ADDRESSING METHODOLOGICAL UNCERTAINTY IN MCDM WITH A SYSTEMATIC PIPELINE APPROACH TO DATA TRANSFORMATION SENSITIVITY ANALYSIS

Juan B. Cabral[1,2,3] **- Alvaro Roy Schachner**[3,4]

[1] **Grupo de Innovación y Desarrollo Tecnológico, Comisión Nacional de Actividades Espaciales (GVT-CONAE), Córdoba, Argentina**
[2] **Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Córdoba, Argentina**
[3] **Facultad de Matemática Astronomía y Física, Universidad Nacional de Córdoba (FAMAF-UNC), Córdoba, Argentina**
[4] **Instituto de Astronomía Téorica y Experimental (IATE-CONICET), Córdoba, Argentina**

jbcabral@unc.edu.ar - alvaro.schachner@mi.unc.edu.ar

ABSTRACT

Multicriteria decision-making methods exhibit critical dependence on the choice of normalization techniques, where different selections can alter 20–40% of the final rankings. Current practice is characterized by the ad-hoc selection of methods without systematic robustness evaluation. We present a framework that addresses this methodological uncertainty through automated exploration of the scaling transformation space. The implementation leverages the existing Scikit-Criteria infrastructure to automatically generate all possible methodological combinations and provide robust comparative analysis.

Keywords: MCDM - IMPLEMENTATION - PIPELINES - PYTHON

1. INTRODUCTION

Multi Criteria Decision Methods (MCDM) have long been affected by a fundamental limitation: a critical dependency of the final results on methodological decisions; in particular, the election of the data scaling methods (Vafaei et al., 2018).

Empirical evidence shows that different normalization methods could alter around 20% to 40% of the final rankings in real applications (Aytekin, 2021; Krishnan, 2022). However, current practices are often characterized by the *ad-hoc* selection of scaling methods, based on methodological precedents, personal preference, or software availability, without a systematic evaluation of their impact on the robustness of the results. Some limitations of prevailing approaches can be outlined as follows: 1) The MCDM literature lacks *unified conceptual frameworks* for systematically evaluating sensitivity to scaling transformations. Existing studies are fragmented, employ heterogeneous methodologies, and are often limited to pairwise comparisons between specific methods without considering the broader methodological landscape (Mukhametzyanov, 2021). 2) Traditional approaches generally provide point estimates without quantifying uncertainty, which may obscure the inherent variability introduced by methodological choices; particularly in contexts where methodological transparency is essential. 3) Available software implementations of MCDM methods typically require users to specify a single normalization procedure prior to the analysis, which restricts the possibility of systematically exploring the methodological space (Kizielewicz et al., 2023), among other limitations.

To address these limitations, we propose a combinatorial approach—in the mathematical sense of systematically exploring all possible combinations from finite sets of

methodological choices—that enables exhaustive evaluation of scaling and aggregation method combinations, providing explicit quantification of methodological uncertainty bounds.

Tools derived from techniques such as Grid Search and their implementation in Scikit-Learn (Pedregosa et al., 2011) illustrate how the field of machine learning has successfully adopted combinatorial pipeline architectures. This design pattern serves as a reference point for our proposed solution and conceptual validation in the MCDM domain, highlighting the feasibility of systematically exploring complex methodological spaces through the automated combination of algorithmic components. Moreover, by leveraging modern parallel computing frameworks, it becomes possible to overcome the computational barriers that have historically limited the practicality of exhaustive combinatorial approaches.

In this work, we introduce a systematic framework for the analysis of MCDM data transformations, which enables the automated exploration of the methodological space and the explicit quantification of its uncertainty, built upon the existing infrastructure of *Scikit-Criteria*. In particular, the *RanksComparator* proved especially valuable, as it already facilitates the comparison and evaluation of different rankings (Cabral, 2025).

2. SCIKIT-CRITERIA RANK COMPARISON INFRASTRUCTURE

Scikit-Criteria has evolved into a comprehensive library providing a powerful set of tools for discrete multi-criteria decision-making methods (Cabral et al., 2016). In this work, we specifically built upon the existing infrastructure: the `RanksComparator`.

The `RankResult` structure represents the output of MCDM methods that produce ordered rankings of alternatives. It incorporates method metadata, intermediate calculations, and also provides the ability to generate untied rankings through the `untied_rank_` property, which resolves ties by following the original order of alternatives while preserving their relative positions. On the other hand, `RanksComparator` enables the comparison of multiple rankings over the same set of alternatives (Cabral, 2025), providing comprehensive uncertainty quantification through multiple complementary metrics: `corr()` computes pairwise correlation of rankings, measuring linear agreement between methodological combinations; `cov()` calculates pairwise covariance of rankings, quantifying joint variability between methods; `r2_score()` computes pairwise coefficient of determination regression score function, measuring how well one ranking can predict another; and `distance()` calculates pairwise distance between rankings, providing geometric interpretation of methodological differences. Additionally, it offers visualizations via flow diagrams, heatmaps, and boxplots, as well as data conversion to widely adopted Python types such as `pandas.DataFrame` (McKinney et al., 2011). This multidimensional characterization enables decision-makers to establish bounds on methodological uncertainty and assess the robustness of their conclusions.

This infrastructure provides the base architecture for our combinatorial framework, where we extend our comparison capabilities toward a systematic exploration over some complete methodological space over `SKCCombinatorialPipeline`.

3. COMBINATORIAL PIPELINES

For several versions of Scikit-Criteria, `SKCPipeline` allows composing processing pipelines where each step transforms the decision matrix until reaching a final aggregator that produces a `RankResult` (see Figure 1). This sequential architecture works with predefined individual steps.

```
from skcriteria.pipeline import mkpipe  # import pipeline
pipeline = mkpipe(  # create pipeline
   NegateMinimize(),  # negate to maximize all criteria
   FilterGT({'criteria': 300}),  # apply satisficing filter
   FilterNonDominated(),  # remove dominated alternatives
   SumScaler(target="weights"),  # normalize weights proportionally
   MinMaxScaler(target="matrix"),  # normalize matrix by min-max
   TOPSIS()  # apply TOPSIS method
)
pipeline.evaluate(dm)  # evaluate some decision-matrix
```

Fig. 1: Modular MCDM pipeline in Scikit-Criteria that shows the systematic composition of data transformations, alternative filters, and normalization methods, prior to the final evaluation.

The new `SKCCombinatorialPipeline` implements true combinatorial exploration by systematically generating the Cartesian product of methodological alternatives at each pipeline step. When a step contains n alternative methods and the next step contains m alternatives, the system automatically constructs all $n \times m$ possible combinations, ensuring exhaustive coverage of the methodological space.

The algorithm works through a systematic four-stage process: (1) combinatorial generation by applying the Cartesian product over collections of algorithmic components, (2) pipeline instantiation through automatic construction of `SKCPipeline` objects with unique identifiers for each generated combination, (3) distributed evaluation via parallel invocation of each constructed pipeline, returning a `RanksComparator` object, and (4) infrastructure integration through transparent utilization of the existing API for ranking comparison and analysis.

For example, by defining two steps: `scaler` with two alternatives `SumScaler` and `VectorScaler`; and `agg`) with two options each `WSM` and `TOPSIS`; the system automatically generates four pipelines: `SumScaler+WSM`, `SumScaler+TOPSIS`, `VectorScaler+WSM`, and `VectorScaler+TOPSIS`. This architecture fully leverages the existing *Scikit-Criteria* infrastructure—particularly `RanksComparator`—to provide automatic methodological sensitivity analysis through systematic generation and comparison of all possible methodological combinations.

4. CASE STUDY: CRYPTOCURRENCY EVALUATION

To demonstrate the practical application of our framework, we present a scaling and aggregation function sensitivity analysis using the cryptocurrency evaluation dataset from Van Heerden et al. (2021), which measures nine cryptocurrencies (ADA, BNB, BTC, DOGE, ETH, LINK, LTC, XLM, XRP) across multiple criteria including market capitaliza-

tion, volatility, and trading volume. The experiment consists of loading this dataset and creating a combinatorial pipeline that includes a preprocessing inverter for minimization criteria (`InvertMinimize()`), three scaling options (`SumScaler`, `VectorScaler`, and `MinMaxScaler` applied to the data matrix), and two aggregation methods (`WeightedSumModel()` and `TOPSIS()`), automatically generating *six* different methodological scenarios (3 scalers × 2 aggregation methods) that evaluate each combination and visualize the resulting ranking distributions. This algorithmic idea can be appreciated in more detail in Figure 2, its Python code implementation in Figure 3. The output of the code can be found in Figure 4 where The left plot displays the resulting box plot of ranking distributions, while the right plot shows the correlation matrix between multi-criteria decision-making methods (Abbreviations: Sum = SumScaler, Vec = VectorScaler, MM = MinMaxScaler, WSM = WeightedSumModel. All methods use InvertMinimize transformation).

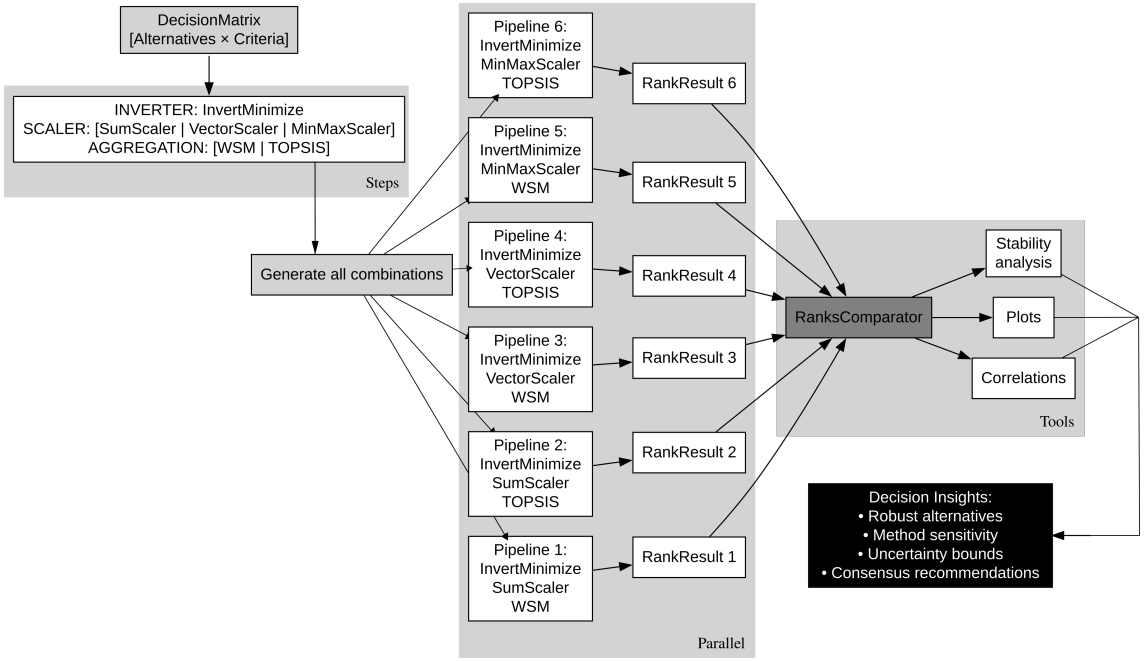


Fig. 2: Combinatorial pipeline framework architecture for systematic MCDM sensitivity analysis.

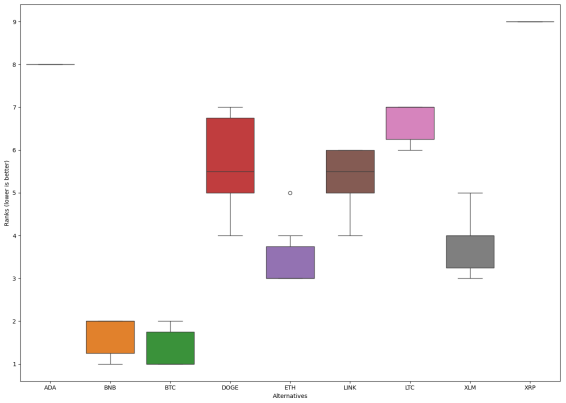
From the box plot analysis, several key insights can be observed. BNB and BTC consistently achieve superior positions across all methodological combinations, indicating that these alternatives are robust options regardless of the decision approach used. Conversely, the rankings of DOGE and XRP remain consistently low across all methods, indicating poor performance under the evaluated criteria. Additionally, mid-ranking alternatives such as ETH and ADA show moderate variations, revealing where methodological sensitivity is highest.

On the other hand, the correlation analysis reveals important patterns in methodological agreement. Sum WSM and Vec WSM show perfect correlation (1.000), indicating that these scaling methods produce identical rankings when paired with WSM aggregation. Similarly, Vec TOPSIS and Sum TOPSIS demonstrate very high correlation (0.983),

```
dm = skc.datasets.load_van2021evaluation()  # Load dataset
pipeline = mkcombinatorial(  # create combinatorial pipeline
  InvertMinimize(),
  [
    SumScaler(target="matrix"),
    VectorScaler(target="matrix"),
    MinMaxScaler(target="matrix")
  ],
  [WeightedSumModel(), TOPSIS()]
)
rank_comparator = pipeline.evaluate(dm) # Evaluate
rank_comparator.plot()  # plot
rank_comparator.corr()  # correlation matrix
```

Fig. 3: Example use of the combinatorial framework.

suggesting robust agreement across different scaling approaches when using TOPSIS. However, the lowest correlations occur between WSM and TOPSIS methods paired with MinMax scaling (0.850), indicating that this represents the frontier of methodological uncertainty. Nevertheless, all correlations exceed 0.850, suggesting that while methodological decisions matter, the fundamental ranking structure remains relatively stable across this methodological space.



Method	Sum WSM	Sum TOPSIS	Vec WSM	Vec TOPSIS	MM WSM	MM TOPSIS
Sum WSM	1.000	0.950	1.000	0.933	0.950	0.850
Sum TOPSIS	0.950	1.000	0.950	0.983	0.933	0.933
Vec WSM	1.000	0.950	1.000	0.933	0.950	0.850
Vec TOPSIS	0.933	0.983	0.933	1.000	0.900	0.967
MM WSM	0.950	0.933	0.950	0.900	1.000	0.867
MM TOPSIS	0.850	0.933	0.850	0.967	0.867	1.000

Fig. 4: Output of the Cryptocurrency Evaluation

5. CONCLUSIONS

In this work we present `SKCCombinatorialPipeline`, a framework that systematically addresses the problem of methodological uncertainty in MCDM through automated exploration of the study space. The implementation intelligently leverages the existing *Scikit-Criteria* infrastructure, particularly `RanksComparator`, to provide robust comparative analysis of multiple methodological configurations. The main contribution could lie in materializing theoretical principles of sensitivity analysis into practical tools that may facilitate exhaustive exploration of the methodological space without manual intervention,

potentially establishing new standards of methodological transparency for contemporary MCDM practice.

6. ACKNOWLEDGMENTS

This work was supported by CONICET and CONAE. A.R.S. was supported by CONICET fellowships. This article has been revised using large language models (Claude, ChatGPT, Gemini) in order to improve the clarity and correctness of the text. All technical-scientific content remains property of the authors.

7. REFERENCES

Aytekin, A. (2021). Comparative analysis of the normalization techniques in the context of mcdm problems. *Decision Making: Applications in Management and Engineering*, 4(2):1–25.

Cabral, J. B. (2025). Beyond single rankings: A systematic approach to compare multi-criteria decision methods. In *XVIII Simposio Argentino de Informática Industrial e Investigación Operativa (SIIIO 2025) - JAIIO 54 (CABA, 2025).* SADIO. In press.

Cabral, J. B., Luczywo, N. A., and Zanazzi, J. L. (2016). Scikit-criteria: Colección de métodos de análisis multi-criterio integrado al stack científico de python. In *XIV Simposio Argentino de Investigación Operativa (SIO 2016)-JAIIO 45 (Tres de Febrero, 2016).*

Kizielewicz, B., Shekhovtsov, A., and Sałabun, W. (2023). pymcdm—the universal library for solving multi-criteria decision-making problems. *SoftwareX*, 22:101368.

Krishnan, A. R. (2022). Past efforts in determining suitable normalization methods for multi-criteria decision-making: A short survey. *Frontiers in big data*, 5:990699.

McKinney, W. et al. (2011). pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9.

Mukhametzyanov, I. (2021). Specific character of objective methods for determining weights of criteria in mcdm problems: Entropy, critic and sd. *Decision Making: Applications in Management and Engineering*, 4(2):76–105.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Vafaei, N., Ribeiro, R. A., and Camarinha-Matos, L. M. (2018). Data normalisation techniques in decision making: case study with topsis method. *International journal of information and decision sciences*, 10(1):19–38.

Van Heerden, N. A., Cabral, J. B., and Luczywo, N. (2021). Evaluation of the importance of criteria for the selection of cryptocurrencies. *arXiv preprint arXiv:2109.00130*.