Package: memga (via r-universe)

November 10, 2026

Type Package

Title Multiple Choice Method for Selecting Sets of Alternatives

Version 0.1.0

Description Selects a set of alternatives that complement each other, instead of ranking the alternatives one by one. Implements the Multiple Choice Method ('MEM') proposed in da Hora and Costa (2015) <doi:10.1590/0103-6513.084812>, which scores a set by the share of criteria combinations in which the set reaches a cutoff, and an adaptation that scores criteria in pairs using the weaker criterion of each pair. The best set is found either by exhaustive search, which ranks every combination, or by a genetic algorithm for problems too large to enumerate, with an optional number of vacancies per group. Includes radar charts to compare sets and two simulated recruiting data sets.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports graphics, grDevices, stats, utils

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

LazyData true

URL https://github.com/safsalles/portfolio-ds/tree/main/mem-ga/r

BugReports https://github.com/safsalles/portfolio-ds/issues

Config/roxygen2/version 8.1.1

Repository https://safsalles.r-universe.dev

Date/Publication 2026-11-10 14:41:27 UTC

RemoteUrl https://github.com/safsalles/portfolio-ds

RemoteRef HEAD

RemoteSha 85b22991a6b9c0c2c431c0b49fa27c99073a890b

RemoteSubdir mem-ga/r

Contents

ga_control 2
mem_problem 3
mem_score 4
memga 5
memga_radar 7
read_mem_problem 8
recruiting_large 8
recruiting_small 9
traditional_choice 10

Index **11**

`ga_control` *Settings of the genetic algorithm*

Description

Controls the search used by ‘memga(method = "ga")‘. The defaults are the ones used in the original implementation.

Usage

```
ga_control(
  iterations = 15000,
  population = 60,
  stagnation = 14,
  mutation_attempts = 7,
  kick_size = 3,
  climb_every = 5
)
```

Arguments

`iterations`	Number of iterations.
`population`	Number of sets kept in the population.
`stagnation`	Iterations without improvement before a mutation.
`mutation_attempts`	Attempts in each mutation.
`kick_size`	Number of alternatives swapped at once in the strong mutation.
`climb_every`	Number of mutations between one strong mutation and the next.

Value

A list of class ‘memga_ga_control‘.

Examples

```
ga_control(iterations = 2000)
```

`mem_problem` *Define a selection problem*

Description

Collects the data the Multiple Choice Method needs: how each alternative performs on each criterion, which direction is better for each criterion, how important each criterion is, and how much the criteria complement each other.

Usage

```
mem_problem(
  values,
  direction = "max",
  weights = 1,
  groups = NULL,
  complementarity = NULL
)
```

Arguments

`values`	A numeric matrix or data frame with one row per alternative and one column per criterion. Row names are used as the names of the alternatives and column names as the names of the criteria.
`direction`	Either ‘"max"‘ (more is better) or ‘"min"‘ (less is better), one per criterion or a single value for all. Numeric ‘1‘ and ‘0‘ are accepted as ‘"max"‘ and ‘"min"‘.
`weights`	A non-negative importance weight for each criterion, or a single value for all.
`groups`	Optional. The group of each alternative, for example a job role or a position. Criteria are then valued differently for each group, and ‘quotas‘ can be used in [memga()].
`complementarity`	Optional. A symmetric criteria-by-criteria matrix in which a higher value means that the two criteria complement each other more. When omitted, it is computed from the data as one minus the absolute correlation between criteria (the conflict measure of the CRITIC method, Diakoulaki et al., 1995), within each group when ‘groups‘ is given.

Value

An object of class ‘mem_problem‘.

References

Diakoulaki, D., Mavrotas, G. and Papayannakis, L. (1995). Determining objective weights in multiple criteria problems: the CRITIC method. *Computers & Operations Research*, 22(7), 763-770.

Examples

```
values <- data.frame(
  Planning = c(9.5, 9.3, 6.4, 7.6),
  Quality  = c(9.0, 8.8, 7.0, 7.8),
  Costs    = c(6.2, 6.0, 9.4, 7.4),
  Absences = c(2, 3, 4, 5),
  row.names = c("A1", "A2", "A3", "A4")
)
problem <- mem_problem(values, direction = c("max", "max", "max", "min"))
problem
```

`mem_score` *Score a set of alternatives*

Description

Scores a set that you chose yourself, with the same rule used by [memga()].

Usage

```
mem_score(problem, team, rule = c("mean", "minimum"), tau = 0.24)
```

Arguments

`problem`	A [mem_problem()].
`team`	The names of the alternatives in the set.
`rule`	‘"mean"‘ or ‘"minimum"‘.
`tau`	The cutoff, a number greater than 0.

Value

The score of the set, between 0 and 1.

Examples

```
mem_score(recruiting_small, c("A1", "A2"))
mem_score(recruiting_small, c("A1", "A3"))
```

`memga` *Select a set of alternatives with the Multiple Choice Method*

Description

Finds the set of alternatives that best complements itself. A set is scored by the share of criteria combinations in which it reaches the cutoff ‘tau‘, so a set whose members are strong in the same criteria scores below a set whose members cover each other’s weak points.

Usage

```
memga(
  problem,
  size = NULL,
  method = c("exhaustive", "ga"),
  rule = NULL,
  tau = 0.24,
  quotas = NULL,
  control = ga_control(),
  seed = NULL,
  progress = FALSE
)
```

Arguments

`problem`	A [mem_problem()]. A matrix or data frame of values is also accepted and converted with the default settings.
`size`	The number of alternatives to select. It can be omitted when ‘quotas‘ is given.
`method`	‘"exhaustive"‘ (the default) or ‘"ga"‘.
`rule`	‘"mean"‘ or ‘"minimum"‘. By default, ‘"mean"‘ with the exhaustive search and ‘"minimum"‘ with the genetic algorithm.
`tau`	The cutoff, a number greater than 0. A combination counts when its value reaches ‘tau‘. The default, 0.24, is the value the thesis obtains for a performance of 50 choose it for other data.
`quotas`	Optional. A named vector with the number of alternatives to select in each group, for example ‘c(analyst = 2, manager = 1)‘. Needs ‘groups‘ in the problem.
`control`	Settings of the genetic algorithm, from [ga_control()].
`seed`	Optional. A seed to repeat a genetic algorithm result.
`progress`	Show a progress bar during the genetic algorithm.

Details

Two scoring rules are available.

* ‘"mean"‘ is the rule of the thesis that proposed the method. The criteria are combined ‘size‘ at a time, and the value of a set in a combination is the average performance of the set in those criteria, times the average weight, times the average complementarity of the criteria. The number of combinations grows fast, so it is limited to 10,000 per evaluation. * ‘"minimum"‘ is an adaptation for continuous data and larger problems. The criteria are combined two at a time, and the value of a set in a pair is the weaker of the two average performances, times the average weight of the pair, times its complementarity. A strong criterion does not cover a weak one.

Two searches are available.

* ‘"exhaustive"‘ tests every combination and returns the full ranking. Tied sets share a position, because the method can point to more than one good choice. It is limited to 20 million evaluations. * ‘"ga"‘ uses a genetic algorithm with a strong mutation, for problems too large to enumerate. It does not guarantee the best answer, so it is worth repeating with other seeds.

Value

An object of class ‘memga‘, a list with

`team`	The names of the selected alternatives. With the exhaustive search, the first set of the ranking.
`score`	The score of ‘team‘, between 0 and 1.
`ranking`	With the exhaustive search, a data frame with the columns ‘position‘, ‘set‘ and ‘score‘, and a list column ‘members‘.
`history`	With the genetic algorithm, the best score at each iteration.
`traditional`	The traditional choice (see [traditional_choice()]) and its score under the same rule, for comparison.
`problem`, `size`, `method`, `rule`, `tau`, `quotas`, `seed`	The inputs used.

References

da Hora, H. R. M. and Costa, H. G. (2015). Proposta de um método multicritério para escolha múltipla. *Production*, 25(2). doi:10.1590/01036513.084812

da Hora, H. R. M. (2013). *Método de Escolha Múltipla: uma proposta multicritério para seleção em conjunto de mais de uma alternativa*. Doctoral thesis, Universidade Federal Fluminense.

See Also

[mem_problem()], [traditional_choice()], [memga_radar()]

Examples

```
# Two of eight candidates, testing every pair
result <- memga(recruiting_small, size = 2)
result
```

```
# The same problem with the adaptation that scores criteria in pairs
memga(recruiting_small, size = 2, rule = "minimum")

# A team of 4 among 300 candidates, with vacancies per group, found by
# the genetic algorithm (few iterations to keep the example fast)
memga(recruiting_large, method = "ga", tau = 0.54,
      quotas = c(analyst = 2, engineer = 1, manager = 1),
      control = ga_control(iterations = 300), seed = 1)
```

`memga_radar` *Radar chart of the selected set*

Description

Draws one axis per criterion and compares the average performance of the selected set with the traditional choice and with the average of all alternatives. A set that complements itself has no deep dips: where one member is weak, another is strong.

Usage

```
memga_radar(x, compare = TRUE, ...)

## S3 method for class 'memga'
plot(x, ...)
```

Arguments

`x`	A result from [memga()].
`compare`	Show the traditional choice (‘TRUE‘, the default) or only the selected set (‘FALSE‘).
`...`	Not used.

Details

Values are the normalised performances, from 0 (worst) to 1 (best), so for criteria in which less is better the axis is already inverted. When the problem has groups, they are adjusted by group, as in the score, so a value can pass 1.

Value

‘NULL‘, invisibly. Called for the chart.

Examples

```
result <- memga(recruiting_small, size = 2)
memga_radar(result)
```

read_mem_problem *Read a problem from CSV files*

Description

Reads ‘criteria.csv‘, ‘alternatives.csv‘ and, when it exists, ‘complementarity.csv‘ from a folder.

Usage

```
read_mem_problem(folder)
```

Arguments

folder	The folder with the files.

Details

* ‘criteria.csv‘ has the columns ‘criterion‘, ‘direction‘ (1 when more is better and 0 when less is better) and ‘weight‘. * ‘alternatives.csv‘ has the column ‘alternative‘, one column per criterion and, optionally, a ‘group‘ column. * ‘complementarity.csv‘ has the column ‘criterion‘ and one column per criterion. The diagonal is ignored.

Value

An object of class ‘mem_problem‘.

Examples

```
folder <- system.file("extdata", "recruiting-small", package = "memga")
read_mem_problem(folder)
```

recruiting_large *Large recruiting example*

Description

A simulated problem for building a project team of 4 among 300 candidates: 150 analysts, 90 engineers and 60 managers, graded from 0 to 10 on eight criteria (sql, statistics, visualization, programming, communication, leadership, negotiation and organization). Testing every possible team is not practical, so the search is done with the genetic algorithm. The groups are the job roles, and each role values the criteria differently. The complementarity is computed from the data. The data is fictional.

Usage

```
recruiting_large
```

Format

A [mem_problem()] with 300 alternatives, 8 criteria and 3 groups.

Examples

```
recruiting_large
```

`recruiting_small` *Small recruiting example*

Description

A simulated problem for choosing 2 of 8 candidates for two openings, graded from 0 to 10 on six criteria. Candidates A1 and A2 have the best averages but are alike, strong in planning, logistics and quality and average in costs and people, while A3 is the opposite. It is the kind of problem used to present the method: the two best candidates one by one are not the best pair.

Usage

```
recruiting_small
```

Format

A [mem_problem()] with 8 alternatives and 6 criteria, with weights and a complementarity matrix.

Details

The criteria are Planning, Logistics, Quality, Costs, People and Absences (less is better). The data is fictional.

Examples

```
recruiting_small
memga(recruiting_small, size = 2)
```

`traditional_choice` *The traditional choice*

Description

The usual choice without the Multiple Choice Method: rate each alternative alone, by the weighted average of its normalised performance, and keep the best ones, respecting the vacancies per group when there are quotas. [memga()] reports it next to the selected set for comparison.

Usage

```
traditional_choice(problem, size = NULL, quotas = NULL)
```

Arguments

`problem`	A [mem_problem()].
`size`	The number of alternatives to select. It can be omitted when ‘quotas‘ is given.
`quotas`	Optional. A named vector with the number of alternatives to select in each group.

Value

The names of the selected alternatives.

Examples

```
traditional_choice(recruiting_small, size = 2)
traditional_choice(recruiting_large, quotas = c(analyst = 2, engineer = 1, manager = 1))
```

Index

∗ **datasets**
recruiting_large, 8
recruiting_small, 9

ga_control, 2

mem_problem, 3
mem_score, 4
memga, 5
memga_radar, 7

plot.memga *(*memga_radar*)*, 7

read_mem_problem, 8
recruiting_large, 8
recruiting_small, 9

traditional_choice, 10