1600  7 1200 1350 2048 115 1.2  32
R3 12.2 0.03 2000  6 1250 1600 1536 150 1.8  38
R4 15.1 0.04 2500  7 1400 1750 3000 165 2.2  45
R5  4.7 0.09 1200  5  950 1250  768  98 1.0  29
R6  3.5 0.10  900  4  800 1000  512  80 0.8  24

# Benefit-Cost vector: 1 = benefit, -1 = cost
bc <- c(1, -1, 1, 1, 1, 1, 1, -1, -1, -1)
```

3.3 Normalization and Weight Calculation

The "calcnormal" function is used to normalize original decision matrices. In the example below, it performs vector normalization, and the normalized matrix is stored in "nmatrix".

```
> nmatrix <- calcnormal(dmatrix=dmatrix, bcvec=bc, type="vector")
> print (nmatrix)
       C1   C2   C3   C4   C5   C6   C7   C8   C9  C10
R1 0.35 0.70 0.42 0.41 0.40 0.43 0.25 0.58 0.59 0.47
R2 0.30 0.58 0.37 0.48 0.43 0.39 0.49 0.63 0.67 0.65
R3 0.53 0.82 0.47 0.41 0.45 0.46 0.37 0.52 0.51 0.58
R4 0.66 0.76 0.58 0.48 0.50 0.50 0.72 0.47 0.40 0.50
R5 0.21 0.46 0.28 0.34 0.34 0.36 0.18 0.68 0.73 0.68
R6 0.15 0.40 0.21 0.28 0.29 0.29 0.12 0.74 0.78 0.74
```

To inspect criterion values within the normalized matrices visually, the "boxplotmcda" function can be executed using the line below, which creates the plot shown in Figure 3.

```
> boxplotmcda(nmatrix, mt="Vector Normalization")
```

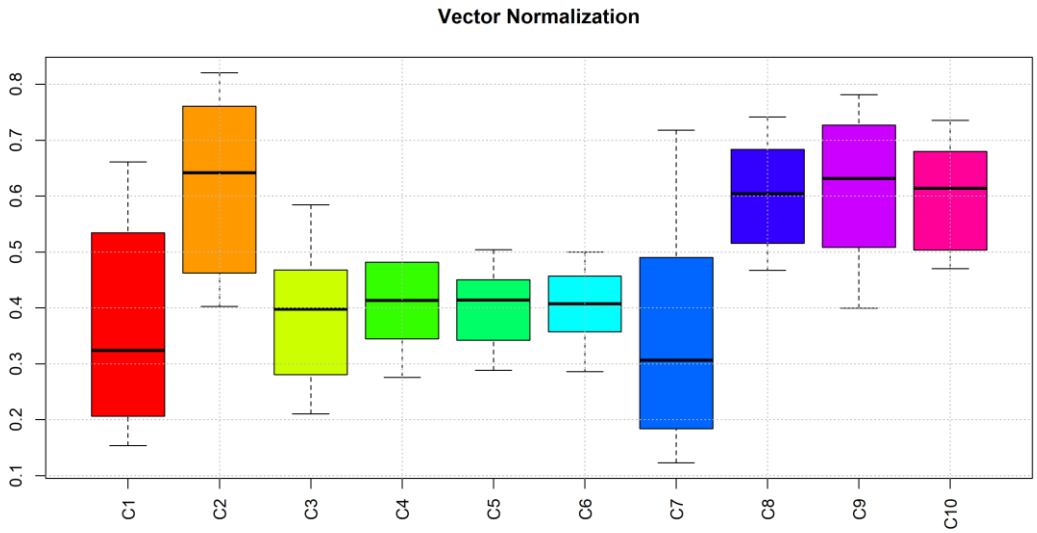

Figure 3. Boxplots of the criteria in the matrix normalized with vector normalization.

The "calcweights" function calculates objective weights using various methods. In the following lines of code, it applies different weighting methods on the vector-normalized "nmatrix" and calculates these weights. Then, they are combined inside a new matrix "wmatrix" for comparison and visualization.

```
> equwei <- calcweights(nmatrix, bcvec=bcvec, type="equal")
> critwei <- calcweights(nmatrix, bcvec=bcvec, type="critic")
> entwei <- calcweights(nmatrix, bcvec=bcvec, type="entropy")
> giniwei <- calcweights(nmatrix, bcvec=bcvec, type="gini")
> sdevwei <- calcweights(nmatrix, bcvec=bcvec, type="sdev")
> merecwei <- calcweights(nmatrix, bcvec=bcvec, type="merec")
> geomwei <- calcweights(nmatrix, bcvec=bcvec, type="geom")
> rocwei <- calcweights(nmatrix, bcvec=bcvec, type="roc")
> rswei <- calcweights(nmatrix, bcvec=bcvec, type="rs")

> wmatrix <- cbind( EQU=equwei, CRI=critwei, ENT=entwei,SD=sdevwei, GEO=geomwei, GIN=giniwei, MER=merecwei, RSM=rswei, ROC=rocwei)
> print(round(wmatrix, 2))
    EQU  CRI  ENT   SD   GEO  GIN   MER  RSM  ROC
C1  0.1 0.1175 0.2578 0.1502 0.0803 0.1810 0.0985 0.1818 0.3414
C2  0.1 0.1055 0.0687 0.1287 0.0925 0.0929 0.0995 0.1636 0.1707
C3  0.1 0.0787 0.1107 0.1026 0.0999 0.1181 0.1001 0.1455 0.1138
C4  0.1 0.0474 0.0384 0.0618 0.1292 0.0669 0.1019 0.1273 0.0854
C5  0.1 0.0439 0.0352 0.0597 0.1119 0.0665 0.1009 0.1091 0.0683
C6  0.1 0.0459 0.0339 0.0586 0.1173 0.0653 0.1012
```

0.0909 0.0569
C7 0.1 0.1292 0.3492 0.1702 0.0744 0.2106 0.0980 0.0727 0.0488
C8 0.1 0.1309 0.0269 0.0791 0.1048 0.0593 0.1004 0.0545 0.0427
C9 0.1 0.1803 0.0518 0.1097 0.0878 0.0802 0.0992 0.0364 0.0379
C10 0.1 0.1208 0.0272 0.0793 0.1018 0.0592 0.1002 0.0182 0.0341

To examine how criteria weights vary across different methods, the previously obtained "wmatrix" is visualized using the "parcorplot" function, which generates a parallel coordinate plot, as shown in Figure 4.

```
> parcorplot (wmatrix, xl="Weighting Methods", yl="Weight", lt="Criteria").
```

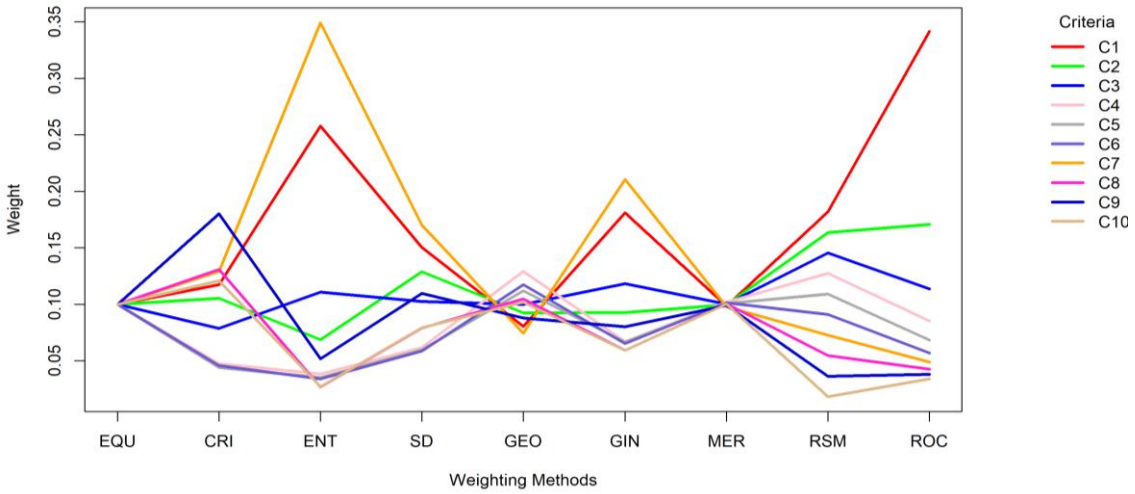

Figure 4. Parallel coordinate plot for the objective weights.

3.4 Applying MCDM Methods

MCDM methods provided by mcdabench are called with their specific parameters, and then the results are returned. The specific parameters for each method are located within the package's help documentation.

The example code chunk below demonstrates the SMART method, where the "str" function displays the returned values and their structures. As in this example, if the user does not provide the specific parameters in an MCDM method call, the function uses default values.

```
> ressmart <- smart(dmatrix=dmatrix, bcvec=bcvec, weights=equwei)
> str(ressmart)
List of 4
 $ Value_Matrix          : num [1:6, 1:10] 0.388 0.284 0.75 1 0.103 ...
 $ Weighted_Value_Matrix : num [1:6, 1:10] 0.0388 0.0284 0.075 0.0103 ...
 $ Overall_Scores        : num [1:6] 0.536 0.507 0.619 0.902 0.273 ...
 $ rank                  : Named num [1:6] 3 4 2 1 5 6
  ..- attr(*, "names")= chr [1:6] "R1" "R2" "R3" "R4" ...
  ..- attr(*, "names")= chr [1:6] "R1" "R2" "R3" "R4" ...

> print(ressmart_1$rank)
 R1  R2  R3  R4  R5  R6
  3   4   2   1   5   6
```

For each method, the package always returns a list component named "rank". It contains the ranking values of the alternatives, calculated using the average ties method, and the best score receives a rank of "1" in all of them.

3.5 Weight Sensitivity Analysis of MCDM Methods

The impact of criterion weights on any MCDM method provided in the package, can be investigated using the example codes in this section.

The example code uses the SMART method and the "weisana" function from the package. The criterion weights are changed at rates of 0.05, 0.1, 0.2, and 0.5, starting with initial weights of the CRITIC type ("critwei"). In the weisana function, "weimethod" and "weipars" specify the weight modification method and its parameters. In this example, the "gradual" method and "list(rp=c(0.05, 0.10, 0.2, 0.5))" for "wp" are used. The preferred MCDM method's name and its parameters can be set with "mcdamethod" and "methodpars". In this example, "smart" and "list(normethod="maxmin")" for "mp" were used.

The results can then be viewed using the "str" function. The "weights_matrix" shows the weights generated during modifications. The "ranking_matrix" lists the final ranks, with each row for a different weight scenario. The "sensitivity_table" shows the unique rank patterns across all scenarios.

```
# Weight sensitivity control using gradual weight modifying technique
> mp <- list(normethod="maxmin")
> wp <- list(rp=c(0.05, 0.10, 0.2, 0.5))

> grawei <- weisana(dmatrix = dmatrix, bcvec = bcvec,
   weights=critwei,  weimethod = "gradual", weipars = wp,
    mcdamethod = smart, methodpars = mp)
```

```
> str(grawei)
 List of 4
  $ computing_time   : 'difftime' num 0.02
   ..- attr(*, "units")= chr "secs"
  $ weights_matrix     : num [1:41, 1:10] 0.1175 0.1116 0.1057 0.094 0.0587 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:41] "init" "C1-5%" "C1-10%" "C1-20%" ...
   .. ..$ : chr [1:10] "C1" "C2" "C3" "C4" ...
  $ ranking_matrix   : num [1:41, 1:6] 3 3 3 4 4 3 3 3 3 3 ...
   ..- attr(*, "dimnames")=List of 2
   .. ..$ : chr [1:41] "init" "C1-5%" "C1-10%" "C1-20%" ...
   .. ..$ : chr [1:6] "R1" "R2" "R3" "R4" ...
  $ sensitivity_table:'data.frame':   5 obs. of  3 variables:
   ..$ Pattern: chr [1:5] "3,4,1,2,5,6" "4,3,1,2,5,6" "4,3,2,1,5,6" "2,4,1,3,5,6" ...
   ..$ Count  : int [1:5] 31 3 5 1 1
   ..$ Percent: num [1:5] 75.61 7.32 12.2 2.44 2.44
```

The previously obtained rank matrix can be used to calculate how many different ranking patterns have been formed, along with their counts and percentages. Figure 5 provides a visual presentation of the "sensitivity_table" extracted.

```
> senstable <- grawei$sensitivity_table # Different ranking patterns
> print(senstable)
```

	Pattern	Count	Percent
Ranking_1	3,4,1,2,5,6	31	75.61
Ranking_2	4,3,1,2,5,6	3	7.32
Ranking_3	4,3,2,1,5,6	5	12.20
Ranking_4	2,4,1,3,5,6	1	2.44
Ranking_5	2,5,1,3,4,6	1	2.44

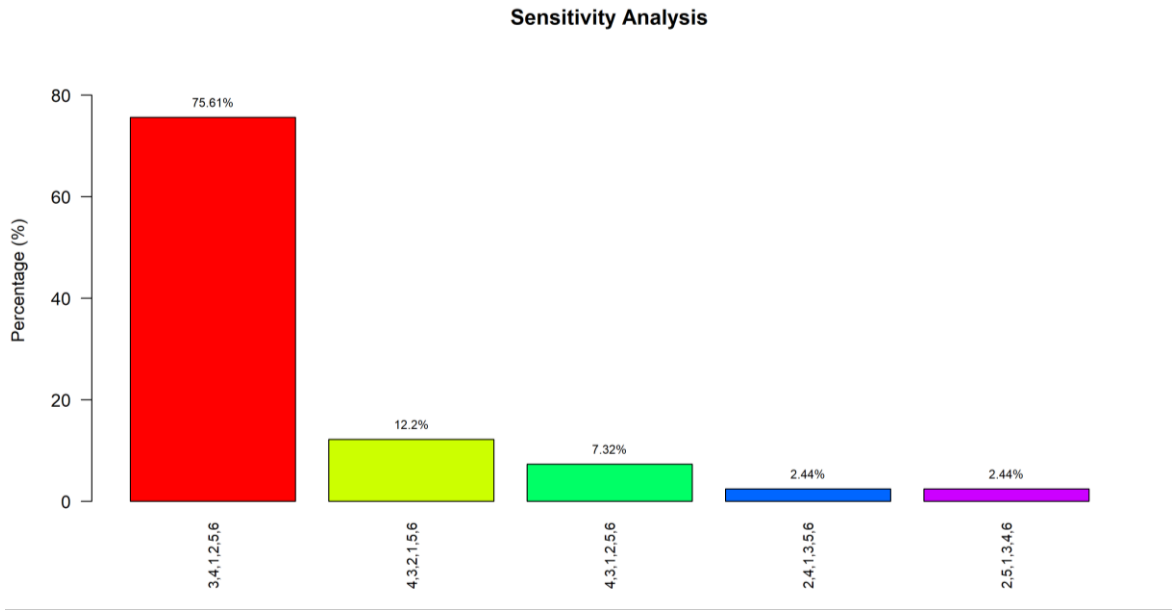

Figure 5. Different ranking patterns returned by weight sensitivity analysis.

According to the results above, "Ranking_1" is the dominant ranking pattern with the highest rate (75.61%) and frequency (31) through different weight scenarios. Despite the dominance of "Ranking_1", alternative ranking patterns have also emerged, indicating some variation in outcomes. For instance, "Ranking_2" and "Ranking_3" show rates of 7.32% and 12.20%, respectively, indicating that certain adjustments to the weighting may influence ranking patterns noticeably. Meanwhile, "Ranking_4" and "Ranking_5" emerged only once, making up a minimal rate of 2.44% each, suggesting that these patterns are much rarer occurrences, possibly resulting from edge-case weight distributions. The general implication is that the method is stable around the dominant pattern, but it can demonstrate variations for some weight modifications.

In addition, heatmaps of the ranks are quite useful for visualizing which weight scenarios lead to the formation of specific ranking patterns. For example, in Figure 6, it can be observed that high-level weight modifications in C1, C7, C9, and C10 have led to the formation of different rankings.

```
> rankmat <- grawei$ranking_matrix  # Ranking matrix
> rankheatmap(rankmat, colpal=4, cellnotes=T, tcol="white", dendro="both")
```

To wrap up the example, SMART has demonstrated remarkable robustness to weight sensitivity in the examined data set. The experiment can be repeated with "random" weight modification instead of "gradual". It can also be extended to benchmark novel MCDM methods when users load their custom functions into the R global environment as external functions. The most important point is that the custom functions should return a "rank" vector in which "1" is the rank for the best score.

In the package vignettes, additional illustrative examples were provided to test the custom MCDM functions.

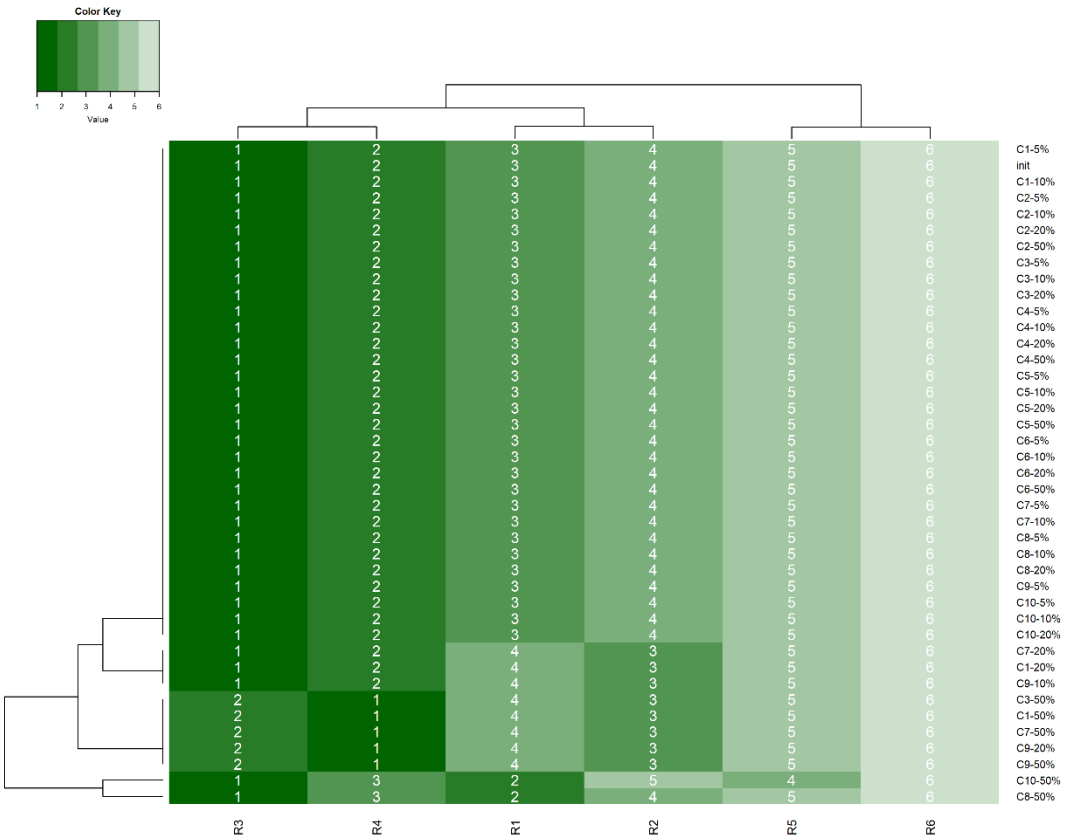

Figure 6. Heatmap of the rankings by SMART using the gradually modified weights.

3.6 Comparison of MCDM Methods

The following example illustrates the simultaneous execution of multiple MCDM methods using "methodbench", the most central function of the mcdabench package. Here, a list of selected methods is supplied to the "mcdm" argument, with their names specified via the "methodlist" vector. The "params" argument receives a list containing the required parameters for the selected methods; if omitted, the function applies default parameter values specific to each method. This parameter list encompasses values for all MCDM methods available in the package.

The "normethod" argument specifies the normalization method, which is applied by algorithms that require normalization.

```
> methodlist <- c("aras", "aroman", "codas", "edas",
    "elect4", "fuca", "gra", "mabac", "moora", "promet2",
    "smart",  "topsis", "vikor", "waspas")

> owei <- calcweights(nmatrix, bcvec = bcvec, type =
"equal")

> resmcda <- methodbench(dmatrix = dmatrix, bcvec =
bcvec,
  wei = owei, mcdm = methodlist, params = paramlist,
  normethod="maxmin")

> rankmat <- resmcda$rankmat

> print(rankmat)
```

	R1	R2	R3	R4	R5	R6
ARAS	4	3	2	1	5	6
AROMAN	4	3	2	1	5	6
CODAS	3	4	2	1	5	6
EDAS	4	3	2	1	5	6
ELECT4	3	4	2	1	5.5	5.5
FUCA	3	4	2	1	5	6
GRA	3	4	2	1	5	6
MABAC	4	3	2	1	5	6
MOORA	4	3	2	1	5	6
PROMET2	4	3	2	1	5	6
SMART	4	3	2	1	5	6
TOPSIS	4	2	3	1	5	6
VIKOR	4	2	3	1	5	6
WASPAS	4	3	2	1	5	6

In the example above, the "rankmat" is a ranking matrix formed by combining ranking vectors generated during the execution of various MCDM methods. This matrix serves as the basis for subsequent comparison and aggregation processes.

For deeper comparisons, heatmaps of ranking matrices can reveal similarities and differences in rankings among methods and may also facilitate clustering. Using the command below, the rankheatmap function generates a heatmap of the "rankmat" which is presented in Figure 7.

```
> rankheatmap(rankmat, colpal=1, cellnotes=TRUE,
tcol="black")
```

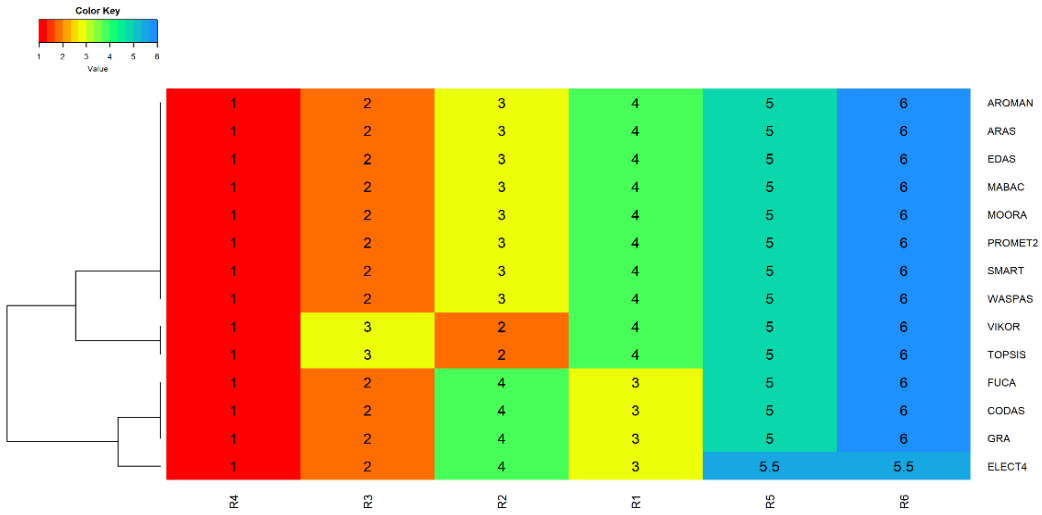

Figure 7. Heatmap of the rankings by the MCDM methods.

Another tool for comparisons is the rankcompare function which is executed to compare the MCDM methods by calculating similarity and dissimilarity coefficients and conducting statistical tests. The nboot and nperms arguments specify the number of iterations for bootstrap and permutation tests, respectively.
The entropyopt argument is set to either "jsd" for Jensen-Shannon entropy or "Shannon" for Shannon entropy. Additionally, the alpha argument determines the Type I error level for statistical tests, while padjmethod specifies the p-value adjustment method.

```
> rescomp <- rankcompare(rankmat,
    nperms = 100, nboot=100, entropyopt = "jsd",
    alpha = 0.05, padjmethod = "fdr")
```

The results obtained from the execution of rankcompare are returned as a list-type object. The Spearman correlation coefficients matrix, which is one of these results is displayed in Figure 8. In this matrix, the lower triangle represents the correlation coefficients, while the upper triangle shows the statistical significances with star notation (*: p<0.05, **: p<0.01, ***: p<0.001).

```
> print(rescomp$src) # Spearman rank correlations matrix
```

	ARAS	AROMAN	CODAS	EDAS	ELECT4	FUCA	GRA	MABAC	MOORA	PROMET2	SMART	TOPSIS	VIKOR	WASPAS
ARAS	1	***	**	***	**	**	**	***	***	***	***	**	**	***
AROMAN	1.00	1	**	***	**	**	**	***	***	***	***	**	**	***
CODAS	0.94	0.94	1	**	***	***	***	**	**	**	**	*	*	**
EDAS	1.00	1.00	0.94	1	**	**	**	***	***	***	***	**	**	***
ELECT4	0.93	0.93	0.99	0.93	1	***	***	**	**	**	**	*	*	**
FUCA	0.94	0.94	1.00	0.94	0.99	1	***	**	**	**	**	*	*	**
GRA	0.94	0.94	1.00	0.94	0.99	1.00	1	**	**	**	**	*	*	**
MABAC	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1	***	***	***	**	**	***
MOORA	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1	***	***	**	**	***
PROMET2	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1	***	**	**	***
SMART	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00	1	**	**	***
TOPSIS	0.94	0.94	0.83	0.94	0.81	0.83	0.83	0.94	0.94	0.94	0.94	1	***	**
VIKOR	0.94	0.94	0.83	0.94	0.81	0.83	0.83	0.94	0.94	0.94	0.94	1.00	1	**
WASPAS	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00	1.00	0.94	0.94	1

Figure 8. Spearman rank correlations matrix.

The "corplot" function is a powerful tool within the mcdabench package that generates correlograms to visualize relationships or similarities among variables as shown in Figure 9. By presenting correlation coefficients in a structured and intuitive format, correlograms facilitate quick assessment of the magnitude and direction of these relationships. The application of color gradients further enhances interpretability, enabling users to readily identify proximity patterns within the matrix. The following code chunk demonstrates the use of corplot for visualizing Spearman rank correlations between the rankings produced by the benchmark MCDM methods. Additionally, the corplot function can be applied to analyze all matrices presented below, including both the original and normalized decision matrices.

```
> corplot(cormat, fsize = 3, shape="square", fcolor =
"white",
    colpal=c("red","gray", "dodgerblue"), xlab="MCDM
Methods",
    ylab="MCDM Methods", title="Spearman Correlation
Matrix")
```

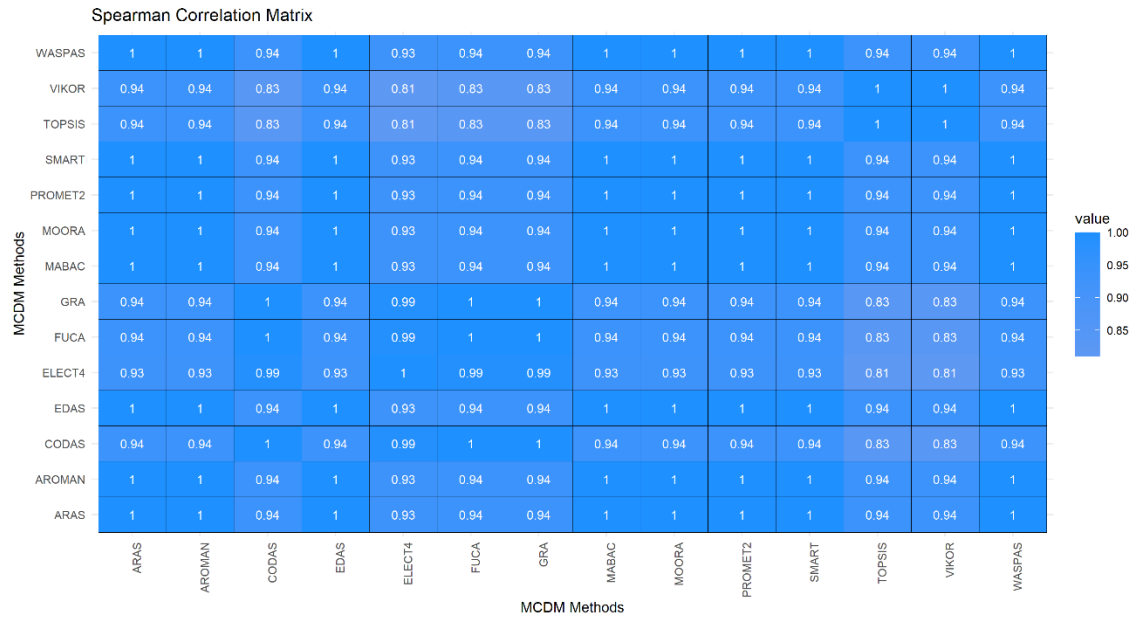

Figure 9. Correlogram for Spearman rank correlation coefficients.

As an additional demonstration, the Salabun-Urbaniak similarities matrix stored in the $wsrs element of the result object is displayed in Figure 10 below.

> print(rescomp$wsrs)

The results of permutation tests conducted for Jensen-Shannon Divergence are shown in Figure 11 where the lower triangle displays the JSD values between method pairs, and the upper triangle shows the corresponding p-values. According to the results in the example, there is no significant difference in JSD values for any of the method pairs ($p > 0.05$). This non-parametric test is a significant contribution of the package to MCDM research.

> print(rescomp$entper)

	ARAS	AROMAN	CODAS	EDAS	ELECT4	FUCA	GRA	MABAC	MOORA	PROMET2	SMART	TOPSIS	VIKOR	WASPAS
ARAS		1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00	1.00	0.90	0.90	1.00
AROMAN	1.00		0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00	1.00	0.90	0.90	1.00
CODAS	0.94	0.94		0.94	0.99	1.00	1.00	0.94	0.94	0.94	0.94	0.85	0.85	0.94
EDAS	1.00	1.00	0.94		0.93	0.94	0.94	1.00	1.00	1.00	1.00	0.90	0.90	1.00
ELECT4	0.93	0.93	1.00	0.93		1.00	1.00	0.93	0.93	0.93	0.93	0.85	0.85	0.93
FUCA	0.94	0.94	1.00	0.94	0.99		1.00	0.94	0.94	0.94	0.94	0.85	0.85	0.94
GRA	0.94	0.94	1.00	0.94	0.99	1.00		0.94	0.94	0.94	0.94	0.85	0.85	0.94
MABAC	1.00	1.00	0.94	1.00	0.93	0.94	0.94		1.00	1.00	1.00	0.90	0.90	1.00
MOORA	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00		1.00	1.00	0.90	0.90	1.00
PROMET2	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00		1.00	0.90	0.90	1.00
SMART	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00		0.90	0.90	1.00
TOPSIS	0.90	0.90	0.81	0.90	0.81	0.81	0.81	0.90	0.90	0.90	0.90		1.00	0.90
VIKOR	0.90	0.90	0.81	0.90	0.81	0.81	0.81	0.90	0.90	0.90	0.90	1.00		0.90
WASPAS	1.00	1.00	0.94	1.00	0.93	0.94	0.94	1.00	1.00	1.00	1.00	0.90	0.90	

Figure 10. Salabun-Urbaniak similarity matrix.

	ARAS	AROMAN	CODAS	EDAS	ELECT4	FUCA	GRA	MABAC	MOORA	PROMET2	SMART	TOPSIS	VIKOR	WASPAS
ARAS		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00
AROMAN	0.00		1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00
CODAS	0.02	0.02		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	1.00
EDAS	0.00	0.00	0.02		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
ELECT4	0.03	0.03	0.01	0.03		1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.97	1.00
FUCA	0.02	0.02	0.00	0.02	0.01		1.00	1.00	0.99	0.99	1.00	1.00	1.00	1.00
GRA	0.02	0.02	0.00	0.02	0.01	0.00		1.00	1.00	1.00	1.00	0.99	0.98	1.00
MABAC	0.00	0.00	0.02	0.00	0.03	0.02	0.02		1.00	1.00	1.00	1.00	1.00	1.00
MOORA	0.00	0.00	0.02	0.00	0.03	0.02	0.02	0.00		1.00	1.00	0.99	1.00	1.00
PROMET2	0.00	0.00	0.02	0.00	0.03	0.02	0.02	0.00	0.00		1.00	0.99	0.98	1.00
SMART	0.00	0.00	0.02	0.00	0.03	0.02	0.02	0.00	0.00	0.00		1.00	0.99	1.00
TOPSIS	0.03	0.03	0.08	0.03	0.09	0.08	0.08	0.03	0.03	0.03	0.03		1.00	1.00
VIKOR	0.03	0.03	0.08	0.03	0.09	0.08	0.08	0.03	0.03	0.03	0.03	0.00		1.00
WASPAS	0.00	0.00	0.02	0.00	0.03	0.02	0.02	0.00	0.00	0.00	0.00	0.03	0.03	

Figure 11. Permutation test matrix for Jensen-Shannon Divergence differences.

As another contribution of the package, Rank Range Similarities (RRS) approach has been offered. Using the RRS, the similarity of the top three ranked alternatives are shown in Figure 12. It is clear from the figure that CODAS, ELECTRE IV, FUCA, and GRA have identical rankings for the top three alternatives. The same evaluation holds for the other group formed by the remaining methods. The figure shows that all methods place at least the same two alternatives in the top three, with a similarity of 66.7%.

```
> print(rescomp$rangesim) # Rank range similarity matrix
> rankheatmap(rescomp$rangesim, colpal=3,
    cellnotes=TRUE, tcol="white")
```

Figure 12. Rank range similarity matrix for top-3 rank.

3.7 Sensitivity Analysis and Rank Aggregation

In this subsection, a sensitivity and stability analysis was conducted using the combined ranking matrix, obtained by running the MCDM methods chosen for benchmarking.

The first example returns Standard Deviation (SD), Two-Stage Standard Deviation (SD2), Coefficient of Relative Volatility (CRV), and Rank Stability Index (RSI) values. These values indicate that alternatives R4, R5, and R6 have stable ranks across all methods, while R1, R2, and R3 are less stable. Subsequently, sensitivity scores are presented, which confirm the stability scores.

```
> ressens <- sensana(rankmat)
> print(ressens$stabtable) # Stability measures
```

	R1	R2	R3	R4	R5	R6
SD	0.47	0.66	0.36	0	0.13	0.13
CRV	0.13	0.21	0.17	0	0.03	0.02
SD2	0.08	0.10	0.05	0	0.01	0.01
SRSI	0.31	0.46	0.15	0	0.15	0.15
RSI	0.93	0.86	0.93	1	0.93	0.93

```
> print(ressens$sensscores) # Sensitivity scores
```

R1	R2	R3	R4	R5	R6
0.29	0.43	0.14	0.00	0.04	0.04

In the second example, ranking results from the chosen MCDM methods are aggregated to arrive at the final ranking, which is then presented for decision-making. In the current version of the mcdabench package, aggregation can be performed using the Rank Sum, Median, Borda-Count, Copeland, Kemeny-Young, and Markov methods. As a contribution to the literature, the package also offers a Top-K result, which evaluates the alternatives based on their inclusion in the top "k" ranks.

```
> resagg <- aggregate(rankmat, topk=3, niters=100)
> print(resagg$preference_ranking)
```

	R1	R2	R3	R4	R5	R6
TOPK3	4	3	1.5	1.5	5.5	5.5
RANKSUM	4	3	2.0	1.0	5.0	6.0
MEDIAN	4	3	2.0	1.0	5.0	6.0
BORDACNT	4	3	2.0	1.0	5.0	6.0
COPELAND	4	3	2.0	1.0	5.0	6.0
KEMYNG	4	3	2.0	1.0	5.0	6.0
MARKOV	4	3	2.0	1.0	5.0	6.0

The third example presents the dominance flow of the alternatives based on the aggregated ranking results. According to the Top-3, R4 and R3 perform the same, while other aggregation methods show that R4 is the best alternative, followed by R3. The alternatives R1 and R2 are identical for each method. While the worst alternatives are revealed as R5 and R6, these ranks are considered the same for the Top-3.

```
> print(resagg$preference_table)
```

	Method	Outranking
1	TOPK3	R4 = R3 > R2 > R1 > R5 = R6
2	RANKSUM	R4 > R3 > R2 > R1 > R5 > R6
3	MEDIAN	R4 > R3 > R2 > R1 > R5 > R6

4	BORDACNT	R4 > R3 > R2 > R1 > R5 > R6
5	COPELAND	R4 > R3 > R2 > R1 > R5 > R6
6	KEMYNG	R4 > R3 > R2 > R1 > R5 > R6
7	MARKOV	R4 > R3 > R2 > R1 > R5 > R6

The dominance flow may be visualized by using "flowplot" which generates horizontal and vertical dominance graphics as shown in Figure 13. The code below draws the dominance flow in horizontal direction for Copeland method.

```
> flowplot(respref$preference_ranking["COPELAND",],
    colpal=cm.colors(7), txtcol = "black", orientation =
"horizontal")
```

Figure 13. Horizontal flow of dominance plot for Copeland aggregation.

4. Conclusion

This paper presented the architecture, functions, and core features of the novel R package, mcdabench. The mcdabench's functions for every stage of MCDM analysis, from data normalization to final ranking aggregation, were demonstrated with illustrative examples.

It was shown that a key strength of the package lies in its benchmarking and comparison functions, which enable simultaneous application of multiple methods and insightful statistical evaluations. By offering implementations of 38 MCDM methods, advanced weighting techniques, and sensitivity analysis, the package delivers not only a broad set of solutions but also a handful of analytical tools for interpreting the results. The aggregation techniques provide final decision rankings, complemented by an innovative "top-k" selection of alternatives. In summary, mcdabench has been described as a flexible, scalable, and transparent decision-making environment with robust, adaptable results suited to both academic and practical MCDM analyses.

While this paper illustrates the benchmarking capabilities using an industrial robot selection problem, the package can be deployed across a wide range of diverse empirical scenarios and disciplines. For instance, in engineering and infrastructure, it can guide renewable energy grid or plant site selection; in operations management, it can optimize green supply chain networks or supplier evaluation; and in finance and healthcare, it can support portfolio optimization or medical equipment procurement. Because mcdabench accepts any standard numeric decision matrix regardless of its dimensions, it is highly scalable. It can robustly handle complex, large-scale datasets with dozens of conflicting criteria and alternatives, ensuring that both academic researchers and industrial decision-makers can conduct rigorous, generalized sensitivity and statistical benchmarking analyses in any multi-criteria domain.

Future versions of mcdabench will include several new features and improvements. For example, missing value imputation methods will be introduced to maintain data consistency. Outlier and extreme-case analyses will be added to ensure method stability under atypical circumstances. Fuzzy versions of the MCDM methods, along with some hybrid examples, will provide applicability for additional uncertain and imprecise decision problems. It is also planned to include supplementary metrics for accuracy, consistency, and computational efficiency to provide more nuance in method performance. Furthermore, ranking methods studied by Gogodze (2021) will be integrated into the package to extend the range of analyses. Advanced Monte Carlo simulations will also be used to maximize the depth of uncertainty analyses by accounting for random variations in criterion weights and decision matrix values.

Author Contributions

The percentages of the author' contributions are presented below. The author reviewed and approved the final version of the manuscript.

	C.C.
C	100
D	100
S	100
DCP	100
DAI	100
L	100
W	100
CR	100
SR	100
PM	100

C= concept, D= design, S= supervision, DCP= data collection and/or processing, DAI= data analysis and/or interpretation, L= literature search, W= writing, CR= critical review, SR= submission and revision, PM= project management.

Conflict of Interest

The author declared that there is no conflict of interest.

Ethical Consideration

Ethics committee approval was not required for this study because of there was no study on animals or humans.

Acknowledgements

This research was not funded by any grant.

List of Abbreviations

AHP: Analytical Hierarchy Process
ARAS: Additive Ratio Assessment
AROMAN: Alternative ranking order method accounting for two-step normalization
COCOSO: Combined compromise solution
CODAS: Combinative Distance based Assessment
COPRAS: Complex Proportional Assessment
CRITIC: CRiteria Importance Through Intercriteria Correlation
CRV: Coefficient of Relative Volatility
DEA: Data Envelopment Analysis
EDAS: Evaluation based on Distance from Average Solution
ELECTRE: Elimination Et Choix Traduisant la Realité
ENTDBST: Bootstrap Test for Entropy Differences
FUCA: Faire Un Choix Adéquat
GRA: Grey Relational Analysis
JSD: Jensen-Shannon Divergence
JSDEPT: Permutation Test using Jensen-Shannon Divergences of the Rankings
LLM: Large Language Model
MABAC: Multi-Attribute Border Approximation Area Comparison
MACONT: Mixed Aggregation by Comprehensive Normalization Technique
MAIRCA: Multi-Attribute Ideal-Real Comparative Analysis
MAUT: Multi-Attribute Utility Theory
MAVT: Multi-Attribute Value Theory
MCDA: Multi-criteria Decision Aiding
MCDA: Multi-criteria Decision Analysis
MCDM: Multi-criteria Decision Making
MEGAN: Multi-Criteria Evaluation via Gradual-Weighting and Aggregation of Normalised Distance Matrices
MIA: Modified Index of Agreement
MOORA: Multi-Objective Optimization on the Basis of Ratio Analysis
ORCA: Operational Competitiveness Rating
ORESTE: Organization, Rangement Et Synthese De Donnes Relationnelles
SCC: Spearman Rank Correlation Coefficients
SD: Standard Deviation
SD2: Two-Stage Standard Deviation
SMART: Simple Multi-Attribute Rating Technique
SRD: Sum of Ranking Differences
SUS: Sałabun-Urbaniak Similarity
PROMETHEE: Preference Ranking Organization Method for Enrichment Evaluations
RAM: Root Assessment Method
ROV: Range of Value
RRS: Rank Range Similarity
RSI: Rank Stability Index
TOPSIS: Technique for Order of Preference by Similarity to Ideal Solution
VIKOR: Visekriterijumska Optimizacija i Kompromisno Resenje
WASPAS: Weighted Aggregated Sum Product Assessment
WRST: Wilcoxon Rank-Sum Test
WSM: Weighted Sum Model
WPM: Weighted Product Model

References

Ali, A., & Meilă, M. (2012). Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences*, *64*(1), 28–40. https://doi.org/10.1016/j.mathsocsci.2011.08.008

Bigaret, S., Hodgett, R. E., Meyer, P., Mironova, T., & Olteanu, A. L. (2017). Supporting the multi-criteria decision aiding process: R and the MCDA package. *EURO Journal on Decision Processes*, *5*(1–4), 169–194. https://doi.org/10.1007/s40070-017-0064-1

Cebeci, C. (2026). Multi-criteria evaluation with gradual-weighting and aggregation of normalised distance matrices: A case study in renewable energy grid selection. *PeerJ Computer Science*, *12*, Article e3819. https://doi.org/10.7717/peerj-cs.3819

Copeland, A. H. (1951). *A reasonable social welfare function*. University of Michigan.

Demir, G., Chatterjee, P., & Pamučar, D. (2024). Sensitivity analysis in multi-criteria decision making: A state-of-the-art research perspective using bibliometric analysis. *Expert Systems with Applications*, *237*, Article 121660. https://doi.org/10.1016/j.eswa.2023.121660

Gamal, A., & Mohamed, M. (2023). A hybrid MCDM approach for industrial robots selection for the automotive industry. *Neutrosophic Systems with Applications*, *4*, 1–11.

Garg, R. K., & Garg, R. (2021). Decision support system for evaluation and ranking of robots using hybrid approach. *IEEE Transactions on Engineering Management*, *70*(9), 3283–3296. https://doi.org/10.1109/TEM.2021.3079704

Gogodze, J. (2021). Ranking methods for multicriteria decision-making: Application to benchmarking of solvers and problems. *Scientific Programming*, *2021*, Article 5513860. https://doi.org/10.1155/2021/5513860

Goswami, S. S., Behera, D. K., Afzal, A., Razak Kaladgi, A., Khan, S. A., Rajendran, P., & Asif, M. (2021). Analysis of a robot selection problem using two newly developed hybrid MCDM models of TOPSIS-ARAS and COPRAS-ARAS. *Symmetry*, *13*(8), Article 1331. https://doi.org/10.3390/sym13081331

Héberger, K. (2010). Sum of ranking differences compares methods or models fairly. *TrAC Trends in Analytical Chemistry*, *29*(1), 101–109. https://doi.org/10.1016/j.trac.2009.09.009

Kemeny, J. G. (1962). *Mathematical models in the social sciences*. MIT Press.

Khan, N. A., Kumar, A., & Rao, N. (2024). Hybrid decision-making program for optimal robot selection using MCDM. In *2024 International Conference on Artificial Intelligence and Emerging Technology (Global AI Summit)* (pp. 173–177). IEEE. https://doi.org/10.1109/GlobalAISummit62156.2024.10947815

Kizielewicz, B., & Sałabun, W. (2024). The pymcdm-reidentify tool: Advanced methods for MCDA model re-identification. *SoftwareX*, *28*, Article 101960. https://doi.org/10.1016/j.softx.2024.101960

Kizielewicz, B., Shekhovtsov, A., & Sałabun, W. (2023). pymcdm—The universal library for solving multi-criteria decision-making problems. *SoftwareX*, *22*, Article 101368. https://doi.org/10.1016/j.softx.2023.101368

Kumar, R. (2025). A comprehensive review of MCDM methods, applications, and emerging trends. *Decision Making Advances*, *3*(1), 185–199. https://doi.org/10.31181/dma31202569

Kumar, R., & Pamučar, D. (2025). A comprehensive and

systematic review of multi-criteria decision-making (MCDM) methods to solve decision-making problems: Two decades from 2004 to 2024. *Spectrum of Decision Making and Applications*, *2*(1), 178–197. https://doi.org/10.31181/sdmap21202524

Najafi, A., & Mirzaei, S. (2025). RMCDA: The comprehensive R library for applying multi-criteria decision analysis methods. *Software Impacts*, *24*, Article 100762. https://doi.org/10.1016/j.simpa.2025.100762

Paradowski, B., Więckowski, J., & Sałabun, W. (2024). PySensMCDA: A novel tool for sensitivity analysis in multi-criteria problems. *SoftwareX*, *27*, Article 101746. https://doi.org/10.1016/j.softx.2024.101746

Pereira, V., Basilio, M. P., & Santos, C. H T. (2024). *Enhancing decision analysis with a large language model: pydecision, a comprehensive library of MCDA methods in Python* (arXiv:2404.06370). arXiv. https://doi.org/10.48550/arXiv.2404.06370

Rachman, A. P., Ichwania, C., Mangkuto, R. A., Pradipta, J., Koerniawan, M. D., & Sarwono, J. (2024). Comparison of multicriteria decision-making methods for selection of optimum passive design strategy. *Energy and Buildings*, *314*, Article 114285. https://doi.org/10.1016/j.enbuild.2024.114285

Rashid, T., Ali, A., & Chu, Y. M. (2021). Hybrid BW-EDAS MCDM methodology for optimal industrial robot selection. *PLOS ONE*, *16*(2), Article e0246738. https://doi.org/10.1371/journal.pone.0246738

Roszkowska, E., Jefmański, B., Dudek, A., & Kusterka-Jefmańska, M. (2024). IFMCDM: An R package for intuitionistic fuzzy multi-criteria decision making methods. *SoftwareX*, *26*, Article 101721. https://doi.org/10.1016/j.softx.2024.101721

Sałabun, W., & Urbaniak, K. (2020). A new coefficient of rankings similarity in decision-making problems. In V. Krzhizhanovskaya et al. (Eds.), *Computational science – ICCS 2020* (Lecture Notes in Computer Science, Vol. 12138). Springer.

Saoud, A., Lachgar, M., Hanine, M., El Dhimni, R., El Azizi, K., & Machmoum, H. (2025). decideXpert: Collaborative system using AHP-TOPSIS and fuzzy techniques for multicriteria group decision-making. *SoftwareX*, *29*, Article 102026. https://doi.org/10.1016/j.softx.2024.102026

Satman, M. H., Yıldırım, B. F., & Kuruca, E. (2021). JMcDM: A Julia package for multiple-criteria decision-making tools. *Journal of Open Source Software*, *6*(65), Article 3430. https://doi.org/10.21105/joss.03430

Shanmugasundar, G., Kalita, K., Čep, R., & Chohan, J. S. (2023). Decision models for selection of industrial robots—A comprehensive comparison of multi-criteria decision making. *Processes*, *11*(6), Article 1681. https://doi.org/10.3390/pr11061681

Shekhovtsov, A., Kizielewicz, B., & Sałabun, W. (2025). Version 1.3: pymcdm—The universal library for solving multi-criteria decision-making problems. *SoftwareX*, *29*, Article 102051. https://doi.org/10.1016/j.softx.2025.102051

Stević, Ž., Pamučar, D., Puška, A., & Chatterjee, P. (2020). Sustainable supplier selection in healthcare industries using a new MCDM method: Measurement of alternatives and ranking according to compromise solution (MARCOS). *Computers and Industrial Engineering*, *140*, Article 106231. https://doi.org/10.1016/j.cie.2019.106231

Stojčić, M., Zavadskas, E. K., Pamučar, D., Stević, Ž., & Mardani, A. (2019). Application of MCDM methods in sustainability engineering: A literature review 2008–2018. *Symmetry*, *11*(3), Article 350. https://doi.org/10.3390/sym11030350

Taherdoost, H., & Madanchian, M. (2023). Multi-criteria decision making (MCDM) methods and concepts. *Encyclopedia*, *3*(1), 77–87. https://doi.org/10.3390/encyclopedia3010006

Tran, N. T., Nguyen, T. S., & Tran, V. H. (2025). A hybrid triangular fuzzy MCDM model for evaluating and selecting the optimal industrial robot for manufacturing plants based on fuzzy AHP and fuzzy ARAS. *Engineering, Technology and Applied Science Research*, *15*(4), 24270–24276. https://doi.org/10.48084/etasr.11061

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution* (pp. 196–202). Springer. https://doi.org/10.1007/978-1-4612-4380-9_16

Willmott, C. J. (1981). On the validation of models. *Physical Geography*, *2*(2), 184–194. https://doi.org/10.1080/02723646.1981.10642213

Willmott, C. J. (1984). On the evaluation of model performance in physical geography. In *Spatial statistics and models* (pp. 443–460). Springer. https://doi.org/10.1007/978-94-017-3048-8_23

Young, P. (1995). Optimal voting rules. *Journal of Economic Perspectives*, *9*(1), 51–64. https://doi.org/10.1257/jep.9.1.51

Zavadskas, E. K., & Turskis, Z. (2011). Multiple criteria decision making (MCDM) methods in economics: An overview. *Technological and Economic Development of Economy*, *17*(2), 397–427. https://doi.org/10.3846/20294913.2011.593291