Beyond Single Rankings: A Systematic Approach to Compare Multi-Criteria Decision Methods

Juan B. Cabral[1,2,3]
jbcabral@unc.edu.ar

[1] Grupo de Innovación y Desarrollo Tecnológico, Gerencia de Vinculación Tecnológica, Centro Espacial Teófilo Tabanera, GVT-CONAE sede Córdoba, Argentina
[2] CONICET - Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina.
[3] Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba (FAMAF-UNC). Córdoba, Córdoba, Argentina

Abstract. *Scikit-Criteria* is a library that has evolved into a powerful toolkit for discrete multi-criteria methods. In this paper, we present the `RanksComparator` tool, designed to compare and evaluate different rankings of alternatives generated by various methods applied to the same decision problem. This infrastructure provides statistical metrics and visualizations that allow users to assess the robustness of results, identify discrepancies between rankings, and effectively communicate similarities and differences between methods, thereby improving the transparency of the decision-making process.

Keywords: Discrete multi-criteria decision, Software, Python, Rankings

Más allá de los rankings individuales: Un enfoque sistemático para comparar métodos de decisión multicriterio

Abstract. *Scikit-Criteria* es una biblioteca que ha evolucionado hasta convertirse en un potente conjunto de herramientas para métodos multicriterio discretos. En este trabajo, presentamos la herramienta `Ranks Comparator` , diseñada para comparar y evaluar diferentes clasificaciones de alternativas generadas por distintos métodos aplicados al mismo problema de decisión. Esta infraestructura proporciona métricas estadísticas y visualizaciones que permiten evaluar la robustez de los resultados, identificar discrepancias entre rankings y comunicar eficazmente las similitudes y diferencias entre métodos, mejorando la transparencia del proceso de toma de decisiones.

Keywords: Decisión multicriterio discreta, Software, Python, Rankings

1 Introduction

Scikit-Criteria [4] is a library that we originally presented as a collection of very simple functions, which over time we have transformed into a powerful toolkit for representation and analysis of discrete multi-criteria methods. As a result, it has been used in multiple works to provide mathematical and technical support for decision-making processes in complex scenarios [6].

In this context, it is often reasonable to apply several ranking methods to the same set of alternatives to evaluate robustness, reveal sensitivities, and improve understanding of the problem. Research indicates that using multiple methods can provide confirmation about which criteria and alternatives are most sensitive to changes in input parameters [12]. However, this often leads to different results. This inconsistency raises several important questions: How does the position of each alternative change according to the method used? How similar or different are the rankings produced by different methods? Are there significant correlations between the results of different techniques? What is the level of agreement or disagreement between different approaches?

To address these questions, in this work we present a high-level view of the design of the `Rank Comparator` tool that is implemented within *Scikit-Criteria* . This tool aims to assisting in the complex decision-making process by providing an analytical framework to compare and critically evaluate different rankings of alternatives, thereby improving the reliability and transparency of the decision-making process.

2 Conceptual Framework for Ranking Comparison

When multiple MCDM methods are applied to the same decision problem, each method produces a unique ranking of alternatives based on its specific algorithmic approach and underlying assumptions. The fundamental premise of ranking comparison lies in the fact that these methods, although addressing the same problem, may yield different results due to their distinct mathematical formulations and preference aggregation mechanisms.

Additionally, there is another case that can produce multiple rankings even when using the same method: ranking reversal. Ranking reversal occurs when the relative order of existing alternatives changes after adding, removing, or replacing alternatives in the decision set, while keeping evaluation criteria and their relative importance constant. This phenomenon manifests in five distinct

[4] https://scikit-criteria.quatrope.org/

types [2]: changes due to irrelevant alternatives (Type 1), replacement of non-optimal alternatives with worse ones (Type 2), transitivity violations (Type 3), problem decomposition (Type 4), and elimination of non-discriminating criteria (Type 5). Detection is performed through the Wang-Triantaphyllou three-test framework [14] that verifies the preservation of optimal alternatives, logical transitivity, and consistency in problem decomposition. The mathematical roots of the phenomenon typically stem from normalization dependencies and changes in reference points inherent to MCDM methods. Ranking reversal constitutes a critical theoretical problem because it violates fundamental axioms of rational decision-making—such as Independence of Irrelevant Alternatives, transitivity, and invariance principles—thereby questioning the scientific validity of MCDM methods by producing logically inconsistent results that can lead to erroneous decisions in critical applications [2].

It is within this context that a robust and systematic ranking analysis tool becomes essential, under the following assumptions: (1) all compared rankings refer to the same set of alternatives, (2) the rankings are meaningful and represent valid preference orders, and (3) differences between rankings can be quantified and interpreted to evaluate the consistency and robustness of the employed methods.

3 Ranks and Ranks–Comparators

For this work, from all the infrastructure offered in *Scikit-Criteria* , we are only interested in two types of data or classes: `RankResult` and `RanksComparator`.

First, the `RankResult` class was designed to represent the output of a multi-criteria decision-making method that delivers an ordered ranking of alternatives as a result. It is designed to handle data representing the ordinal classification of alternatives (where lower values generally indicate higher preference). The class incorporates data on which method created the ranking, support for storing intermediate calculations, ranking manipulation, and tie handling. All these features can be seen in Code 1.

Before concluding with `RankResult`, probably the most interesting property to analyze is `untied_rank_`. It is implemented in such a way that it resolves ties in a ranking by using the `numpy.argsort()` function [5]. The algorithm first checks if there are ties; if there are none, it simply returns the original ranking. If there are ties, it applies `argsort()` to the ranking values and adds 1 to the result to maintain the convention that rankings start at 1 (not 0). This approach assigns unique and consecutive positions to each alternative while preserving the general relative order, transforming for example a ranking `[1, 1, 2, 3]` (with a tie for first place) into `[1, 2, 3, 4]` since `argsort()` prioritizes the first occurrence in case of equal values. This property is useful for statistical analyses and visualizations that require distinct positions for each element.

Second, we analyze the functionalities offered by `RanksComparator`, which emerged from the need to compare multiple rankings generated by different

[5] https://numpy.org/doc/2.2/reference/generated/numpy.argsort.html

```
>>> import skcriteria as skc
# Create a RankResult object representing a ranking of alternatives
>>> rank = skc.agg.RankResult(
...     method="AMethod", alternatives=["A", "B", "C"], values=[1, 1, 2],
...     extra={"intermediate_calculations": 0.8})
>>> rank.method, rank.alternatives, rank.values # Basic properties
'AMethod', ['A', 'B', 'C'], [1, 1, 2]
>>> rank.extra_.intermediate_calculations # Extra information
0.8
>>> rank.has_ties_  # ties in the ranking?
True
>>> rank.ties_  # Frequency of each value in the ranking
Counter({1: 2, 2: 1})
>>> rank.untied_rank_  # a version without ties
[1, 2, 3]
```

Code 1: Example of creating and inspecting a RankResult object. The object represents a ranking where alternatives `A` and `B` are tied for first place and `C` is in second place. The code demonstrates accessing the basic properties of the ranking including method name, alternatives, values, intermediate calculations, tie detection, and untied rank generation.

methods or different configurations of multi-criteria decision-making methods. It implements an iterable interface (rankings can be traversed with a for-loop) and offers methods to analyze the consistency and similarity between various rankings on the same set of alternatives. Additionally, it provides functionalities for conversion to `pandas.Dataframes` [9], statistical calculations (including correlations, covariances, coefficients of determination, and distances between rankings), specialized visualizations through the plot accessor (such as flow diagrams, heat maps, correlation graphs, and more), and flexible indexing mechanisms that allow accessing individual rankings or creating sub-comparators. This component directly addresses the need to evaluate the robustness of multi-criteria decisions, reveal sensitivities in the methods, and provides an analytical framework to compare and critically evaluate different classifications of alternatives, thus improving the reliability and transparency of the decision-making process. This infrastructure can be visualized graphically in the class diagram in Figure 1.

The use of our implementation can be seen in Code 2 [6]. The cryptocurrency example demonstrates these concepts in practice: three different MCDM methods (TOPSIS, ELECTRE, and MOORA [4, 5, 7, 10]) evaluate nine cryptocurrencies using multiple criteria such as market capitalization, volatility, and

[6] A much more extensive application example can be found at https://scikit-criteria.quatrope.org/en/0.8.7/tutorial/rankcmp.html

```
>>> import matplotlib.pyplot as plt
>>> import skcriteria as skc
>>> from skcriteria.pipeline import mkpipe
>>> from skcriteria.preprocessing.invert_objectives import (
...     InvertMinimize, NegateMinimize)
>>> from skcriteria.preprocessing.filters import FilterNonDominated
>>> from skcriteria.preprocessing.scalers import SumScaler, VectorScaler
>>> from skcriteria.agg.simple import (
...     WeightedProductModel, WeightedSumModel)
>>> from skcriteria.agg.similarity import TOPSIS

>>> dm = skc.datasets.load_van2021evaluation() # Load a decision matrix

>>> # Create pipelines for different MCDM methods
>>> ws_pipe = mkpipe(
...     InvertMinimize(), FilterNonDominated(),
...     SumScaler(target="weights"), VectorScaler(target="matrix"),
...     WeightedSumModel())
>>> wp_pipe = mkpipe(
...     InvertMinimize(), FilterNonDominated(),
...     SumScaler(target="weights"), VectorScaler(target="matrix"),
...     WeightedProductModel())
>>> tp_pipe = mkpipe(
...     NegateMinimize(), FilterNonDominated(),
...     SumScaler(target="weights"), VectorScaler(target="matrix"),
...     TOPSIS())

>>> # Evaluate the decision matrix using three different methods
>>> r0, r1, r2 = ws_pipe.evaluate(dm), wp_pipe.evaluate(dm),
↪  tp_pipe.evaluate(dm)
>>> rcmp = mkrank_cmp(r0, r1, r2)  # a comparator from the three rankings
>>> rcmp.corr()  # Calculate the correlation matrix between rankings
Method                 WeightedSumModel  WeightedProductModel    TOPSIS
Method
WeightedSumModel               1.000000              0.816667  0.783333
WeightedProductModel           0.816667              1.000000  0.916667
TOPSIS                         0.783333              0.916667  1.000000

>>> # Plot the positions across methods
>>> rcmp.plot(); plt.show()  # box diagram showing alternatives change
< The output is included as a figure in the paper >
```

Code 2: Code snippet demonstrating the basic usage of the `RanksComparator` infrastructure. The example loads a cryptocurrency evaluation dataset from [13] with a 7-day window, evaluates it using three different multi-criteria decision methods, creates a comparator, calculates correlations between the rankings, and generates a box visualization. This is a simplified adaptation of the *Scikit-Criteria* tutorial.

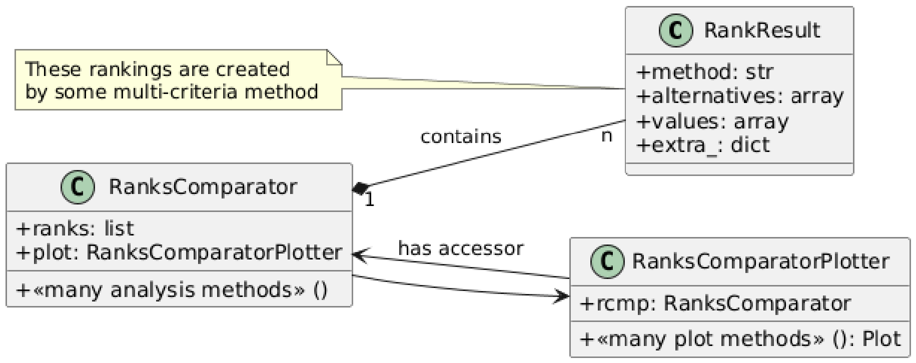

Fig. 1. Class diagram showing the relationships between `RankResult`, `RanksComparator`, and `RanksComparatorPlotter` classes. The diagram illustrates how `RanksComparator` contains multiple `RankResult` objects (which represent rankings created by multi-criteria methods) and has an accessor relationship with `RanksComparatorPlotter` that provides various visualization methods. Each class is shown with its key attributes and methods.

trading volume. The resulting visualization can be observed in Figure 2, where we can notice that the consistent positioning of BNB and BTC at the top across all methods suggests that these alternatives are robust choices regardless of the decision approach used, while the lower rankings of DOGE and XRP across all methods indicate poor performance under the evaluated criteria. The moderate variations observed for middle-ranked alternatives (such as ETH and ADA) reveal where method sensitivity is highest, providing valuable information for decision-makers about which alternatives require more careful consideration and when results are sufficiently robust to make decisions with confidence. When rankings show high correlation (as indicated by the statistical metrics provided by `RanksComparator`), decision-makers can have greater confidence in their choices, while low correlations signal the need for deeper analysis of why methods disagree [8].

4 Conclusions

The `RanksComparator` module significantly enhances multi-criteria decision analysis by providing a comprehensive framework for comparing different ranking methods. Through statistical metrics (correlation, covariance, R^2, distances) and informative visualizations, this tool not only provides mechanisms for statistical comparison but also offers insights into the stability and sensitivity of multi-criteria decision methods. By making these comparisons more accessible and interpretable, we enable decision analysts to improve the transparency and reliability of decision-making processes across various domains where multiple criteria must be balanced.

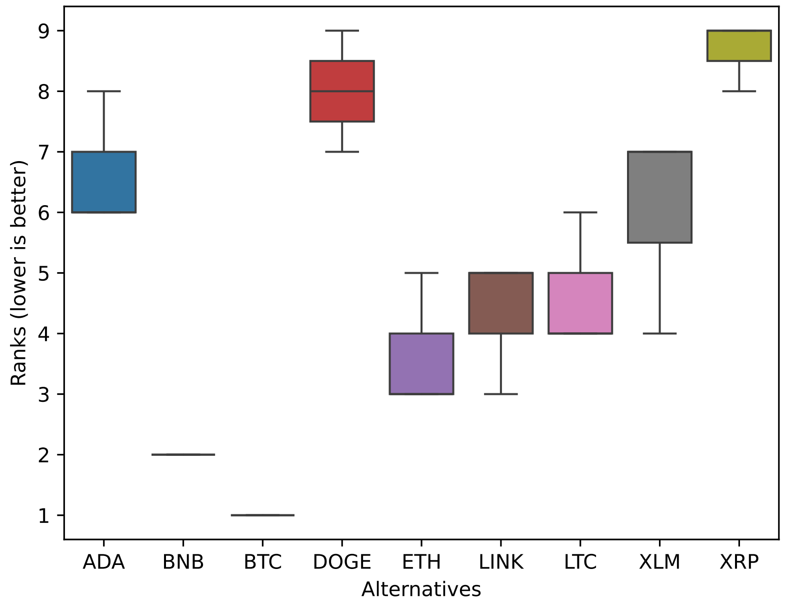

Fig. 2. Box plot showing the distribution of ranks across different multi-criteria decision methods for nine cryptocurrency alternatives. The y-axis represents ranking positions where lower values indicate better performance. The cryptocurrencies *BNB* and *BTC* consistently achieve top positions, while *DOGE* and *XRP* typically rank lower. This visualization was generated using the `RanksComparator`'s box plot functionality, based on the cryptocurrency dataset from [13] and following examples from the *Scikit-Criteria* tutorial.

Our future work aims to develop a series of algorithms for automated and reproducible ranking reversal analysis [15]. Since all definitions of rank reversals involve comparing multiple rankings, the data structure presented in this paper serves as the foundation for addressing various ranking reversal algorithms. As a preview, we can mention that one algorithm for ranking reversal analysis—which evaluates what happens when a non-optimal alternative is modified or removed from the decision matrix—is already implemented in *Scikit-Criteria* [7]. This implementation demonstrates the practical utility of the `RanksComparator` infrastructure for addressing complex decision analysis challenges.

Acknowledgments. This work was partially supported by the Consejo Nacional de Investigaciones Cient´ıficas y Tecnicas (CONICET, Argentina). This research employed the Cornell University xxx.arxiv.org repository, the Python programming language, the Numpy and Scipy libraries, and the other packages. To ensure grammatical and spelling accuracy, this paper has been reviewed and refined using advanced language models including Claude (developed by Anthropic), ChatGPT (developed by OpenAI), and Gemini (developed by Google) [1, 3, 11], while all content, ideas, and technical assertions remain solely those of the authors.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aires, R. F. d. F., & Ferreira, L. (2018). The rank reversal problem in multicriteria decision making: A literature review. *Pesquisa Operacional*, *38*(2), 331–362.

Anthropic. (2025). *Claude 3.7 sonnet and claude code*. Anthropic. Retrieved March 2, 2025, from https://www.anthropic.com/news/claude-3-7-sonnet

Brauers, W. K., & Zavadskas, E. K. (2006). The moora method and its application to privatization in a transition economy. *Control and cybernetics*, *35*, 445–469.

Brauers, W. K. M., & Zavadskas, E. K. (2012). Robustness of multimoora: A method for multi-objective optimization. *Informatica*, *23*(1), 1–25.

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

Hwang, C.-L., & Yoon, K. (1981). Methods for multiple attribute decision making. In *Multiple attribute decision making* (pp. 58–191). Springer.

[7] https://github.com/quatrope/scikit-criteria/blob/master/skcriteria/cmp/ranks_rev/rank_inv_check.py

Kou, G., Lu, Y., Peng, Y., & Shi, Y. (2012). Evaluation of classification algorithms using mcdm and rank correlation. *International Journal of Information Technology & Decision Making*, *11*(01), 197–225.

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.

Roy, B. (1990). The outranking approach and the foundations of electre methods. In *Readings in multiple criteria decision aid* (pp. 155–183). Springer.

Team, G., Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., et al. (2023). Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Triantaphyllou, E., & Sánchez, A. (1997). A sensitivity analysis approach for some deterministic multi-criteria decision-making methods. *Decision sciences*, *28*(1), 151–194.

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

Wang, X., & Triantaphyllou, E. (2008). Ranking irregularities when evaluating alternatives by using some electre methods. *Omega*, *36*(1), 45–63.

Wang, X., & Triantaphyllou, E. (2014). Ranking irregularities when evaluating alternatives by using some multi-criteria decision analysis methods. *Industrial and Systems Engineering*, *819*.