# MCEWR

Multi-Criteria Evaluation Model for Water Resources

## Methods Explained

The ten evaluation methods of MCEWR, the weighting schemes (objective, equal, Best-Worst) and the robustness analyses, one by one: the idea, the steps, a solved example computed by hand, and the clicks that reproduce it on the website

Model developed by Mohamed Rami Mahmoud, 1995-2026

**Prof. Mohamed Rami Mahmoud, Ph.D., P.E.**
Research Professor Emeritus, National Water Research Center, Cairo, Egypt
rami@m-rami.org · www.m-rami.org · rami.mrami.org
ORCID: 0000-0002-3393-987X · LinkedIn: linkedin.com/in/mohamed-rami-64947570

WAM · CP · PROMETHEE II · ELECTRE II · AHP · EV · RA · TOPSIS · VIKOR · ELECTRE III · entropy · CRITIC · MEREC · equal · Best-Worst

Edition 2.1.0 - September 2026

*A teaching document for students and academic colleagues. One small problem - three alternatives, three criteria - is solved with every method, weighted by every scheme and tested for robustness, so that everything can be compared on the same numbers. Companion documents: MCEWR_Quick_Start.pdf, MCEWR_Web_Tutorial.pdf (the website step by step), MCEWR_Web_User_Guide.pdf (every control), MCEWR_Method_Reference.pdf (the full mathematics, options and verification).*

# Contents

**1. How to use this tutorial** ..... 1
1.1 The ten methods at a glance ..... 1
**2. The problem used throughout, and the two preparation steps** ..... 3
2.1 The decision table ..... 3
2.2 Preparation step 1 – standardization ..... 3
2.3 Preparation step 2 – the weights ..... 4
**3. Weighted Average Method (WAM)** ..... 5
3.1 The idea ..... 5
3.2 What it needs ..... 5
3.3 The steps ..... 5
3.4 Solved example ..... 5
3.5 Do it on the website ..... 5
3.6 When to use it, and the usual mistakes ..... 6
**4. Compromise Programming (CP)** ..... 7
4.1 The idea ..... 7
4.2 What it needs ..... 7
4.3 The steps ..... 7
4.4 Solved example with s = 1 ..... 7
4.5 Solved example with s = 2 ..... 8
4.6 Do it on the website ..... 8
4.7 When to use it, and the usual mistakes ..... 8
**5. PROMETHEE II (and PROMETHEE I)** ..... 9
5.1 The idea ..... 9
5.2 What it needs ..... 9
5.3 The steps ..... 9
5.4 Solved example ..... 9
5.5 Do it on the website ..... 10
5.6 When to use it, and the usual mistakes ..... 10
**6. ELECTRE II** ..... 11
6.1 The idea ..... 11
6.2 What it needs ..... 11
6.3 The steps ..... 11
6.4 Solved example ..... 11
6.5 Do it on the website ..... 12
6.6 When to use it, and the usual mistakes ..... 12
**7. Analytic Hierarchy Process (AHP)** ..... 13
7.1 The idea ..... 13
7.2 What it needs ..... 13
7.3 The steps ..... 13
7.4 Solved example ..... 13
7.5 Do it on the website ..... 14
7.6 When to use it, and the usual mistakes ..... 14
**13. Weights: objective schemes, equal weights, Best-Worst** ..... 15
13.1 The idea ..... 15
13.2 What each of the three rewards ..... 15
13.3 What each of them needs – and what happens when the data do not oblige ..... 15
13.4 CRITIC, solved by hand ..... 15
13.5 Entropy and MEREC on a problem you can type in ..... 16
13.6 Using them, and the usual mistakes ..... 17
13.7 Equal weights: the neutral starting point ..... 18
13.8 The Best-Worst Method: fewer questions, a linear program ..... 18
**14. How firm is the answer? Sensitivity and rank reversal** ..... 21
14.1 Why ask ..... 21
14.2 The weight sweep and the critical share ..... 21
14.3 Rank acceptability under random perturbation ..... 22
14.4 Rank reversal: does the answer depend on the list of alternatives? ..... 22
14.5 Putting the three analyses together ..... 23
14.6 The usual mistakes ..... 23
14.7 Version 2.1: the decision values, one at a time ..... 23

14.8 Version 2.1: the settings of the method....23
14.9 Version 2.1: Monte Carlo robustness on the data and the weights....24
**8. Expected Value (EV)**....25
8.1 The idea....25
8.2 What it needs....25
8.3 The steps....25
8.4 Solved example....25
8.5 Do it on the website....26
8.6 When to use it, and the usual mistakes....26
**9. Regime Analysis (RA)**....27
9.1 The idea....27
9.2 What it needs....27
9.3 The steps....27
9.4 Solved example....27
9.5 Do it on the website....28
9.6 When to use it, and the usual mistakes....28
**10. TOPSIS**....29
10.1 The idea....29
10.2 What it needs....29
10.3 The steps....29
10.4 Solved example....29
10.5 Do it on the website....30
10.6 When to use it, and the usual mistakes....30
**11. VIKOR**....31
11.1 The idea....31
11.2 What it needs....31
11.3 The steps....31
11.4 Solved example....31
11.5 Do it on the website....32
11.6 When to use it, and the usual mistakes....32
**12. ELECTRE III**....33
12.1 The idea....33
12.2 What it needs....33
12.3 The steps....33
12.4 Solved example....33
12.5 Do it on the website....34
12.6 When to use it, and the usual mistakes....34
**15. The ten answers side by side**....35
15.1 What this table is telling you....35
15.2 What to do when methods disagree....35
**16. A checklist for your own study**....37

# 1. How to use this tutorial

This tutorial explains the ten evaluation methods of MCEWR one by one (chapters 3 to 12), then the three ways of computing weights from the data itself (chapter 13) and the analyses that test how firm a ranking is (chapter 14). Every chapter follows the same six parts, so that a method can be learnt in twenty minutes and compared with the others:

- **The idea** – what the method believes a good decision is, in plain words.
- **What it needs** – the data and the settings the method requires.
- **The steps** – the calculation, one numbered step at a time, with the formula written in ordinary letters.
- **Solved example** – the same small problem for every method, computed here by hand, number by number, so that every intermediate table can be checked with a calculator.
- **Do it on the website** – the exact clicks that reproduce the same numbers at mcewr.m-rami.org.
- **When to use it, and the usual mistakes** – how to choose the method, and what goes wrong most often.

One small problem runs through the whole tutorial: three alternatives, three criteria (section 2). It is the textbook example of Nijkamp, Rietveld and Voogd (1990, page 68), it is the first verification example of the 1995 dissertation, and it is small enough to compute by hand. Chapters 13 and 14 then leave the ranking methods aside for the two questions that decide whether a result is worth reporting – where the weights come from, and how firm the answer is – and chapter 15 puts the ten answers side by side, which is where the real lesson of multi-criteria evaluation appears.

> **Note.** Every number printed in this tutorial was produced by the MCEWR engine and then reproduced by hand. If your screen shows the same numbers, you have understood the method and the settings; if it does not, the settings differ – each chapter lists them exactly.

## 1.1 The ten methods at a glance

| No. | Method | It ranks by | Best value | Chapter |
|---|---|---|---|---|
| 1 | Weighted Average Method (WAM) | a weighted sum of the standardized values | largest | 3 |
| 2 | Compromise Programming (CP) | the distance to an ideal point | smallest | 4 |
| 3 | PROMETHEE II (and I) | how much each alternative beats the others, pair by pair | largest | 5 |
| 4 | ELECTRE II | agreement and veto between pairs, then two graphs | smallest | 6 |
| 5 | Analytic Hierarchy Process (AHP) | priorities from pairwise judgements | largest | 7 |
| 6 | Expected Value (EV) | an order of importance turned into weights | largest | 8 |
| 7 | Regime Analysis (RA) | the probability of winning a pair over all admissible weights | largest | 9 |
| 8 | TOPSIS | closeness to the ideal and distance from the worst | largest | 10 |
| 9 | VIKOR | a compromise between group utility and worst regret | smallest | 11 |
| 10 | ELECTRE III | credibility of outranking with thresholds, then two distillations | smallest | 12 |

Five of them – WAM, CP, PROMETHEE II, ELECTRE II and AHP – are the methods of the original 1995 program and are **audited**: the Python edition reproduces the published results digit for digit. The other five were added in version 2.0 and are marked **provisional**: they follow the primary literature and pass independent checks, but a

published worked example has still to be reproduced by the author. Use them, compare them, and say so when you cite them.

# 2. The problem used throughout, and the two preparation steps

## 2.1 The decision table

Three alternatives (say three project sites) are judged on three criteria. For every criterion, **larger is better**. The file is `NIJKAMP.MCE`, the first example in the list on the website.

| | Criterion 1 (weight 0.4) | Criterion 2 (weight 0.2) | Criterion 3 (weight 0.4) |
|---|---|---|---|
| Alternative 1 | 50 000 | 500 | 80 |
| Alternative 2 | 100 000 | −100 | 60 |
| Alternative 3 | 80 000 | 100 | 50 |

Look at the table before computing anything. No alternative is best everywhere: alternative 2 leads on criterion 1, alternative 1 leads on criteria 2 and 3, alternative 3 is second on criterion 1 and in the middle on criterion 2. **That is why a method is needed**: if one alternative were best on every criterion, it would simply win, and no multi-criteria evaluation would be necessary.

## 2.2 Preparation step 1 – standardization

The three criteria are measured in different units (money, an index that can be negative, a percentage). They cannot be added as they stand: 50 000 units of criterion 1 and 80 units of criterion 3 do not compare. Every method therefore starts by putting all criteria on the same scale. The usual choice, and the one used in this tutorial, is **standardization 1**:

```
r = (v − min) / (max − min)
```

where v is the value, and min and max are the smallest and the largest value of that criterion among the alternatives present. The result runs from 0 (the worst alternative on that criterion) to 1 (the best). For a criterion to be minimized the model inverts the formula, so that 1 always means good.

**Step 1.** Criterion 1: min = 50 000, max = 100 000, so max − min = 50 000. Alternative 1: (50 000 − 50 000)/50 000 = **0**. Alternative 2: (100 000 − 50 000)/50 000 = **1**. Alternative 3: (80 000 − 50 000)/50 000 = **0.6**.

**Step 2.** Criterion 2: min = −100, max = 500, so max − min = 600. Alternative 1: (500 + 100)/600 = **1**. Alternative 2: (−100 + 100)/600 = **0**. Alternative 3: (100 + 100)/600 = **0.333333**.

**Step 3.** Criterion 3: min = 50, max = 80, so max − min = 30. Alternative 1: (80 − 50)/30 = **1**. Alternative 2: (60 − 50)/30 = **0.333333**. Alternative 3: (50 − 50)/30 = **0**.

**The standardized table.** Every chapter of this tutorial starts from it:

| | Criterion 1 | Criterion 2 | Criterion 3 |
|---|---|---|---|
| Alternative 1 | 0 | 1 | 1 |
| Alternative 2 | 1 | 0 | 0.333333 |
| Alternative 3 | 0.6 | 0.333333 | 0 |

> **Important.** Standardization 1 uses the best and the worst value **among the alternatives present**. Add or remove an alternative and the scale changes – which is the root of the rank-reversal question of chapter 13. Standardization 0 (use the values as given) avoids it when the values are already comparable.

## 2.3 Preparation step 2 – the weights

The weights say how important each criterion is: 0.4, 0.2 and 0.4 here. They are relative, so the model divides them by their total within each group of criteria; 4, 2, 4 would mean exactly the same as 0.4, 0.2, 0.4. In our example they already add to 1, so the shares are 0.4, 0.2 and 0.4.

Two things are worth noticing now, because three methods depend on them. First, criteria 1 and 3 have the **same** weight – they are tied in importance. Second, the weights are ordinary numbers here; chapters 8 and 9 show what to do when you can only say *criterion 1 matters more than criterion 2* without putting a number on it.

# 3. Weighted Average Method (WAM)

## 3.1 The idea

Give every alternative a score: multiply how well it does on each criterion by the importance of that criterion, and add up. The largest total wins. This is the oldest and simplest idea in multi-criteria evaluation, and the one every other method is compared against.

Its assumption is **full compensation**: a bad performance on one criterion can always be bought off by a good performance on another, provided the weights allow it. That is reasonable for money, and questionable for safety or for an irreversible environmental loss – which is why the outranking methods of chapters 5, 6 and 12 exist.

## 3.2 What it needs

- Measured values for every alternative on every criterion (numbers, not ranks).
- A weight for every criterion.
- A standardization, because the values are added together.

## 3.3 The steps

**Step 1.** Standardize the decision table (section 2.2).

**Step 2.** Multiply every standardized value by the weight of its criterion.

**Step 3.** Add the products of each row. That total is the ranking value: **V(a) = w1 × r1(a) + w2 × r2(a) + w3 × r3(a)**.

**Step 4.** Sort from the largest total to the smallest.

## 3.4 Solved example

Step 2, the weighted values (weights 0.4, 0.2, 0.4 applied to the standardized table of section 2.2):

| | 0.4 × Criterion 1 | 0.2 × Criterion 2 | 0.4 × Criterion 3 | Total V(a) |
|---|---|---|---|---|
| Alternative 1 | 0.4 × 0 = 0 | 0.2 × 1 = 0.2 | 0.4 × 1 = 0.4 | **0.6** |
| Alternative 2 | 0.4 × 1 = 0.4 | 0.2 × 0 = 0 | 0.4 × 0.333333 = 0.133333 | **0.533333** |
| Alternative 3 | 0.4 × 0.6 = 0.24 | 0.2 × 0.333333 = 0.066667 | 0.4 × 0 = 0 | **0.306667** |

**The answer** (the largest value is best):

| Rank | Alternative | Ranking value | Domination |
|---|---|---|---|
| 1 | Alternative 1 | 0.6 | 11.111 % |
| 2 | Alternative 2 | 0.533333 | 42.500 % |
| 3 | Alternative 3 | 0.306667 | 0 % |

*Domination* is how far an alternative is ahead of the next one, as a percentage: alternative 1 leads alternative 2 by 11 %, and alternative 2 leads alternative 3 by 42 %. The first two are close; the third is clearly behind. Say that in your report – a rank alone hides it.

## 3.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **Weighted Average Method (WAM)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 3.6 When to use it, and the usual mistakes

- **Use it** whenever the criteria are measured, the weights are meaningful numbers, and trade-offs between criteria are acceptable. Always run it, even when another method is your main one: it is the reference everybody understands.
- **Mistake 1 – forgetting to standardize.** With standardization 0 on this table, criterion 1 (tens of thousands) would swamp criterion 3 (tens), whatever the weights say.
- **Mistake 2 – reading the ranking value as a score out of 1.** It is only comparable inside one run, with one set of settings.
- **Mistake 3 – hiding a near-tie.** 0.6 against 0.533 is not a decisive victory. Chapter 14 and the Sensitivity step of the website tell you how fragile it is.

# 4. Compromise Programming (CP)

## 4.1 The idea

Imagine the perfect alternative: the best value of every criterion at once. It usually does not exist – it is an *ideal point*. Compromise Programming measures how far each real alternative is from that ideal, and the **smallest distance wins**. The method asks a different question from WAM: not *how much value do I collect*, but *how little do I give up*.

The exponent s says how much a large shortfall on a single criterion should hurt. With **s = 1** all shortfalls are simply added. With **s = 2** they are squared first, so one big weakness costs more than several small ones. Larger s means less willingness to compensate.

## 4.2 What it needs

- Measured values and weights, as for WAM.
- An exponent s: 1 or 2.
- An ideal and an anti-ideal for every criterion – either taken from the alternatives present (the default when no limits are given) or **fixed in advance** in the input file (the Upper and Lower limit values).

> **Important.** This choice matters more than it looks. Ideal points computed from the alternatives move when the set of alternatives changes; ideal points fixed in advance do not, and the ranking then cannot reverse when an alternative is added or removed. That is the property discussed in chapter 13, and it is why the fixed-limits form is worth the effort of choosing the limits.

## 4.3 The steps

**Step 1.** Standardize the decision table.

**Step 2.** For every criterion find the ideal (the best standardized value) and the anti-ideal (the worst).

**Step 3.** For every alternative and every criterion compute the relative shortfall: **d = (ideal − r) / (ideal − anti)**. It is 0 for the best alternative on that criterion and 1 for the worst.

**Step 4.** Multiply each shortfall by the weight of its criterion and raise it to the power s: **(w × d) raised to s**.

**Step 5.** Add over the criteria. That total is the ranking value: **L(a) = sum over the criteria of (w × d) to the power s**.

**Step 6.** Sort from the smallest total to the largest.

The classical formula takes the s-th root of the total. MCEWR follows the 1995 program and does not, because the root does not change the order; the switch *CP: the s-th root of the distance* adds it if you want the published form of the number.

## 4.4 Solved example with s = 1

The standardized table already runs from 0 to 1 on every criterion, so ideal = 1, anti-ideal = 0, and the shortfall is simply d = 1 − r:

| | d on criterion 1 | d on criterion 2 | d on criterion 3 | L(a) = Σ w × d |
|---|---|---|---|---|
| Alternative 1 | 1 − 0 = 1 | 1 − 1 = 0 | 1 − 1 = 0 | 0.4×1 + 0.2×0 + 0.4×0 = **0.4** |
| Alternative 2 | 1 − 1 = 0 | 1 − 0 = 1 | 1 − 0.333333 = 0.666667 | 0.4×0 + 0.2×1 + 0.4×0.666667 = **0.466667** |
| Alternative 3 | 1 − 0.6 = 0.4 | 0.666667 | 1 | 0.4×0.4 + 0.2×0.666667 + 0.4×1 = **0.693333** |

**The answer with s = 1** (the smallest value is best): alternative 1 (0.4), then alternative 2 (0.466667), then alternative 3 (0.693333). Notice that 0.4 = 1 − 0.6 and 0.466667 = 1 − 0.533333: with this standardization and ideal points taken from the data, CP with s = 1 is the exact mirror of WAM. The two methods are the same method seen from the two ends. That is worth knowing before quoting them as two independent confirmations.

## 4.5 Solved example with s = 2

Now each weighted shortfall is squared before adding:

| | (w × d) squared, criterion by criterion | L(a) |
|---|---|---|
| Alternative 1 | $(0.4\times1)^2 + (0.2\times0)^2 + (0.4\times0)^2 = 0.16 + 0 + 0$ | **0.16** |
| Alternative 2 | $(0.4\times0)^2 + (0.2\times1)^2 + (0.4\times0.666667)^2 = 0 + 0.04 + 0.071111$ | **0.111111** |
| Alternative 3 | $(0.4\times0.4)^2 + (0.2\times0.666667)^2 + (0.4\times1)^2 = 0.0256 + 0.017778 + 0.16$ | **0.203378** |

**The answer with s = 2**: alternative **2** first (0.111111), alternative 1 second (0.16), alternative 3 last (0.203378). **The winner has changed.** Alternative 1 is perfect on two criteria but is the worst on criterion 1; squaring makes that single large weakness dominate. Alternative 2 is more balanced. This is not a defect of either exponent – it is the question you are asking: *the highest total* (s = 1) or *no serious weakness* (s = 2).

## 4.6 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **Compromise Programming (CP)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** *Method options*: set `cp_power` to **1** (then repeat with **2**), and **untick `cp_use_provided_limits`**.

**Step 6.** The untick matters: this example file carries Upper = Lower = 5, which are not real limits. Left ticked, every distance is zero and the result is meaningless. When your own file has proper limits, tick it – the ranking is then stable against adding or removing alternatives.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 4.7 When to use it, and the usual mistakes

- **Use it** when you want to be near an agreed ideal rather than to collect a maximum, and especially when you can fix the ideal and the anti-ideal in advance from standards, regulations or targets.
- **Mistake 1 – leaving meaningless limits switched on**, as above. Check the Upper and Lower columns in step 3 (Criteria) before ticking that box.
- **Mistake 2 – comparing L values across s.** The numbers of s = 1 and s = 2 are on different scales; only the orders are comparable.
- **Mistake 3 – forgetting that CP with s = 1 mirrors WAM** on standardized data, and presenting them as two independent methods.

# 5. PROMETHEE II (and PROMETHEE I)

## 5.1 The idea

Stop adding scores; compare the alternatives **two at a time**, as a jury would. For every pair and every criterion, ask: is a better than b, and by enough to matter? Add the weights of the criteria on which a wins – that is how strongly a is preferred to b. Do it for every ordered pair, then measure how much each alternative beats the others (its *leaving flow*) and how much it is beaten by them (its *entering flow*). The difference is the net flow, and the largest net flow wins.

*By enough to matter* is where PROMETHEE is subtle. Each criterion carries a **preference function** that turns a difference into a degree of preference between 0 and 1: the *usual* function (number 1) counts any difference, however small, as a full preference; the others need the difference to exceed an indifference threshold q, and reach full preference only above a preference threshold p. Our example uses the usual function on every criterion, which keeps the arithmetic easy to follow.

## 5.2 What it needs

- Measured values and weights.
- A preference function per criterion (1 to 6), with its thresholds p and q where the function uses them.
- No standardization is strictly required – the thresholds are in the units of the criterion – but standardizing is common and is what this example does.

## 5.3 The steps

**Step 1.** For every ordered pair (a, b) and every criterion, compute the degree of preference of a over b. With the usual function it is 1 when a is strictly better and 0 otherwise.

**Step 2.** Multiply by the weights and add over the criteria. This is the **preference index P(a,b)**: the share of the weight on which a beats b.

**Step 3.** **Leaving flow**: for each a, add P(a,b) over all other b and divide by the number of alternatives.

**Step 4.** **Entering flow**: for each a, add P(b,a) over all other b and divide by the number of alternatives.

**Step 5.** **Net flow** = leaving flow – entering flow. Sort from the largest to the smallest. (The 1995 program divides by the number of alternatives n; the text-book switch *PROMETHEE flows divided by n – 1* uses the modern divisor. Neither changes the order.)

## 5.4 Solved example

Step 2, the preference indices. Read the first line as: *alternative 1 beats alternative 2 on criteria 2 and 3, whose weights are 0.2 and 0.4, so P(1,2) = 0.6*.

| Pair | Criteria on which the first beats the second | Preference index |
|---|---|---|
| 1 over 2 | criterion 2 (0.2) and criterion 3 (0.4) | **0.6** |
| 1 over 3 | criterion 2 (0.2) and criterion 3 (0.4) | **0.6** |
| 2 over 1 | criterion 1 (0.4) | **0.4** |
| 2 over 3 | criterion 1 (0.4) and criterion 3 (0.4) | **0.8** |
| 3 over 1 | criterion 1 (0.4) | **0.4** |
| 3 over 2 | criterion 2 (0.2) | **0.2** |

Steps 3 to 5, the flows (three alternatives, so divide by 3):

| | Leaving flow | Entering flow | Net flow |
|---|---|---|---|
| Alternative 1 | (0.6 + 0.6) / 3 = 0.4 | (0.4 + 0.4) / 3 = 0.266667 | **0.133333** |
| Alternative 2 | (0.4 + 0.8) / 3 = 0.4 | (0.6 + 0.2) / 3 = 0.266667 | **0.133333** |
| Alternative 3 | (0.4 + 0.2) / 3 = 0.2 | (0.6 + 0.8) / 3 = 0.466667 | **−0.266667** |

**The answer** (the largest net flow is best): alternatives 1 and 2 are **tied** at 0.133333, and alternative 3 is last at −0.266667. The tie is not a failure of the method; it is the honest result. Alternative 1 wins its pairs by 0.6 and is beaten by 0.4; alternative 2 wins by 0.4 and 0.8 and is beaten by 0.6 and 0.2. The books balance exactly.

### PROMETHEE I – the partial ranking

PROMETHEE II forces a complete order by subtracting the two flows. PROMETHEE I keeps them apart and reports four possible relations for each pair: a is **preferred** to b when it is at least as good on both flows and strictly better on one; they are **indifferent** when both flows are equal; they are **incomparable** when one has the better leaving flow and the other the better entering flow; otherwise the pair is undecided.

On this example alternatives 1 and 2 have the same leaving flow (0.4) and the same entering flow (0.266667), so PROMETHEE I calls them **indifferent** – genuinely equivalent on this evidence – and both are preferred to alternative 3. Tick *also the PROMETHEE I partial ranking* on the website to see these relations for every pair.

## 5.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **PROMETHEE II**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** Tick **also the PROMETHEE I partial ranking** to get the pairwise relations as well.

**Step 6.** The published six-alternative example of Brans, Vincke and Mareschal (1986) is the file `PROM_EX.MCE`; open it with standardization **0** and the box *use the weights as given* **ticked** to reproduce the article's table.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 5.6 When to use it, and the usual mistakes

- **Use it** when small differences should not count as real preferences, when the criteria have their own natural thresholds, or when you want to be told that two alternatives are simply not comparable rather than being given an artificial order.
- **Mistake 1 – leaving the usual function everywhere.** It is the crudest setting: a difference of 0.001 counts as much as a difference of 1. If you know that differences below some size are meaningless, say so with q and p.
- **Mistake 2 – setting thresholds in the wrong units.** With standardization 1 the data run from 0 to 1, so a threshold of 10 never bites. Set thresholds either on raw data (standardization 0) or on the 0–1 scale.
- **Mistake 3 – hiding a tie or an incomparability** behind the complete PROMETHEE II order. Report the partial ranking as well.

# 6. ELECTRE II

## 6.1 The idea

ELECTRE also compares pairs, but it adds a right of veto. For a pair (a, b) it asks two questions. **Concordance**: what share of the weight agrees that a is at least as good as b? **Discordance**: on the criteria where b is better, how large is the worst difference? If enough of the weight agrees *and* no single criterion objects too strongly, we say that **a outranks b**. Nothing is added up; a very bad performance cannot be compensated, only vetoed.

The outranking relations form a graph, which is then turned into a ranking. ELECTRE II builds two graphs – a *strong* one with severe thresholds and a *weak* one with milder thresholds – ranks the alternatives in each, and takes the average of the two positions.

## 6.2 What it needs

- Measured values and weights.
- Three concordance levels p*, p0, p– (strong, average, weak) and two discordance levels q*, q0. In our file: p* = 0.7, p0 = 0.6, p– = 0.55, q* = 0.55, q0 = 0.3.
- A scale for the discordance – the standardized 0–1 range here.

## 6.3 The steps

**Step 1.** Standardize the decision table.

**Step 2.** **Concordance matrix**: for every ordered pair (a, b), add the weights of the criteria on which a is strictly better than b. Divide by the total weight (1 here).

**Step 3.** **Discordance matrix**: for every ordered pair, take the criteria on which b is better than a and keep the largest difference, expressed on the criterion's scale.

**Step 4.** **Strong graph**: draw an arrow a → b when the concordance is at least p* and the discordance is at most q*, and a beats b at least as much as b beats a.

**Step 5.** **Weak graph**: the same with the milder levels p0 / p– and q0.

**Step 6.** Rank the alternatives in each graph (an alternative that nothing outranks is first, and so on), then take the **median of the two positions** as the ranking value. The smallest value is best.

## 6.4 Solved example

Step 2, the concordance matrix. It is the same arithmetic as the PROMETHEE preference indices of chapter 5 – the share of the weight on which the row beats the column:

| Concordance C(row, column) | Alternative 1 | Alternative 2 | Alternative 3 |
|---|---|---|---|
| Alternative 1 | — | 0.6 | 0.6 |
| Alternative 2 | 0.4 | — | 0.8 |
| Alternative 3 | 0.4 | 0.2 | — |

Step 3, the discordance matrix. For the pair (1, 2): alternative 2 is better only on criterion 1, where the standardized difference is 1 − 0 = 1, so the discordance is 1. For the pair (2, 3): alternative 3 is better only on criterion 2, by 0.333333 − 0 = 0.333333.

| Discordance D(row, column) | Alternative 1 | Alternative 2 | Alternative 3 |
|---|---|---|---|
| Alternative 1 | — | 1 | 0.6 |
| Alternative 2 | 1 | — | 0.333333 |

| Discordance D(row, column) | Alternative 1 | Alternative 2 | Alternative 3 |
|---|---|---|---|
| Alternative 3 | 1 | 0.4 | — |

Steps 4 and 5, the graphs. Test every pair against the levels:

- Pair (2, 3): concordance 0.8 ≥ p* = 0.7 **and** discordance 0.333333 ≤ q* = 0.55 → **2 outranks 3**, in the strong graph and in the weak graph.
- Pair (1, 2): concordance 0.6 is respectable, but the discordance is 1 – criterion 1 objects with the maximum possible force. **Vetoed.** No arrow.
- Pair (1, 3): concordance 0.6, discordance 0.6, above q* = 0.55. **Vetoed.** No arrow.
- All other pairs fail the concordance test.

So the only relation the data support is *alternative 2 outranks alternative 3*. Step 6 ranks the graphs. In the strong graph nothing outranks alternative 1 or alternative 2, so both are first and alternative 3 is second. In the weak graph the same single arrow puts alternative 2 first, with 1 and 3 second.

| | Position in the strong graph | Position in the weak graph | Median = ranking value |
|---|---|---|---|
| Alternative 1 | 1 | 2 | **1.5** |
| Alternative 2 | 1 | 1 | **1** |
| Alternative 3 | 2 | 2 | **2** |

**The answer** (the smallest value is best): alternative **2** first, alternative 1 second, alternative 3 last. ELECTRE II disagrees with WAM – and says why. Alternative 1 collects the highest total, but on criterion 1 it is as bad as an alternative can be, and ELECTRE refuses to let two good performances erase that. Alternative 2 has no such weakness.

## 6.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **ELECTRE II**.

**Step 4.** *Standardization*: **1 (v – min) / (max – min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** The five levels come from the file (the group line). To experiment, save the problem as `.mcewr`, edit the levels, and open it again.

**Step 6.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 6.6 When to use it, and the usual mistakes

- **Use it** when some criteria must not be compensated – a safety limit, an environmental threshold, a legal minimum – and when you accept that the method may leave alternatives tied rather than invent an order.
- **Mistake 1 – error 201.** If the outranking relations form a cycle (a outranks b outranks c outranks a) the graph cannot be ranked. Raise the concordance levels or lower the discordance levels, or use ELECTRE III (chapter 12), whose distillations always produce a ranking.
- **Mistake 2 – levels copied from another study.** p*, p0, p–, q*, q0 belong to the problem, not to the method. Levels that are too severe give an empty graph (everything tied); levels that are too mild give arrows everywhere.
- **Mistake 3 – expecting the discordance to be in the original units.** It is measured on the standardized scale when the data are standardized.

# 7. Analytic Hierarchy Process (AHP)

## 7.1 The idea

Sometimes nobody can put a number on a criterion, but everybody can compare two things at a time: *access matters a little more than cost*, *site B is clearly better than site A for the community*. AHP is built on that. You compare the items in pairs on a 1-to-9 scale, and the method turns the whole set of judgements into priorities that add to 1. It then checks whether your judgements were consistent.

The 1-to-9 scale: 1 = the two are equally good, 3 = slightly better, 5 = clearly better, 7 = much better, 9 = extremely better (2, 4, 6, 8 are the intermediate steps). If a is 4 compared with b, then b is 1/4 compared with a – the matrix is filled in automatically from the half you give.

## 7.2 What it needs

- Either pairwise judgements (the *Saaty* mode) or ordinary measured values – MCEWR accepts both, criterion by criterion.
- Weights, which may themselves come from pairwise judgements between the criteria.

## 7.3 The steps

**Step 1.** Write the comparison matrix: 1 on the diagonal, your judgement above it, the reciprocal below it.

**Step 2.** Compute the priorities. By hand: divide every column by its total, then average each row. Exactly: take the principal eigenvector of the matrix, which is what MCEWR computes.

**Step 3.** Check the consistency. Compute the largest eigenvalue, then **CI = (largest eigenvalue − n) / (n − 1)**, and the consistency ratio **CR = CI / RI**, where RI is a published constant for the size of the matrix (0.58 for a 3×3). **A consistency ratio above 0.10 means the judgements contradict each other and should be revised.**

**Step 4.** Do this for the alternatives under every criterion, and for the criteria themselves, then combine: the priority of an alternative is the sum over the criteria of (weight of the criterion × its priority under that criterion).

## 7.4 Solved example

From the file `AHP_EXM3.MCE` (Saaty's own example), the three alternatives compared under criterion 1. Read the second row as: *alternative 2 is 4 times preferred to alternative 1, and 3 times preferred to alternative 3*.

| | Alternative 1 | Alternative 2 | Alternative 3 |
|---|---|---|---|
| Alternative 1 | 1 | 1/4 = 0.25 | 1/2 = 0.5 |
| Alternative 2 | 4 | 1 | 3 |
| Alternative 3 | 2 | 1/3 = 0.333333 | 1 |

Step 2 by hand. The column totals are 7, 1.583333 and 4.5. Divide each entry by its column total and average each row:

| | Column 1 ÷ 7 | Column 2 ÷ 1.583333 | Column 3 ÷ 4.5 | Row average = priority |
|---|---|---|---|---|
| Alternative 1 | 0.142857 | 0.157895 | 0.111111 | **0.137288** |
| Alternative 2 | 0.571429 | 0.631579 | 0.666667 | **0.623225** |
| Alternative 3 | 0.285714 | 0.210526 | 0.222222 | **0.239488** |

MCEWR uses the exact eigenvector rather than this average, and reports **0.136, 0.625, 0.238** – the same answer to three decimals. The column-average is the classical hand approximation; the difference between the two is itself a measure of how consistent the matrix is.

Step 3, the consistency. The largest eigenvalue of this matrix is 3.018295, so

```
CI = (3.018295 − 3) / (3 − 1) = 0.009147
CR = CI / RI = 0.009147 / 0.58 = 0.015771
```

which is the value MCEWR prints in the summary. It is far below 0.10, so the judgements hold together: if alternative 2 is 4 times alternative 1 and 3 times alternative 3, then alternative 3 being 2 times alternative 1 is coherent.

**The answer** for the whole file, after combining the six criteria with their own pairwise weights: alternative 1 first (0.401928), alternative 2 second (0.344462), alternative 3 third (0.25361).

> **Note.** Run AHP on our three-criteria example with ordinary measured values and you get 0.6, 0.533333 and 0.306667 – exactly the WAM result of chapter 3. That is correct and worth understanding: with measured values and given weights, AHP's aggregation *is* a weighted sum. AHP earns its keep when the input is judgement rather than measurement.

## 7.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `AHP_EXM3.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **Analytic Hierarchy Process (AHP)**.

**Step 4.** *Standardization*: **0  none: the data as given**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** Steps 3 and 4 show *Saaty* instead of numbers, because the values are pairwise judgements; they are edited in the file, not in the table.

**Step 6.** Read the consistency ratio of every matrix in the **Summary** under the ranking. Any value above 0.10 is a warning about your judgements, not about the program.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 7.6 When to use it, and the usual mistakes

- **Use it** when the evidence is expert judgement, when the criteria form a hierarchy, or when you need to document *why* a weight is what it is – the pairwise judgements are auditable.
- **Mistake 1 – ignoring the consistency ratio.** Above 0.10 the priorities are built on contradictory statements. Find the pair that causes it and revise the judgement.
- **Mistake 2 – comparing too many things at once.** Nine items make 36 comparisons and consistency collapses. Group them into a hierarchy of at most seven items per level.
- **Mistake 3 – treating the 1-to-9 numbers as measurements.** They are ordered labels of judgement; do not average them with real data.

# 13. Weights: objective schemes, equal weights, Best-Worst

## 13.1 The idea

Every method in the previous chapters needed weights, and every weight was a judgement. Sometimes there is no one qualified to make that judgement, or a study must show what the numbers alone would say. **Objective weights** are computed from the decision table itself, without asking anybody.

They answer a narrow question – *which criterion carries the most information about the differences between these alternatives?* – and not the question *which criterion matters most to society*. A criterion on which every alternative scores the same tells you nothing and receives almost no weight, however important it may be in reality. Keep that limit in mind: objective weights are a useful check on your own weights, and a poor substitute for them.

MCEWR offers three schemes. They are on the **3. Weights** tab, they are computed as soon as a problem is loaded, and any of them can be used with any of the ten methods.

## 13.2 What each of the three rewards

| Scheme | A criterion receives a large weight when … | It is blind to |
|---|---|---|
| **Entropy** | its values are spread unevenly over the alternatives – the column carries much information in the sense of Shannon | how the criterion relates to the other criteria |
| **CRITIC** | its values are dispersed **and** little correlated with the other criteria: it says something the others do not | the absolute size of the spread, because the columns are rescaled to 0–1 first (see section 13.5) |
| **MEREC** | removing it changes the overall performance of the alternatives a great deal | the same as entropy: relations between criteria |

## 13.3 What each of them needs – and what happens when the data do not oblige

Entropy takes logarithms of shares, and MEREC takes logarithms of ratios, so both need **strictly positive values**. CRITIC needs only that the alternatives differ. Our example fails the first requirement: criterion 2 contains the value −100. The Weights step (step 5) therefore shows:

| Group | Criterion | Type | Given | Entropy | CRITIC | MEREC |
|---|---|---|---|---|---|---|
| main criteria | 1 | max | 0.4000 | — | 0.4700 | — |
| main criteria | 2 | max | 0.2000 | — | 0.2816 | — |
| main criteria | 3 | max | 0.4000 | — | 0.2484 | — |

with two notes printed underneath: *entropy: criterion values must be non-negative with a positive sum* and *MEREC: criterion values must be strictly positive*. The program refuses rather than inventing a number, and says why. That is the behaviour to expect throughout MCEWR.

> **Important.** The requirement is on the **raw** values, not on the standardized ones, and it applies to the whole group of criteria: one negative column blocks the scheme for the group. If your criterion can be negative – a net benefit, a temperature, a change – shift it by a constant, or use CRITIC.

## 13.4 CRITIC, solved by hand

**Step 1.** Standardize every criterion to 0–1 in the direction of its type – the same (v − min)/(max − min) as section 2.2. CRITIC always does this itself, whatever standardization the Evaluate tab uses.

**Step 2.** For every criterion compute the standard deviation of its standardized column. This is the *contrast*.

**Step 3.** For every pair of criteria compute the linear correlation r between their standardized columns.

**Step 4.** For each criterion add up (1 − r) over all criteria, itself included. This is the *conflict*: a criterion that moves against the others gets a large sum, a criterion that repeats what another says gets a small one.

**Step 5.** Multiply contrast by conflict: **C = standard deviation × sum of (1 − r)**.

**Step 6.** Divide each C by the total of the three, so that the weights add to 1.

Steps 1 and 2 on our example. The standardized table is the one from section 2.2, and its columns have standard deviations 0.410961, 0.415740 and 0.415740.

Steps 3 and 4, the correlations. Criterion 1 runs 0, 1, 0.6 while criterion 2 runs 1, 0, 0.333333 – they move in opposite directions, so r is strongly negative:

| Correlation r | with criterion 1 | with criterion 2 | with criterion 3 | Sum of (1 − r) |
|---|---|---|---|---|
| Criterion 1 | 1 | −0.997176 | −0.737043 | 0 + 1.997176 + 1.737043 = **3.734220** |
| Criterion 2 | −0.997176 | 1 | 0.211324 | 1.997176 + 0 + 0.788676 = **2.211462** |
| Criterion 3 | −0.737043 | 0.211324 | 1 | 1.737043 + 0.788676 + 0 = **1.951329** |

Steps 5 and 6:

```
C1 = 0.410961 × 3.734220 = 1.534619
C2 = 0.415740 × 2.211462 = 0.919393
C3 = 0.415740 × 1.951329 = 0.811245
total = 3.265257

weights = 1.534619 / 3.265257 = 0.4700    0.919393 / 3.265257 = 0.2816    0.811245 / 3.265257 =
0.2484
```

which is exactly the CRITIC column of the table in section 13.3. Criterion 1 receives almost half the weight, because it disagrees with both other criteria and therefore carries information they do not.

### Does it change the answer?

Run WAM again with the CRITIC weights instead of the declared 0.4 / 0.2 / 0.4:

| | With the declared weights | With the CRITIC weights |
|---|---|---|
| Alternative 1 | 0.6 — **first** | 0.530016 — second |
| Alternative 2 | 0.533333 — second | 0.5528 — **first** |
| Alternative 3 | 0.306667 — third | 0.375847 — third |

**The winner changes.** CRITIC gives criterion 1 more weight than the expert did (0.47 instead of 0.40), and alternative 2 – the leader on criterion 1 – overtakes alternative 1. This is not a verdict that the expert was wrong. It is a precise statement of *how much* the answer depends on that one weight, which is exactly what chapter 14 measures directly.

## 13.5 Entropy and MEREC on a problem you can type in

To see the two schemes our example cannot compute, make a small strictly positive problem of your own. On the website: step **1 Project** → **New problem...** → 3 alternatives and 3 criteria, then type these values into the pay-off matrix (all criteria maximizing, all weights left at 1):

| | Criterion A | Criterion B | Criterion C |
|---|---|---|---|
| Alternative 1 | 10 | 20 | 30 |
| Alternative 2 | 20 | 20 | 20 |

| | Criterion A | Criterion B | Criterion C |
|---|---|---|---|
| Alternative 3 | 30 | 21 | 10 |

Criterion A rises, criterion C falls in the mirror image of A, and criterion B is practically constant – it hardly separates the alternatives at all. Now open the **3. Weights** tab:

| Criterion | Entropy | CRITIC | MEREC |
|---|---|---|---|
| A | **0.4992** | 0.2568 | **0.4917** |
| B | **0.0015** | 0.2779 | **0.0121** |
| C | **0.4992** | 0.4653 | **0.4961** |

### Entropy, step by step

**Step 1.** Divide each value by the total of its criterion, giving shares that add to 1. Criterion A: 10/60, 20/60, 30/60 = 0.1667, 0.3333, 0.5.

**Step 2.** Compute the entropy of the column: **E = − (sum of p × ln p) / ln m**, where m is the number of alternatives (3 here, so ln 3 = 1.098612). For criterion A this gives E = 0.920620.

**Step 3.** The information carried is **d = 1 − E**. Criterion A: 0.079380. Criterion B: its shares are 0.3279, 0.3279, 0.3443 – almost equal – so its entropy is 0.999757 and d is only **0.000243**.

**Step 4.** Divide each d by the total of the three: 0.079380 / 0.159003 = **0.4992** for A and for C, and 0.000243 / 0.159003 = **0.0015** for B.

### MEREC, step by step

**Step 1.** Normalize each maximizing criterion by dividing its smallest value by each value (so the best alternative gets the smallest number), then take the absolute logarithm of each.

**Step 2.** Compute the overall performance of every alternative from **all** the criteria: S = ln(1 + the mean of those logarithms). Here S = 0.312036, 0.379872 and 0.323870.

**Step 3.** Remove one criterion and compute the performances again. Without criterion A: 0.312036, 0.207867, 0.016133.

**Step 4.** Add up how much every alternative moved: that is the effect of removing the criterion. Without A the total movement is 0.479743; without B only **0.011834**; without C, 0.484042.

**Step 5.** Divide each effect by the total: **0.4917, 0.0121, 0.4961**.

### The lesson in this table

Entropy and MEREC agree almost exactly: criterion B is nearly useless for telling these three alternatives apart, and both give it about one per cent of the weight. **CRITIC gives it 0.2779** – more than a quarter. Why? Because CRITIC rescales every column to 0–1 before measuring the spread, so criterion B's range of one single unit is stretched to the same 0-to-1 range as criterion A's range of twenty. After rescaling, B looks as dispersed as the others, and its low correlation with them then earns it a large weight.

> **Note.** This is not a defect to hide: it is what the method is. CRITIC asks *does this criterion separate the alternatives in a way the others do not?* and answers on the shape of the column, not on its size. When the physical size of the differences matters, read entropy and MEREC as well. Reporting all three – as the Weights step does – is the honest practice.

## 13.6 Using them, and the usual mistakes

**Step 1.** Open the problem and look at the **3. Weights** tab before choosing anything: it shows your declared weights and the three objective ones side by side.

**Step 2.** Where they agree, your weights are supported by the data. Where they differ sharply – as criterion 1 does here – you have found the number your conclusion depends on.

**Step 3.** To rank with an objective scheme, go to the **2. Evaluate** tab and choose it in the *Weights* box, then run as usual. The scheme's name appears in the summary and in the report.

**Step 4.** Report both runs. *With the expert weights the order is A, B, C; with CRITIC weights it is B, A, C* is a far stronger sentence than either result alone.

- **Mistake 1 – treating objective weights as the truth.** They measure information in a table, not importance in the world. A criterion that all alternatives satisfy equally – a legal minimum, for instance – gets almost no weight, and may still be the most important thing in the study.
- **Mistake 2 – using them on data with a different number of alternatives.** All three schemes depend on the set of alternatives present; add a site and the weights move.
- **Mistake 3 – forgetting the data requirements** and concluding that the program is broken when the entropy and MEREC columns show dashes. Read the note under the table: it names the requirement that was not met.
- **Mistake 4 – mixing them with Saaty weights.** A group whose weights come from pairwise judgements keeps those weights; the objective schemes apply to ordinary numeric groups.

## 13.7 Equal weights: the neutral starting point

Every criterion of a group receives the same weight 1/n. This is not a method but a reference point: run the study with equal weights first, then with the weights you believe in, and see what the weights changed. On the example of chapter 2 (three criteria, weights 0.4, 0.2, 0.4) the equal weights are 1/3 each; WAM with standardization 1 then gives 0.6667 for alternative 1, 0.4444 for alternative 2 and 0.3111 for alternative 3. The order does not change, but the gap between the first two widens from 0.067 to 0.222: criterion 2, on which alternative 1 is best and alternative 2 worst, now weighs 1/3 instead of 0.2. Choose the card **Equal weights** in step 5 of the website.

## 13.8 The Best-Worst Method: fewer questions, a linear program

A full pairwise comparison of n criteria (the Saaty matrix of AHP) asks n(n − 1)/2 questions and is hard to keep consistent. The **Best-Worst Method** of Rezaei (2015) asks 2n − 3. The decision maker names the **best** (most important) criterion B and the **worst** (least important) criterion W, then answers on the 1–9 scale (1 = equally important, 3 = moderately, 5 = strongly, 7 = very strongly, 9 = extremely more important): *how much more important is the best than each criterion?* – the Best-to-Others vector a(B,j) – and *how much more important is each criterion than the worst?* – the Others-to-Worst vector a(j,W). The comparison of the best with the worst, a(B,W), appears in both vectors.

Perfect judgements would give weights with w(B)/w(j) = a(B,j) and w(j)/w(W) = a(j,W) exactly. Real judgements contradict one another, so the weights are chosen to make the largest violation as small as possible. The original model minimizes the largest ratio error and can have several optimal weight vectors; the **linear model** of Rezaei (2016) minimizes the largest difference error and has one:

```
minimize  xi
subject to  | w(B) - a(B,j) w(j) | <= xi   for every criterion j
            | w(j) - a(j,W) w(W) | <= xi   for every criterion j
            w(1) + ... + w(n) = 1,   w(j) >= 0
```

Each absolute value is two linear inequalities, so this is a linear program with n + 1 unknowns and about 4n constraints; MCEWR solves it with its own simplex routine (`mcewr/lp.py`, no third-party package) and verifies the answer against the closed-form optimum of Ratandhara and Kumar (2025) and an exact-arithmetic simplex. The optimum ξ* is the largest violation of a comparison by the weights: 0 when the judgements are fully consistent – a(B,j)·a(j,W) = a(B,W) for every j, and then w(j) = a(j,W) / Σ a(k,W) exactly – and the larger the more the judgements contradict one another.

### A consistent system: the example of chapter 2

Three criteria, best = criterion 1, worst = criterion 2. Best-to-Others: a(B,1) = 1, a(B,2) = 4, a(B,3) = 2. Others-to-Worst: a(1,W) = 4, a(2,W) = 1, a(3,W) = 2. Every product a(B,j)·a(j,W) equals a(B,W) = 4 (1·4, 4·1, 2·2), so the system is consistent, ξ* = 0, and the weights are the Others-to-Worst vector divided by its sum: **4/7 = 0.5714, 1/7 = 0.1429, 2/7 = 0.2857** – the proportions 4 : 1 : 2, which is what the judgements said. On the website: step 5 → card **Best-Worst Method** → enter these six numbers → **Compute the weights**. WAM with standardization 1 and these weights puts alternative 2 first with 0.6667, then alternative 1 with 0.4286 and alternative 3 with 0.3905 – the same as with the weights 4, 1, 2 typed into step 3, and a different winner from the weights 0.4, 0.2, 0.4 of chapter 2: the judgements matter.

### An inconsistent system, solved by hand

Four criteria; best = criterion 1, worst = criterion 4 (the fourth worked example of Ratandhara and Kumar 2025). Best-to-Others: 1, 5, 4, 8. Others-to-Worst: 8, 4, 1, 1. Check the products: criterion 2 gives 5·4 = 20 and criterion 3 gives 4·1 = 4, against a(B,W) = 8 – the decision maker has placed criterion 2 too high and criterion 3 too low relative to the ends of the scale, so ξ* will be positive.

**Step 1.** **Write the constraints** (a(B,1) = 1 and a(4,W) = 1 give nothing): |w1 − 5w2| ≤ ξ, |w1 − 4w3| ≤ ξ, |w1 − 8w4| ≤ ξ, |w2 − 4w4| ≤ ξ, |w3 − w4| ≤ ξ, w1 + w2 + w3 + w4 = 1.

**Step 2.** **Find the largest contradiction.** Measure the weights in units of w4 (the worst) for a moment. Criterion 2 says w2 = 4 w4 (from the worst) and w1 = 5 w2 = 20 w4 (from the best), while the best-to-worst comparison says w1 = 8 w4. The difference 20 − 8 = 12 must be absorbed by the three comparisons w1 ↔ w4, w1 ↔ w2 and w2 ↔ w4; spread evenly it costs 12 / (5 + 2) = 12/7 per comparison. Criterion 3 is a smaller contradiction (8 − 4 = 4, spread over 4 + 2 = 6 → 2/3), and the pair 2–3 gives (20 − 4)/(5 + 4 + 2) = 16/11: the largest is 12/7, so it decides ξ.

**Step 3.** **Solve the four binding constraints with the sum.** Take w4 = 1 and let the comparisons of criterion 2 all miss by the same 12/7: w1 = 8 + 12/7 = 68/7, w2 = 4 − 12/7 = 16/7, w3 = 1 + 12/7 = 19/7, w4 = 1. Divide by the sum 110/7: **w = 68/110, 16/110, 19/110, 7/110 = 0.6182, 0.1455, 0.1727, 0.0636**, and **ξ* = (12/7)/(110/7) = 12/110 = 0.1091**.

**Step 4.** **Check every constraint** (multiply by 110): |68 − 80| = 12, |68 − 76| = 8, |68 − 56| = 12, |16 − 28| = 12, |19 − 7| = 12 – none exceeds 12, four of the five are exactly 12, and the sum is 110. The simplex of MCEWR finds the same fractions, and the exact-arithmetic run proves that no feasible weights give a smaller ξ.

**Step 5.** **Judge the consistency.** Two ratios accompany the weights. The consistency ratio CR = ξ* / CI(n, a(B,W)), where CI is the largest ξ* any system with 4 criteria and a(B,W) = 8 can produce, 0.2887: CR = 0.1091/0.2887 = **0.378** (0 = perfect, 1 = the worst possible). The input-based ratio of Liang, Brunelli and Rezaei (2020), max |a(B,j) a(j,W) − a(B,W)| / (a(B,W)² − a(B,W)) = 12/56 = **0.214**, needs no optimization. Both say the same: the judgements are usable but not clean – criterion 2 deserves a second look.

> **Note.** What the website shows: the BWM column of the weights table (0.6182, 0.1455, 0.1727, 0.0636), the line "ξ* = 0.1091, consistency ratio 0.378 (CI = 0.2887), input-based ratio 0.214", and the buttons to use these weights as the given weights or to forget the judgements. The judgements are saved in the project file.

### When to use it, and the usual mistakes

- **Use it** when one expert (or a group speaking with one voice) must weight more than four or five criteria: the questions are few and natural, and the two ratios tell at once whether the answers hang together.
- **Mistake 1 – a best-to-worst value that differs between the two vectors.** It is one comparison; the website mirrors it so that it cannot differ.
- **Mistake 2 – reading the objective as an error of the weights.** ξ* measures the judgements, not the weights: a large ξ* means the answers contradict one another, and the cure is to ask again, not to shrink the weights.

- **Mistake 3 – comparing the objective across problems.** $\xi^*$ grows with n and with a(B,W); compare the consistency ratio, which is scaled by the worst case CI(n, a(B,W)).

# 14. How firm is the answer? Sensitivity and rank reversal

## 14.1 Why ask

Chapter 3 produced a winner. Two questions decide whether it is worth reporting. **How much would the weights have to change before the winner changes?** – if the answer is *a little*, the study has produced an opinion, not a result. And **does the order survive a change in the list of alternatives?** – if adding a site reverses two others that were not touched, the ranking is an artefact of the set rather than a property of the alternatives.

MCEWR answers both from the same run, in the steps **9 Sensitivity**, **10 Robustness** and **11 Rank reversal**. This chapter works through them on the same three-alternative example, with WAM and standardization 1.

## 14.2 The weight sweep and the critical share

The sweep varies the share of one criterion from 0 to 1 while the other criteria keep their proportions to one another, and records who wins along the way. Where the winner changes, a bisection finds the exact share: the **critical share**.

**Step 1.** In step 6, run the method whose robustness you want to test (here WAM, standardization 1).

**Step 2.** Open the **5. Sensitivity** tab. *Group*: the main criteria. *Steps of the sweep*: 8. *Perturbation ± %*: 10. *Samples*: 200. *Seed*: 1995.

**Step 3.** Press **Run the sensitivity analysis** and read the upper table.

| Criterion | Base share | Winner by interval of the share | Critical share | Change needed |
|---|---|---|---|---|
| 1 | 0.4000 | 0.000–0.438: alternative 1; 0.438–1.000: alternative 2 | **0.4375** | **9.4 %** |
| 2 | 0.2000 | 0.000–0.143: alternative 2; 0.143–1.000: alternative 1 | **0.1429** | 28.6 % |
| 3 | 0.4000 | 0.000–0.333: alternative 2; 0.333–1.000: alternative 1 | **0.3333** | 16.7 % |

### Checking the first line by hand

Under WAM the value of an alternative is a straight line in the weights, so the crossing point can be computed exactly. Let t be the share of criterion 1; the other two keep their 1-to-2 proportion, so criterion 2 receives (1 − t)/3 and criterion 3 receives 2(1 − t)/3. Using the standardized table of section 2.2:

```
V(alternative 1) = t × 0 + (1 − t)/3 × 1 + 2(1 − t)/3 × 1 = 1 − t
V(alternative 2) = t × 1 + (1 − t)/3 × 0 + 2(1 − t)/3 × 0.333333 = t + 0.222222 × (1 − t)

they are equal when   1 − t = t + 0.222222 − 0.222222 t
                      0.777778 = 1.777778 t
                      t = 7/16 = 0.4375
```

The same algebra gives 1/7 = 0.142857 for criterion 2 and 1/3 = 0.333333 for criterion 3 – the two other lines of the table. The program finds these numbers numerically; you can confirm them with a pen.

### Reading the result

- **Criterion 1 is the weak point.** Its share only has to rise from 0.400 to 0.4375 – a change of 9.4 % – and alternative 2 becomes the winner. Anyone who argues that criterion 1 deserves 45 % of the weight has changed the conclusion of the study.
- **The other two criteria are safe.** Criterion 2 would have to lose more than a quarter of its weight, criterion 3 a sixth, before anything moves.
- **This explains chapter 15.** The methods that put alternative 2 first are, in effect, giving criterion 1 a little more influence than the declared weights do. The disagreement between methods and the fragility of the weight are the same fact seen twice.

## 14.3 Rank acceptability under random perturbation

The sweep moves one criterion at a time. The second half of the tab moves them all at once: it draws 200 random weight vectors within ±10 % of the declared weights, re-ranks every time, and counts the results.

| Alternative | First in … | Second in … | Third in … | Mean rank |
|---|---|---|---|---|
| Alternative 1 | **99.5 %** | 0.5 % | 0 % | 1.00 |
| Alternative 2 | 0.5 % | **99.5 %** | 0 % | 2.00 |
| Alternative 3 | 0 % | 0 % | **100 %** | 3.00 |

Alternative 1 is first in 199 of the 200 draws. Read together with section 14.2, the message is precise: *within ten per cent of the stated weights the answer is stable; the winner changes only if criterion 1 is given about 44 % of the weight or more.* That sentence is worth more in a report than any single ranking.

> **Note.** The draws are random, but the run is repeatable: the **seed** (1995 here) is recorded in the result and in the report, so the same table appears on any computer. Quote the seed and the number of samples whenever you publish a percentage.

## 14.4 Rank reversal: does the answer depend on the list of alternatives?

Most methods measure each alternative against the others – the standardization uses the best and worst values present, TOPSIS builds its ideal from the set, PROMETHEE compares every pair. So the arrival or departure of a third alternative can, in principle, swap two that were not touched. That is **rank reversal**, and the **6. Reversal** tab tests it directly.

| Test | What the program does | A reversal means |
|---|---|---|
| remove | removes each alternative in turn and re-ranks the rest | two of the remaining alternatives swapped places |
| duplicate | adds an exact copy of each alternative in turn | two of the original alternatives swapped places |
| dominated | adds one alternative that is worse than every other on every criterion | the same – a hopeless newcomer changed the order of the others |

On our three-alternative example nothing happens: every method passes every test, 0 reversals out of 3 removals, 3 duplications and 1 dominated alternative. With three well-separated alternatives there is little room for trouble. The effect appears on realistic problems, so the tutorial borrows the Han River example (`COMP_HAN.MCE`, 9 alternatives and 11 criteria) for one table:

| Method | remove | duplicate | dominated | Reversed pairs |
|---|---|---|---|---|
| Weighted Average Method | 0 / 9 | 0 / 9 | 0 / 1 | **0** |
| Compromise Programming (provided limits) | 0 / 9 | 0 / 9 | 0 / 1 | **0** |
| Analytic Hierarchy Process | 0 / 9 | 0 / 9 | 0 / 1 | **0** |
| PROMETHEE II | 1 / 9 | 1 / 9 | 0 / 1 | 2 |
| VIKOR | 1 / 9 | 0 / 9 | 1 / 1 | 10 |
| TOPSIS | **7 / 9** | **8 / 9** | 0 / 1 | **51** |

- **Compromise Programming with the limits given in the file never reverses**, on this or any file: its ideal and anti-ideal are fixed in advance, so removing an alternative cannot move the yardstick. This is the property that makes it the safest choice when the list of alternatives may grow – and it is the same computation that the recent literature calls SPOTIS.

- **TOPSIS reverses most often**, because both of its reference points are recomputed from whichever alternatives are present. Seven removals out of nine changed something, and 51 pairs were affected in total.
- **A reversal is not a bug**, in MCEWR or in the method. It is a documented property, and the tab exists so that you can report it instead of being surprised by it in review.

## 14.5 Putting the three analyses together

**Step 1.** **Run the comparison** (step 8): do the methods agree? Where they do, the finding is method-independent.

**Step 2.** **Run the sensitivity** (step 9): how far can the weights move? Report the critical share of the weakest criterion – here 9.4 % on criterion 1.

**Step 3.** **Run the reversal test** (step 11): if the list of alternatives is provisional, prefer a method that passed, or state that the ranking holds for this set of alternatives only.

**Step 4.** **Write the three answers into your report.** A ranking with these three sentences behind it is a result; a ranking without them is a table.

## 14.6 The usual mistakes

- **Reporting the winner without the critical share.** A 9.4 % margin and a 90 % margin are entirely different findings, and the ranking table looks the same in both cases.
- **Running the analyses with settings that differ from the main run.** The analysis steps use the method and settings of the Evaluate tab; set them first.
- **Publishing a perturbation percentage without the seed and the sample size**, which makes it impossible to reproduce.
- **Treating a reversal-prone method as disqualified.** TOPSIS is a perfectly respectable method; it simply must not be used when alternatives will be added later without re-running everything.
- **Adjusting the weights until the preferred alternative wins.** The sensitivity analysis makes this visible to any reader – which is precisely why it should be run and reported by the author first.

## 14.7 Version 2.1: the decision values, one at a time

The weights are not the only uncertain input. Step 9, part **Decision values**, asks the same question of the data: how much must one value change before the winner changes? Each value of the chosen alternative (by default the winner) is moved from *value − range* to *value + range* of its criterion, the method is re-run, and a bisection finds the nearest **critical value**. The change is reported as a share of the criterion's range, so that a value of 0 and values in different units can be compared.

On the example (WAM, standardization 1, winner alternative 1): on criterion 1 no value of alternative 1 within the sweep changes the winner; on criterion 2 the value must fall from 500 to 33.3 (77.8 % of the range) before alternative 2 wins; on criterion 3 it must fall from 80 to **70** – one third of the range 30 – and that is the most sensitive value of the study. The 70 can be checked by hand: at 70 the standardized values of criterion 3 are 1, 0.5 and 0, alternative 1 scores 0.2 + 0.4 = 0.6 and alternative 2 scores 0.4 + 0.2 = 0.6 – the two weighted sums meet.

## 14.8 Version 2.1: the settings of the method

Step 9, part **Method parameters**, gives one numeric setting several values and re-runs the method for each: the standardization code for any method, the strategy weight v of VIKOR, the power of Compromise Programming, the thresholds α and β of ELECTRE III. For every value it reports the winner, the whole order (ties joined by =) and Spearman's rho of the positions against the base run.

Two readings on the example. VIKOR (chapter 9) prefers alternative 2 for v = 0, 0.25, 0.5 and 0.75 and switches to alternative 1 at v = 1, group utility alone – the compromise depends on the strategy weight, as chapter 9 warned. WAM with the standardization codes 0, 1, 2, 3 gives the orders 2 > 3 > 1, 1 > 2 > 3, 1 > 3 > 2 and 1 > 2 > 3: without

standardization the raw units decide (criterion 1 is in thousands), with any standardization alternative 1 wins – the lesson of section 2.2, now as a table.

## 14.9 Version 2.1: Monte Carlo robustness on the data and the weights

Step **10 Robustness** perturbs everything at once. Every value of the decision matrix, and if you wish every weight, is multiplied by a random factor 1 + u, with u uniform in [−p, +p] or triangular on that interval with mode 0 (small errors more likely than large ones); the method is re-run for every sample and the positions are counted. The table is the **rank acceptability** of section 14.3 extended to the data, with the share of samples in which each alternative keeps its base position and its mean rank; the headline gives the **winner stability**.

On the example with WAM, standardization 1, 200 samples and seed 1995: values uncertain by ±10 % (weights fixed) keep alternative 1 first in 86.5 % of the samples and alternative 2 takes the other 13.5 %; weights uncertain by ±10 % (values fixed) keep it first in 99.5 % – the table of section 14.3; both uncertain, triangular: 93.0 %. The decision is firmer against the weights than against the data, which is what section 14.7 found one value at a time: the winner's value on criterion 3 is only 10 units (one third of the range) away from the crossing. Alternative 3 is last in every sample.

- **Choose p from the data**, not from habit: ±10 % suits measured quantities; estimates of costs or of public acceptance may deserve ±25 %.
- **Report the seed** with the table; the same seed reproduces it on any computer.
- **A refused sample is information**: an objective weight scheme that needs positive values will refuse a sample in which a value crossed zero; the headline counts them.

# 8. Expected Value (EV)

## 8.1 The idea

Often you cannot defend a weight of 0.4 – but you can defend an order: *criterion 1 and criterion 3 matter most, criterion 2 matters least*. Expected Value takes that order seriously and nothing more. Among all the weight vectors that respect the order, it takes the **average** one, and uses it in a weighted sum. The same treatment is applied to the data: only the ranking of the alternatives under each criterion is used, replaced by its average value.

The method is honest about what you know. If you only know an order, you should not pretend to know numbers – but you should not throw the order away either.

## 8.2 What it needs

- An **order of importance** of the criteria (ties allowed). MCEWR reads it from the declared weights: they are used as a ranking, not as numbers.
- For the data, only the order of the alternatives under each criterion – so it works with ranks, classes or words as well as with measurements.
- Nothing else: there are no thresholds and no parameters.

## 8.3 The steps

**Step 1.** Rank the criteria by importance, 1 = most important. Criteria with equal weight share the average rank (two criteria tied at the top get rank 1.5 each).

**Step 2.** Turn the ranks into weights with the **rank-order-centroid** formula: the criterion in position i out of n receives (1/i + 1/(i+1) + … + 1/n) / n. Tied criteria receive the average of the weights of the positions they share. These weights add to 1.

**Step 3.** Rank the alternatives under each criterion, 1 = best, ties sharing the average rank.

**Step 4.** Turn those ranks into scores: the alternative with rank k out of m receives **(m + 1 − k) / (m + 1)**.

**Step 5.** Multiply the scores by the criterion weights and add, exactly as in WAM. The largest total wins.

## 8.4 Solved example

Step 1 and 2. The weights are 0.4, 0.2, 0.4, so criteria 1 and 3 are tied for first place (rank 1.5 each) and criterion 2 is third. For three criteria the rank-order-centroid weights of positions 1, 2 and 3 are

```
position 1 :  (1/1 + 1/2 + 1/3) / 3 = 0.611111
position 2 :  (1/2 + 1/3) / 3       = 0.277778
position 3 :  (1/3) / 3              = 0.111111
```

Criteria 1 and 3 share positions 1 and 2, so each receives (0.611111 + 0.277778) / 2 = **0.444444**; criterion 2 takes position 3 and receives **0.111111**. Notice what has happened: the declared weights 0.4 / 0.2 / 0.4 have become 0.444 / 0.111 / 0.444. Knowing only the order, the two leading criteria are pulled further apart from the third than the original numbers said.

Steps 3 and 4, the scores. With three alternatives, rank 1 gives (3 + 1 − 1)/4 = 0.75, rank 2 gives 0.5 and rank 3 gives 0.25:

| | Criterion 1 rank → score | Criterion 2 rank → score | Criterion 3 rank → score |
|---|---|---|---|
| Alternative 1 | 3 → 0.25 | 1 → 0.75 | 1 → 0.75 |
| Alternative 2 | 1 → 0.75 | 3 → 0.25 | 2 → 0.5 |
| Alternative 3 | 2 → 0.5 | 2 → 0.5 | 3 → 0.25 |

Step 5, the weighted sums:

| | Calculation | Ranking value |
|---|---|---|
| Alternative 1 | 0.444444×0.25 + 0.111111×0.75 + 0.444444×0.75 | **0.527778** |
| Alternative 2 | 0.444444×0.75 + 0.111111×0.25 + 0.444444×0.5 | **0.583333** |
| Alternative 3 | 0.444444×0.5 + 0.111111×0.5 + 0.444444×0.25 | **0.388889** |

**The answer** (the largest value is best): alternative **2** first, alternative 1 second, alternative 3 last. Again the winner differs from WAM, and for an instructive reason: EV never sees *how much* alternative 2 wins criterion 1 by, nor how much alternative 1 wins criteria 2 and 3 by. It sees only who is first, second and third. Alternative 2 is first on the most important criterion and second on another; alternative 1 is first twice but last on a leading criterion.

## 8.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **Expected Value (EV) (provisional)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** *Method options*: leave `ev_weights` and `ev_scores` on **ordinal** – that is the method proper.

**Step 6.** Set both to **cardinal** and run again: the result becomes exactly the WAM result of chapter 3. That single experiment shows what the ordinal treatment costs and what it protects you from.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 8.6 When to use it, and the usual mistakes

- **Use it** when the weights are an opinion about order rather than a measurement, and when the data themselves are ranks or classes – a common situation in environmental and social criteria.
- **Mistake 1 – giving carefully estimated weights and then using EV**, which throws the numbers away and keeps only their order. If your weights are real, use WAM.
- **Mistake 2 – forgetting the effect of ties.** Two criteria with equal declared weights are treated as genuinely tied, and the centroid weights change accordingly, as above.
- **Mistake 3 – quoting the ranking value as a score.** It is an expected value over a set of weight vectors, not a performance.

# 9. Regime Analysis (RA)

## 9.1 The idea

Regime Analysis asks a question no other method here asks: **given that I only know the order of importance of the criteria, how confident can I be that a is better than b?** For each pair it looks at every weight vector that respects the stated order, and measures the share of them for which a comes out ahead. That share is read as a probability. Averaging the probabilities over the opponents gives each alternative its ranking value.

The result is not *a beats b*, but *a beats b in 100 % of the admissible weightings*, or *in 62 % of them* – which is a far more useful sentence in a report than a rank.

## 9.2 What it needs

- An order of importance of the criteria (as EV; MCEWR reads it from the declared weights).
- Only the order of the alternatives under each criterion – ordinal data are welcome.
- A number of sampled weight vectors and a random seed, when the probabilities are estimated by sampling. MCEWR records the seed (`ra_seed`, 1995 by default) so that any published number can be reproduced exactly.

## 9.3 The steps

**Step 1.** Rank the criteria by importance (ties share the average rank), exactly as in EV.

**Step 2.** For every ordered pair (a, b), split the criteria into those on which a is better and those on which b is better.

**Step 3.** a is preferred to b, for a given weight vector, when the total weight on a's side is larger than the total on b's side.

**Step 4.** Compute the share of the admissible weight vectors for which that holds. That is **P(a is preferred to b)**.

**Step 5.** The ranking value of a is the average of P(a preferred to b) over all the other alternatives b. The largest value wins.

## 9.4 Solved example

The criteria ranking is the same as in chapter 8: criteria 1 and 3 tied at the top, criterion 2 last. So every admissible weight vector satisfies w1 = w3, both at least as large as w2, and all three positive.

| Pair | Criteria for the first | Criteria for the second | The first wins when | Probability |
|---|---|---|---|---|
| 1 against 2 | 2 and 3 (w2 + w3) | 1 (w1) | w2 + w3 > w1 — always true, since w3 = w1 and w2 > 0 | **1** |
| 1 against 3 | 2 and 3 (w2 + w3) | 1 (w1) | the same condition | **1** |
| 2 against 3 | 1 and 3 (w1 + w3) | 2 (w2) | w1 + w3 > w2 — always true | **1** |

Every pair is decided with certainty on this example, so no sampling is needed to see the answer. Averaging over the opponents:

| | Probabilities against the others | Ranking value |
|---|---|---|
| Alternative 1 | 1 against alternative 2, 1 against alternative 3 | (1 + 1) / 2 = **1** |
| Alternative 2 | 0 against alternative 1, 1 against alternative 3 | (0 + 1) / 2 = **0.5** |

| | Probabilities against the others | Ranking value |
|---|---|---|
| Alternative 3 | 0 against alternative 1, 0 against alternative 2 | (0 + 0) / 2 = **0** |

**The answer** (the largest value is best): alternative 1, alternative 2, alternative 3 – the same order as WAM, but with a much stronger statement behind it. Alternative 1 is ahead of both others **for every weighting that respects the stated order of importance**. No choice of numbers within that order can overturn it. That is a robustness result, obtained without a sensitivity analysis.

> **Note.** Compare with chapter 8. EV and RA use exactly the same ordinal information and disagree about the winner. EV averages the weights first and then computes one weighted sum; RA keeps every admissible weighting and counts the outcomes. When they disagree, it is a sign that the decision is being carried by the *sizes* of the weights within the order – which is precisely what you said you did not know.

## 9.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **Regime Analysis (RA) (provisional)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** *Method options*: leave `ra_weights` on **ordinal**, `ra_samples` at 10000 and `ra_seed` at 1995.

**Step 6.** The seed is printed in the detail file `regime.txt` (the *Output files* buttons). Quote it in any publication: with the same seed the probabilities are identical on any computer.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 9.6 When to use it, and the usual mistakes

- **Use it** when the weights are genuinely uncertain, when the data are ordinal, or when you want to report confidence rather than a bare order – for example *site A is first under 96 % of the admissible weightings*.
- **Mistake 1 – reporting a probability without the seed and the number of draws.** Then the number cannot be reproduced. MCEWR records both; copy them into your report.
- **Mistake 2 – reading the probability as a chance that the site is good.** It is the share of admissible weightings, nothing more.
- **Mistake 3 – using RA when you have solid numeric weights.** You would be discarding information you actually have.

# 10. TOPSIS

## 10.1 The idea

TOPSIS – *Technique for Order Preference by Similarity to the Ideal Solution* – measures two distances instead of one. The best alternative should be **near the ideal** (the best value of every criterion) and at the same time **far from the anti-ideal** (the worst value of every criterion). The score is the balance between the two, between 0 and 1, and the largest wins.

Why two distances? An alternative can be close to the ideal and also close to the worst point, if it sits in the middle of everything. TOPSIS prefers alternatives that are near the good corner *and* clearly away from the bad one.

## 10.2 What it needs

- Measured values and weights.
- No thresholds, no exponent – there is nothing to tune, which is one reason for its popularity.

## 10.3 The steps

**Step 1.** Standardize the decision table.

**Step 2.** Normalize each criterion by the length of its column: divide every value by the square root of the sum of the squares of that column.

**Step 3.** Multiply by the weights. Call the result the weighted table.

**Step 4.** Read off the **ideal** (the best entry of each column of the weighted table) and the **anti-ideal** (the worst entry).

**Step 5.** For each alternative compute **D+**, the straight-line distance to the ideal, and **D−**, the distance to the anti-ideal: square the differences, add them, take the square root.

**Step 6.** The score is the **closeness C = D− / (D+ + D−)**. Sort from the largest to the smallest.

## 10.4 Solved example

Step 2, the column lengths of the standardized table: criterion 1 gives the square root of $(0^2 + 1^2 + 0.6^2) = 1.166190$; criteria 2 and 3 both give the square root of $1.111111 = 1.054093$. Step 3 multiplies by 0.4, 0.2 and 0.4, giving the weighted table:

| | Criterion 1 | Criterion 2 | Criterion 3 |
|---|---|---|---|
| Alternative 1 | 0 | 0.189737 | 0.379473 |
| Alternative 2 | 0.342997 | 0 | 0.126491 |
| Alternative 3 | 0.205798 | 0.063246 | 0 |
| **Ideal** | **0.342997** | **0.189737** | **0.379473** |
| **Anti-ideal** | **0** | **0** | **0** |

Step 5, the two distances. For alternative 1 the distance to the ideal is the square root of $(0 - 0.342997)^2 + 0 + 0 = 0.342997$, and the distance to the anti-ideal is the square root of $0.189737^2 + 0.379473^2 = 0.424264$.

| | D+ (to the ideal) | D− (from the anti-ideal) | C = D− / (D+ + D−) |
|---|---|---|---|
| Alternative 1 | 0.342997 | 0.424264 | **0.552959** |
| Alternative 2 | 0.316228 | 0.365578 | **0.536191** |
| Alternative 3 | 0.422875 | 0.215297 | **0.337365** |

**The answer** (the largest value is best): alternative 1 (0.552959), alternative 2 (0.536191), alternative 3 (0.337365). Alternative 2 is actually the **closest** to the ideal – its D+ is the smallest of the three – but it is also closer to the anti-ideal, and the balance goes to alternative 1. That single comparison is the whole idea of TOPSIS in one line.

## 10.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **TOPSIS (provisional)**.

**Step 4.** *Standardization*: **1 (v – min) / (max – min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 10.6 When to use it, and the usual mistakes

- **Use it** when you want a distance method with nothing to tune and a score that is easy to explain (0 = as bad as possible, 1 = as good as possible).
- **Mistake 1 – forgetting that the ideal is computed from the alternatives present.** Add or remove an alternative and both reference points move; TOPSIS is the method most often caught by the rank-reversal test of chapter 14. If that matters, use Compromise Programming with fixed limits.
- **Mistake 2 – standardizing twice.** TOPSIS normalizes internally; standardization 0 gives the textbook method on the raw data, standardization 1 gives TOPSIS on standardized data. Both are legitimate – say which one you used.
- **Mistake 3 – reading C as a probability.** It is a ratio of distances.

# 11. VIKOR

## 11.1 The idea

VIKOR looks for a **compromise**: an alternative that is good for the group as a whole and, at the same time, not badly unfair to any single criterion. It computes two numbers for each alternative – the total shortfall S (the group utility) and the worst single shortfall R (the individual regret) – and blends them into an index Q. The smallest Q wins.

The blend is controlled by v. With v = 1 only the total counts (that is Compromise Programming with s = 1). With v = 0 only the worst criterion counts. The usual v = 0.5 weighs the two equally. VIKOR also **tests whether the winner deserves to be called a winner** – the two conditions below – and if not, it reports a set of compromise solutions instead of a single one. No other method here does that.

## 11.2 What it needs

- Measured values and weights.
- The strategy weight v, between 0 and 1 (`vikor_v`, default 0.5).

## 11.3 The steps

**Step 1.** Standardize, then for each criterion note the best value f* and the worst value f−.

**Step 2.** For each alternative and criterion compute the weighted relative shortfall **w × (f* − value) / (f* − f−)**.

**Step 3.** **S** = the sum of those shortfalls (the group utility). **R** = the largest single one (the individual regret).

**Step 4.** Let S* and S− be the smallest and largest S, and R* and R− the smallest and largest R. Then **Q = v × (S − S*) / (S− − S*) + (1 − v) × (R − R*) / (R− − R*)**.

**Step 5.** Sort by Q, smallest first, then test the two conditions of step 6.

**Step 6.** **C1, acceptable advantage**: the gap in Q between the first and the second must be at least 1/(m − 1). **C2, acceptable stability**: the first by Q must also be first by S or by R. If either fails, VIKOR reports a set of compromise solutions rather than a single winner.

## 11.4 Solved example

Step 2 and 3. The shortfalls are the same ones computed for Compromise Programming in section 4.4, so S is identical to the CP value with s = 1; R is the largest term of each row:

| | Weighted shortfalls (criteria 1, 2, 3) | S (sum) | R (largest) |
|---|---|---|---|
| Alternative 1 | 0.4, 0, 0 | **0.4** | **0.4** |
| Alternative 2 | 0, 0.2, 0.266667 | **0.466667** | **0.266667** |
| Alternative 3 | 0.24, 0.133333, 0.4 | **0.693333** | **0.4** |

Step 4, with S* = 0.4, S− = 0.693333, R* = 0.266667, R− = 0.4 and v = 0.5:

```
Q(1) = 0.5 × (0.4 − 0.4)/0.293333 + 0.5 × (0.4 − 0.266667)/0.133333 = 0 + 0.5 = 0.5
Q(2) = 0.5 × (0.466667 − 0.4)/0.293333 + 0.5 × (0.266667 − 0.266667)/0.133333 = 0.113636 + 0 =
0.113636
Q(3) = 0.5 × 1 + 0.5 × 1 = 1
```

**The answer** (the smallest value is best): alternative **2** first (0.113636), alternative 1 second (0.5), alternative 3 last (1). Alternative 1 has the best total S, but the worst regret R – it is the worst of all on criterion 1 – and VIKOR will not overlook that.

Step 6, the conditions. With three alternatives the required gap is 1/(3 − 1) = 0.5, and the actual gap is 0.5 − 0.113636 = 0.386364, which is **smaller**. So condition C1 fails: alternative 2's lead is not decisive. Condition C2 holds (alternative 2 is also first on R). VIKOR therefore reports a **set of compromise solutions, {2, 1}** – the two are recommended together – and the website prints exactly that as a warning under the result. Reporting *alternative 2 wins* without the warning would misrepresent the method.

## 11.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **VIKOR (provisional)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** *Method options*: `vikor_v` = **0.5**. Read the warnings under the summary – the C1 and C2 conditions are reported there.

**Step 6.** Run again with `vikor_v` = **1** (only the total counts) and **0** (only the worst criterion counts) and watch the winner move. That is the clearest demonstration in this whole tutorial of what a parameter really does.

**Step 7.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 11.6 When to use it, and the usual mistakes

- **Use it** when a decision must be defensible to several parties: the R term protects the criterion that would otherwise be sacrificed, and the compromise set is an honest answer when the evidence does not single out a winner.
- **Mistake 1 – ignoring the C1 and C2 warnings** and quoting a single winner. The set is the result.
- **Mistake 2 – comparing Q values between runs.** Q is rescaled to 0–1 within each run by construction; only the order and the gaps within one run mean anything.
- **Mistake 3 – leaving v at 0.5 without thinking.** It is a statement about your problem, not a default.

# 12. ELECTRE III

## 12.1 The idea

ELECTRE III is ELECTRE II with a memory of how imprecise measurements are. Instead of deciding that a is better than b as soon as one number exceeds the other, each criterion carries two thresholds: an **indifference threshold q**, below which a difference is treated as noise, and a **preference threshold p**, above which the preference is complete. Between q and p the preference grows gradually – it is *fuzzy*.

Each ordered pair then receives a **credibility** between 0 and 1: how believable is the statement *a is at least as good as b*. A **veto threshold** can be added per criterion: a difference larger than the veto destroys the credibility whatever the other criteria say. The ranking is obtained by two *distillations* – one that repeatedly picks the best alternatives, one that repeatedly picks the worst – and by averaging the two positions.

## 12.2 What it needs

- Measured values and weights.
- Per criterion: q (indifference), p (preference, at least q) and optionally a veto. In step 3 (Criteria, with *Show thresholds* on) these are the columns q, p and veto.
- Two coefficients of the discrimination threshold used by the distillations, alpha and beta (`electre3_alpha` = 0.30 and `electre3_beta` = −0.15 by default).

> **Important.** The veto threshold is stored in the project file (`.mcewr`), the CSV and the Excel workbook, but **not** in the 1995 `.MCE` format, which had no such column. Save a study that uses vetoes as `.mcewr`.

## 12.3 The steps

**Step 1.** Standardize, and scale the thresholds q, p and the veto with the data.

**Step 2.** For every ordered pair and criterion, compute the degree of concordance: 1 if the difference in b's favour is below q, 0 if it exceeds p, and a straight-line interpolation in between.

**Step 3.** Multiply by the weights and add: this is the **comprehensive concordance C(a,b)**.

**Step 4.** Apply the vetoes: on a criterion where b beats a by more than the veto, the credibility is reduced (to zero in the extreme). Without vetoes the **credibility equals the concordance**.

**Step 5.** **Descending distillation**: take the highest credibility level, keep the alternatives that outrank the others at that level, and repeat inside each group – this gives an order from the best.

**Step 6.** **Ascending distillation**: the same from the worst upwards.

**Step 7.** The ranking value is the **average of the two positions**; the smallest is best. Where the two distillations disagree about a pair, the alternatives are reported as **incomparable**.

## 12.4 Solved example

Our example file has q = 0 and p = 0 on every criterion and no vetoes. That is the **degenerate case**, and it is worth doing first because it shows what ELECTRE III adds and what it does not. With q = p = 0 any difference, however small, counts fully, so the concordance is the plain share of the weight – the same matrix as ELECTRE II in section 6.4 – and with no vetoes the credibility is that matrix unchanged:

| Credibility(row, column) | Alternative 1 | Alternative 2 | Alternative 3 |
|---|---|---|---|
| Alternative 1 | — | 0.6 | 0.6 |
| Alternative 2 | 0.4 | — | 0.8 |
| Alternative 3 | 0.4 | 0.2 | — |

The distillations then give: **descending** – alternative 2 alone at the top, then alternatives 1 and 3 together; **ascending** – alternative 3 alone at the bottom, then alternatives 1 and 2 together. Averaging the two positions:

| | Descending position | Ascending position | Ranking value |
|---|---|---|---|
| Alternative 1 | 2 | 1 | **1.5** |
| Alternative 2 | 1 | 1 | **1** |
| Alternative 3 | 2 | 2 | **2** |

**The answer**: alternative 2 first, alternative 1 second, alternative 3 last – identical to ELECTRE II, as it must be when the thresholds are switched off. This is the standard check that an ELECTRE III implementation is sound, and MCEWR runs it as a test at every release.

To see what ELECTRE III actually adds, give the criteria real thresholds. On the 0–1 standardized scale, q = 0.1 and p = 0.4 would mean *differences below a tenth of the range are noise, differences above four tenths are decisive*. Edit the q and p columns in step 3 and run again: differences that are small relative to the measurement error stop creating preferences, and pairs may become incomparable instead of being forced into an order.

## 12.5 Do it on the website

**Step 1.** Open **https://mcewr.m-rami.org** and click **Evaluate**.

**Step 2.** Step **1 Project** → in the *Open an example* list choose `NIJKAMP.MCE`.

**Step 3.** Step **6 Method and settings** → *Evaluation method*: **ELECTRE III (provisional)**.

**Step 4.** *Standardization*: **1 (v − min) / (max − min)**. *Weights*: **as given in the file**, box *use the weights as given* unticked.

**Step 5.** Then edit the columns **q**, **p** and **veto** of the criteria table in step 3 (click a yellow cell, type, press Enter or Tab) and run again to see the thresholds at work.

**Step 6.** Press **Run the evaluation** and compare the Overall ranking with the table above. Press **Report → PDF** to keep the full working.

## 12.6 When to use it, and the usual mistakes

- **Use it** when the measurements are not precise enough to justify strict comparisons – which is the normal situation in water-resources and environmental studies – and when a very bad performance on one criterion must be able to block an alternative.
- **Mistake 1 – leaving q and p at zero**, which turns ELECTRE III back into ELECTRE II, as this example shows.
- **Mistake 2 – thresholds in the wrong units.** If you standardize, q and p live on the 0–1 scale. Set them from what you know about the measurement error of each criterion.
- **Mistake 3 – saving a study with vetoes as `.MCE`**, which silently loses them. Use `.mcewr`.

# 15. The ten answers side by side

Every chapter used the same three alternatives, the same three criteria, the same weights and the same standardization. Only the method changed. Here is what the ten methods concluded:

| Method | 1st | 2nd | 3rd | Winner |
|---|---|---|---|---|
| Weighted Average Method | Alternative 1 | Alternative 2 | Alternative 3 | 1 |
| Compromise Programming, s = 1 | Alternative 1 | Alternative 2 | Alternative 3 | 1 |
| Compromise Programming, s = 2 | Alternative 2 | Alternative 1 | Alternative 3 | 2 |
| PROMETHEE II | Alternatives 1 and 2 tied | — | Alternative 3 | tie |
| ELECTRE II | Alternative 2 | Alternative 1 | Alternative 3 | 2 |
| AHP (with measured values) | Alternative 1 | Alternative 2 | Alternative 3 | 1 |
| Expected Value | Alternative 2 | Alternative 1 | Alternative 3 | 2 |
| Regime Analysis | Alternative 1 | Alternative 2 | Alternative 3 | 1 |
| TOPSIS | Alternative 1 | Alternative 2 | Alternative 3 | 1 |
| VIKOR | Alternative 2 (with 1) | Alternative 1 | Alternative 3 | 2 |
| ELECTRE III | Alternative 2 | Alternative 1 | Alternative 3 | 2 |

## 15.1 What this table is telling you

- **Alternative 3 is last under every method.** That is a robust finding, and it is the kind of statement a multi-criteria study can make with confidence. Nothing in the data supports alternative 3.
- **The winner depends on the method.** Five methods choose alternative 1, five choose alternative 2, one ties them. This is not a defect of the model or a sign that one method is wrong: the two alternatives are genuinely close, and the methods differ in what they reward.
- **The split is not random.** The methods that allow compensation and look at totals – WAM, CP with s = 1, AHP, TOPSIS, RA – put alternative 1 first, because it collects the highest total. The methods that punish a serious weakness – CP with s = 2, ELECTRE II and III, VIKOR – put alternative 2 first, because alternative 1 is the worst of all on criterion 1. EV joins them because, seeing only ranks, it cannot see that alternative 1's two victories are large.
- **So the real question is not which method is right, but which question is right for your problem.** Is a poor value on criterion 1 acceptable if it buys good values elsewhere? If yes, alternative 1. If no, alternative 2. The methods have turned a vague argument into a precise one.

## 15.2 What to do when methods disagree

**Step 1.** **Run them all** – the *4. Compare* tab does it in one click and shows the ranks side by side with the agreement between the methods.

**Step 2.** **Look for what is stable.** Alternative 3 last, everywhere: report that as settled. The first place is contested: report it as contested, with the reason.

**Step 3.** **Ask the sensitivity question** (chapter 14): the winner changes as soon as criterion 1 is given 43.75 % of the weight instead of 40 % – a margin of 9.4 %. That narrow margin *is* the disagreement between the methods, seen from the other side.

**Step 4.** **Ask the rank-reversal question** (chapter 14): does the order survive adding or removing an alternative? Compromise Programming with fixed limits cannot reverse; TOPSIS reverses most often.

**Step 5.** **Ask what the data alone would say** (chapter 13): with CRITIC weights instead of the declared ones, WAM itself puts alternative 2 first. The expert weights, not the method, are carrying the result.

**Step 6.** **Choose the method before seeing the results**, and say why in your report. Choosing afterwards, from the answer you liked, is not analysis.

# 16. A checklist for your own study

**Step 1.** **Write down the alternatives and the criteria** before any calculation. Every criterion must be measurable for every alternative, and must matter to the decision.

**Step 2.** **Decide the direction of each criterion** – larger is better, or smaller is better – and record the units.

**Step 3.** **Collect the values.** Missing values are refused by the model, not treated as zero; find the number or drop the criterion.

**Step 4.** **Set the weights** and be able to defend them. If you can only defend an order of importance, plan to use EV and RA (chapters 8 and 9).

**Step 5.** **Choose the standardization.** Different units almost always mean standardization 1.

**Step 6.** **Choose the methods before running them**: one compensatory method (WAM or AHP), one distance method (CP or TOPSIS), one outranking method (PROMETHEE II, ELECTRE II or III), and add EV and RA when the inputs are ordinal.

**Step 7.** **Run, and read the warnings.** They are part of the result.

**Step 8.** **Compare the methods** (step 8), then **test the weights, the values and the settings** (step 9), **the robustness** (step 10) and **the set of alternatives** (step 11).

**Step 9.** **Download the report** of every run you intend to quote. It records the settings, the engine version and every intermediate table, so that the result can be reproduced years later.

**Step 10.** **Report honestly**: the method and its settings, what is stable across methods, what is not, and where the decision would change.

Everything in this tutorial – the tabs, the settings, the reports – is described control by control in the **Web User Guide**, step by step in the **Web Tutorial**, and mathematically in the **Method Reference**, all on the *Downloads & documents* page of the website. Questions, corrections and worked examples are welcome at rami@m-rami.org.