Package ‘goalp’

July 22, 2025

Type Package

Title Weighted and Lexicographic Goal Programming Interface

Version 0.3.1

Description Solves goal programming problems of the weighted and lexicographic type, as well as combinations of the two, as described by Ignizio (1983) <doi:10.1016/0305-0548(83)90003-5>. Allows for a simple human-readable input describing the problem as a series of equations. Relies on the 'lpSolve' package to solve the underlying linear optimisation problem.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.0.0)

Imports lpSolve

RoxygenNote 7.2.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author David Palma [aut, cre],
Richard Hodgett [ctb]

Maintainer David Palma <`D.Palma@leeds.ac.uk`>

Repository CRAN

Date/Publication 2022-11-29 10:30:02 UTC

Contents

goalp — *Solves a (linear) goal programming problem*

Description

Given a set of equations representing goals of a linear goal programming problem, it finds the optimal solution.

Usage

```
goalp(
  eqs,
  A = NULL,
  m = NULL,
  b = NULL,
  w = NULL,
  p = NULL,
  varType = NULL,
  normW = FALSE,
  silent = FALSE
)
```

Arguments

`eqs`	Character vector describing a set of linear equations. The vector can either contain a single element with one equation per line, or multiple elements, each with a single equation. Equations must be valid R expressions (see details).
`A`	Numeric matrix with the coefficients of the variables. One row per equation, one column per variable. Columns can be named according to the variables they correspond to. Rows can be named for their corresponding goals. Ignored if argument `eqs` is provided.
`m`	Character vector with the relationship between the left and right-hand side of the goals. It can be any of `=`, `==`, `<=`, `>=`. = allows for positive (excess) and negative (lack) deviations. == do not allow any deviation, enforcing fulfillment of the goal. <= automatically assigns a weight equal to zero to the negative (lack) deviation. >= automatically assigns a weight equal to zero to the positive (excess) deviation.
`b`	Numeric vector with the values on the right hand side of the goals. Ignored if argument `eqs` is provided.

`w` Numeric matrix with the weights associated to the deviations from each goal. It should have as many rows as goals, and two columns: the first column corresponding to the weight of the positive deviation (excess), and the second column corresponding to the weight of the negative deviation (lack). This argument is ignored if `eqs` is provided. If omitted and `eqs` is not provided either, default weights are dependent on the type of goal, as follows.

- =: Positive and negative deviations are assigned equal weights of 1.
- ==: Positive and negative deviations are assigned equal weights of NA, as these deviations will be excluded from the problem, i.e. the goal will be enforced.
- >=: Positive deviation is assigned a weight of 0, so it does not influence the objective function (and therefore the solution to the problem). The negative deviation is assigned a weight of 1, but if manually set to NA, then the inequality is enforced.
- <=: Negative deviation is assigned a weight of 0, so it does not influence the objective function (and therefore the solution to the problem). The positive deviation is assigned a weight of 1, but if manually set to NA, then the inequality is enforced.

`p` Numeric matrix indicating the priority of each deviation under a lexicographic approach. Lower numbers represent higher priority (e.g. from lower to higher priority: 1, 2, 3, ...). It must have as many rows as goals, and two columns. This argument is ignored if `eqs` is provided. If omitted and not provided in `eqs` either, default priorities are dependent on the type of goal, as follows.

- =: Positive and negative deviations are assigned equal priority of 1.
- ==: Positive and negative deviations are assigned equal priority of NA, as these deviations will be excluded from the problem, i.e. the goal will be enforced.
- >=: Positive deviation is assigned a priority of +Inf, making it irrelevant. The negative deviation is assigned a priority of 1.
- <=: Negative deviation is assigned a priority of +Inf, making it irrelevant. The positive deviation is assigned a priority of 1.

`varType` Named character vector. Defines the type of each variable. It can be defined as `c(x1="int", x2="cont")`. Omitted variables are assumed to be integer. Each element can be either `"continuous"` (i.e. non-negative real values), `"integer"` (i.e. non-negative natural values), or `"binary"` (i.e. only take values 0 or 1). Using only the first letters is accepted too. If omitted, all variables are assumed to be integer.

`normW` Logical. TRUE to scale the weights by the inverse of the corresponding right-hand size value of the goal (b). Useful to balance the relevance of all goals. Equivalent to normalising the problem so `b=1` for all goals.

`silent` Logical. TRUE to prevent the function writing anything to the console (or the default output). Default is FALSE.

Details

The actual solution of the linear programming problem is found using lp_solve `https://lpsolve.sourceforge.net/`, through its R interface (the lpSolve package).

Argument 'eqs' defines the goals of the goal programming problem through easy human-readable text. When writing a constranit, all variables must be on the left-hand side, with only numeric values on the right-hand side. Equations must be valid R expressions. Examples of valid equations are the following:

- `"3*x + 2*y = 16"`
- `"4*x - y = 3"`

On the other hand, the following are not valid expressions:

- `"3*x = 16 - 2*y"`
- `"4x + 1y = 5"`

While optional, it is highly encouraged to provide names for each goal. The user can also provide weights and/or lexicographic priorities for the positive (excess) and negative (lack) deviations associated to each goal. The following example shows how to provide this information: `" Labour : 20*A + 12*B + 40*C = 1200 | 0.2 0.1 | 1# 2# Profit : 11*A + 16*B + 8*C = 1000 | 0.1 0.3 | 3# 4# Batteries: 4*A + 3*B + 6*C = 200 | 0.2 0.1 | 5# 6#"` The name of the goal must be followed by a colon (`:`) or split vertical bars (`|`). Then the goal. Then the weights associated to the positive deviation first (excess), and the negative deviation (lack) last, separated by an empty space. Finally, the lexicographic priorities for the positive (excess) and negative (lack) deviations can be provided as numbers, each followed by a hashtag, and separated by an space, in that order. Lower values imply a higher priority, and the same priority can be assigned to multiple deviations. Only the equation is mandatory. If the weights are omitted, all of them are assumed to be equal to one. If the lexicographic priorities are omitted, all of them are assumed to be equal to one.

Value

goalp object. It contains the following elements.

- `A`: The coefficient matrix of the decision variables. It does not include the coefficients of the deviations.
- `m`: The relationship between the left- and right-hand side of the goals.
- `b`: The right-hand side of the goals.
- `w`: The weights of the deviation variables.
- `p`: The lexicographic priorities of deviations variables.
- `A_`: The coefficient matrix of the decision and deviation variables.
- `w_`: The weights of the decision and deviation variables.
- `eqs`: Text version of the goal programming problem.
- `varType`: Vector describing the type of the decision variables (binary, integer, or continuous).
- `x`: Optimal value of the decision variables.
- `d`: Optimal value of the deviation variables.
- `obj`: The value of the objective function (sum of weighted deviations). If using lexicographic priorities, the value for the objective function using all deviations (i.e. ignoring the priority) in each stage.
- `X`: The value of the decision variables for the optimal solution in each stage of the lexicographic problem. If there are no lexicographic priorities, then a single row matrix.

- `lp`: lp object describing the solution of the underlying linear programming problem. See lp.object. When using lexicographic priorities, the solution to the last stage.
- `solutionFound`: Logical taking value TRUE if a solution was found, FALSE otherwise.

`msg` *: msg: Formats and prints a message to screen.*

Description

Message function

Usage

```
msg(...)
```

Arguments

`...`	A series of objects (usually strings and numbers) to concatenate and print to screen.

`new_goalp` *new_goalp: Creates a new goalp object*

Description

Constructor of goalp object

Usage

```
new_goalp(lp, A, m, b, w, p, varType, X, obj, eqs)
```

Arguments

`lp`	lp object. The solution of the underlying linear program.
`A`	Numeric matrix with goals coefficients. Only for original variables. Rows and columns must be named.
`m`	Character vector containg the relation between Ax and b. Each element can be =, ==, >, <. >=, <=.
`b`	Numeric vector with the right hand side of the goals.
`w`	Numeric matrix (nC x 2) with the weights of each deviation.
`p`	Numeric matrix containing the priorities of each deviation variable for lexicographic goal programming. Lower numbers imply higher priority.
`varType`	Character vector describing the type of the original variables, as either "b", "i", or "c".

`X`	Numeric matrix with the value of the (decision) variables in each iteration of the lexicographic optimisation.
`obj`	Numeric vector with the value of the objective function in each iteration of the lexicographic optimisation.
`eqs`	Character vector with the human-readable formulation of the problem. Generated automatically from A, b and w if not provided.

Details

It doesn't do any checks, but it does generate objects

- `x`: Vector with the optimal value of decision variables.
- `d`: Matrix with the optimal value of the deviations.
- `solutionFound`: TRUE if a solution was found, FALSE otherwise.

Value

A goalp object.

`parseGoal` *Parses text describing goal programming problem.*

Description

Given a character vector describing a series of linear equations, it parses them into an `A` numerical matrix describing the variables coefficient in the left hand size, a `b` numerical vector with values on the right hand size, and an `m` character vector indicating the relation between the left and right hand side (`=`, `==`, `<=`, `>=`, `<`, `>`).

Usage

```
parseGoal(eqs)
```

Arguments

`eqs`	Character vector describing a set of linear equations. The vector can either contain a single element with one equation per line, or multiple elements, each with a single equation. Equations must be valid R expressions (see details).

Details

This function can only parse linear equations. All variables must be on the left-hand side, with only numeric values on the right-hand side. Equations must be valid R expressions. Examples of valid equations are the following:

- `"3*x + 2*y = 16"`
- `"4*x - y = 3"`

The following are not valid expressions:

- `"3*x = 16 - 2*y"`
- `"4x + 1y = 5"`

Signs = and == are considered equivalent, and the first will be replaced by the second after parsing.

Optionally, names, weights and lexicographic priorities can be provided for each goal (equation) using the following format: `" Labour : 20*A + 12*B + 40*C = 1200 | 0.2 0.1 | 1# 2# Profit : 11*A + 16*B + 8*C = 1000 | 0.1 0.3 | 3# 4# Batteries: 4*A + 3*B + 6*C = 200 | 0.2 0.1 | 5# 6#"` The name of the goal must be followed by a colon (`:`) or vertical bars (`|`). Then the goal. Then the weights associated to the negative deviation first (lack), and the positive deviation (excess) last, separated by an empty space. Finally, the lexicographic priorities for the negative (lack) and positive (excess) deviations can be provided as numbers, each followed by a hashtag (#), and separated by an space, in that order. Lower values imply a higher priority, and the same priority can be assigned to multiple deviations. Only the equation is mandatory. If the weights are omitted, all of them are assumed to be equal to one for equations with the = sign. If the equation is actually an inequality with >=, then the default positive (excess) deviation weight is zero. If <=, then the default negative (lack) deviation is zero. If the lexicographic priorities are omitted, all of them are assumed to be equal to one for equations, but for inequalities >= the positive (excess) deviation is given a priority of +Inf (i.e. it will never be minimised), and for inequalities <= the negative (lack) deviation is given a default priority of +Inf (i.e. it will never be minimised).

Value

List with five elements.

- `A`: Numeric matrix with the coefficients of the variables. One row per equation, one column per variable. Columns are named according to the variables they represent. Rows are named for each equation, if a name for them was provided.
- `b`: Numeric vector with the values on the right hand side of the equations.
- `m`: Character vector with as many elements as equations. Each element is one of `=`, `==`, `<=`, `>=`, `<`, `>`.
- `w`: Numeric matrix with the weights associated to the deviations of each goal. Each row corresponds to a goal. The first column corresponds to the positive deviation (excess) and the second column to the negative deviation (lack).
- `p`: Numeric matrix with the lexicographic priority associated to each goal. Lower values represent higher priority. Each row corresponds to a goal. The first column corresponds to the positive deviation (excess) and the second column to the negative deviation (lack).

`print.goalp` *: print.goalp: Prints a summary of a goalp object to screen.*

Description

Prints a human-readable formulation of a goal programming problem.

Usage

```
## S3 method for class 'goalp'
print(x, ...)
```

Arguments

`x`	A goalp object.
`...`	Additional arguments. Ignored.

Value

A scalar character (i.e. a text string) with a human-readable formulation of the goal programming problem represented by goalp object `x`. This can be edited and used as an input to goalp, if modifications to the goal programming problem are required.

`solveGP` *Solves a weighted Linear Goal Programming problem*

Description

Does not perform any check. It receives set of matrices and vectors describing the original problem, and expands them adding the corresponding deviations. It omits deviations with weight equal to NA.

Usage

```
solveGP(A, b, w, varType, silent = FALSE)
```

Arguments

`A`	Numeric matrix of coefficients of the goals (left-hand-side).
`b`	Numerical vector. Right hand-side of the goals.
`w`	Numerical matrix of the weights of the constrains. As many rows as goals, and two columns (positive and negative deviations).
`varType`	Character vector. Type of each variable ("i", "c" or "b" for integer, continuous or binary, respectively). Must have as many elements as columns in `A`.
`silent`	Logical. TRUE to prevent the function writing anything to the console (or the default output). Default is FALSE.

Value

An lp object, generated by the lpSolve package, which in turn calls the lp_solve C package.

`summary.goalp` *: summary.goalp: Prints a summary of a goalp object to screen.*

Description

Prints a summary of a goalp object to the console.

Usage

```
## S3 method for class 'goalp'
summary(object, ...)
```

Arguments

`object`	A goalp object.
`...`	Additional arguments. Ignored.

Value

No return value (NULL). Called for its side effect of printing a summary of a goalp object to the standard output (usually the console).

`validateMatrices` *Validates the input of a goal programming problem*

Description

Validates the input of a goal programming problem

Usage

```
validateMatrices(
  A,
  b,
  m,
  w = NULL,
  p = NULL,
  setDefaults = FALSE,
  silent = FALSE
)
```

Arguments

`A`	Numeric matrix with the coefficients of the variables. One row per equation, one column per variable.
`b`	Numeric vector with the values on the right hand side of the goals.
`m`	Character vector with the relationship between the left and right-hand side of the goals. It can be any of `=`, `==`, `<=`, `>=`.
`w`	Numeric matrix with the weights associated to the deviations from each goal. It should have as many rows as goals, and two columns: the first column corresponding to the weight of the positive deviation (excess), and the second column corresponding to the weight of the negative deviation (lack).
`p`	Numeric matrix indicating the priority of each deviation under a lexicographic approach. Lower numbers represent higher priority (e.g. from lower to higher priority: 1, 2, 3, ...). It must have as many rows as goals, and two columns.
`setDefaults`	Scalar logical. If TRUE, A, b, m, w, and p are filled in with default values as required.
`silent`	Logical. TRUE to prevent the function writing anything to the console (or the default output). Default is FALSE.

`validate_goalp` *: validate_goalp: A validator for goalp objects.*

Description

Checks that the internals of a goalp object are consistent.

Usage

```
validate_goalp(gp)
```

Arguments

`gp`	A goalp object.

Value

The unmodified input invisibly.

Index