

Package: ROI.plugin.lpsolve (via r-universe)

October 15, 2026

Version 1.0-2

Title 'lp_solve' Plugin for the 'R' Optimization Infrastructure

Description Enhances the 'R' Optimization Infrastructure ('ROI') package with the 'lp_solve' solver.

Imports stats, methods, utils, checkmate, ROI (>= 0.3-0), lpSolveAPI (>= 5.5.2.0-1)

Suggests slam

License GPL-3

URL <https://roigrp.gitlab.io>,
<https://gitlab.com/roigrp/solver/ROI.plugin.lpsolve>

Repository <https://roigrp.r-universe.dev>

Date/Publication 2026-08-16 15:53:48 UTC

RemoteUrl <https://gitlab.com/roigrp/solver/roi.plugin.lpsolve>

RemoteRef HEAD

RemoteSha ff656f54ba3b7e96606d5cc4b21721f42e159e5d

Contents

Example-1	2
lp_control	2
read.lp	5
write.lp	6
Index	8

Example-1

*Linear Problem 1***Description**

$$\text{maximize } 2x_1 + 4x_2 + 3x_3$$

subject to :

$$3x_1 + 4x_2 + 2x_3 \leq 60$$

$$2x_1 + x_2 + 2x_3 \leq 40$$

$$x_1 + 3x_2 + 2x_3 \leq 80$$

$$x_1, x_2, x_3 \geq 0$$

Examples

```
library(ROI)
mat <- matrix(c(3, 4, 2,
               2, 1, 2,
               1, 3, 2), nrow=3, byrow=TRUE)
x <- OP(objective = c(2, 4, 3),
        constraints = L_constraint(L = mat,
                                   dir = c("<=", "<=", "<="),
                                   rhs = c(60, 40, 80)),
        maximum = TRUE)

opt <- ROI_solve(x, solver = "lpsolve")
opt
## Optimal solution found.
## The objective value is: 7.666667e+01
solution(opt)
## [1] 0.000000 6.666667 16.666667
```

lp_control

*Construct lp_solve Control Parameters***Description**

A helper function to construct the control argument passed to ROI_solve when using the **lpsolve** solver. All arguments are optional and default to NULL, in which case they are dropped from the resulting list and the **lpSolveAPI** / lp_solve default is used.

Usage

```
lp_control(basis = NULL, branch.mode = NULL, branch.weights = NULL,
  verbose = NULL, anti.degen = NULL, basis.crash = NULL,
  bb.depthlimit = NULL, bb.floorfirst = NULL, bb.rule = NULL,
  break.at.first = NULL, break.at.value = NULL, epslevel = NULL,
  epsb = NULL, epsd = NULL, epsel = NULL, epsint = NULL,
  epsperturb = NULL, epspivot = NULL, improve = NULL, infinite = NULL,
  maxpivot = NULL, mip.gap = NULL, negrange = NULL, obj.in.bas = NULL,
  pivoting = NULL, presolve = NULL, scalelimit = NULL, scaling = NULL,
  simplextype = NULL, timeout = NULL)
```

Arguments

- | | |
|----------------|---|
| basis | <p>an optional named list used to set the initial basis of the LP, e.g.,
 <code>R> lp_control(basis = list(basis = c(1, 2, 3),
 + nonbasic = TRUE, default = TRUE))</code>.
 The elements of <code>basis</code> are passed to the lpSolveAPI function set.basis as arguments.</p> <ul style="list-style-type: none"> • <code>basis</code> a numeric vector giving the indices of the basis. • <code>nonbasic</code> an optional logical, if <code>TRUE</code> the nonbasic variables included in the basis as well. • <code>default</code> an optional logical, if <code>TRUE</code> the default basis is used and the arguments <code>basis</code> and <code>nonbasic</code> are ignored. |
| branch.mode | <p>an optional list used to set the branch and bound mode, e.g.,
 <code>R> lp_control(branch.mode = list(columns = c(1, 2, 3),
 + modes = c("ceiling", "auto", "floor")))</code>
 The elements of <code>branch.mode</code> are passed to the lpSolveAPI function set.branch.mode as arguments.</p> <ul style="list-style-type: none"> • <code>columns</code> a vector of integers giving the column indices for which the mode is set. • <code>modes</code> a character vector giving the modes of the columns specified in <code>columns</code>, one of "ceiling", "floor", "auto" or "default". |
| branch.weights | <p>an optional numeric vector giving the weights for the decision variables. The length of <code>branch.weights</code> must be equal to the length of the objective function.</p> |
| verbose | <p>a character string controlling the severity of messages reported by <code>lp_solve</code>: one of "neutral", "critical", "severe", "important", "normal", "detailed" or "full" (for more information see lp.control.options).</p> |
| anti.degen | <p>a character vector containing one or more anti-degeneracy options, e.g., "fixedvars", "stalling" or "infeasible". The default used by <code>lp_solve</code> is <code>c("infeasible", "stalling", "fixedvars")</code> (for more information see lp.control.options).</p> |
| basis.crash | <p>a character string specifying the basis crash mode, one of "none", "mostfeasible" or "leastdegenerate". The default is "none" (for more information see lp.control.options).</p> |
| bb.depthlimit | <p>an integer giving the maximum branch-and-bound depth. The default is -50 (for more information see lp.control.options).</p> |
| bb.floorfirst | <p>a character string specifying which branch to take first, one of "ceiling", "floor" or "auto". The default is "auto" (for more information see lp.control.options).</p> |

bb.rule	a character vector giving the branch-and-bound rule, e.g., "first", "gap", "range", "fraction", "pseudocost", "pseudononint" or "pseudoratio", optionally augmented by additional modes. The default is <code>c("pseudononint", "greedy", "dynamic", "rcostfixing")</code> (for more information see lp.control.options).
break.at.first	a logical controlling whether the branch-and-bound algorithm should be stopped at the first solution or the branch-and-bound algorithm continues until an optimal solution is found. The default is FALSE (for more information see lp.control.options).
break.at.value	a numeric, if given the branch-and-bound algorithm stops when the objective function becomes better than the specified value. The default is to continue until an optimal value is found (for more information see lp.control.options).
epslevel	a character string giving the type of tolerance thresholds, one of "tight", "medium", "loose" or "baggy". The default is "tight" (for more information see lp.control.options).
epsb	a positive numeric giving the tolerance used to determine whether a right-hand-side value should be considered zero. The default is $1.0e-10$ (for more information see lp.control.options).
epsd	a positive numeric giving the tolerance used to determine whether a computed reduced cost should be considered zero. The default is $1.0e-9$ (for more information see lp.control.options).
epsel	a positive numeric giving the tolerance used for rounding values to zero. The default is $1.0e-12$ (for more information see lp.control.options).
epsint	a positive numeric giving the tolerance used to determine whether a floating-point number is an integer. The default is $1.0e-7$ (for more information see lp.control.options).
epsperturb	a positive numeric giving the perturbation scalar for degenerate problems. The default is $1.0e-5$ (for more information see lp.control.options).
epspivot	a positive numeric giving the tolerance used to determine whether a pivot element is zero. The default is $2.0e-7$ (for more information see lp.control.options).
improve	a character vector specifying the iterative improvement level, e.g., "none", "solution", "dualfeas", "thetagap" or "bbsimplex". The default is <code>c("dualfeas", "thetagap")</code> (for more information see lp.control.options).
infinite	a positive numeric specifying the practical value used for infinity. The default is $1.0e30$ (for more information see lp.control.options).
maxpivot	a positive integer giving the maximum number of pivots between re-inversion of the matrix. The default is 250 (for more information see lp.control.options).
mip.gap	a numeric vector of length two giving the absolute and relative MIP gap used in the branch-and-bound algorithm. The default is $1.0e-11$ (for more information see lp.control.options).
negrange	a nonpositive numeric below which variables are split into negative and positive parts. The default is $-1.0e6$ (for more information see lp.control.options).
obj.in.bas	a logical specifying whether the objective function is stored in the constraint matrix. The default is TRUE (for more information see lp.control.options).
pivoting	a character vector specifying the pivot rule and mode, e.g., "firstindex", "dantzig", "devex" or "steepestedge", optionally augmented by additional modes. The default is <code>c("devex", "adaptive")</code> (for more information see lp.control.options).

presolve	a character vector specifying the presolve steps to be carried out before solving, e.g., "rows", "cols", "lindp". The default is "none" (for more information see lp.control.options).
scalelimit	a numeric giving the relative scaling convergence criterion. The default is 5 (for more information see lp.control.options).
scaling	a character vector specifying the scaling algorithm, e.g., "none", "extreme", "range", "mean", "geometric" or "curtisreid", optionally augmented by additional modes. The default is c("geometric", "equilibrate", "integers") (for more information see lp.control.options).
simplextype	a character vector of length one or two giving the simplex type, one of c("primal"), c("dual"), c("primal", "dual") or c("dual", "primal"). The default is c("dual", "primal") (for more information see lp.control.options).
timeout	an integer giving the number of seconds after which a timeout occurs. If zero, no timeout occurs (for more information see lp.control.options).

Value

a named list containing only the non-NULL arguments, suitable for use as the control argument of ROI_solve.

See Also

[control](#), [lp.control](#), [lp.control.options](#).

Examples

```
lp_control(epsel = 1e-10, presolve = c("rows", "cols"), timeout = 30)
```

read.lp

Read Optimization Problem

Description

Read an optimization problem from a file.

Usage

```
read.lp(file, type=c("lp", "mps", "freemps"))
```

Arguments

file	a character giving the name of the file the optimization problem is read from.
type	a character giving the name of the file format the optimization problem is stored in.

write.lp

7

```
        bounds = V_bound(ui = seq_len(3), ub = c(1000, Inf, 1000), nobj = 3),
        maximum = TRUE)
write.lp(x, "optimization_problem.lp")

## End(Not run)
```

Index

`control`, [5](#)

`Example-1`, [2](#)

`lp.control`, [5](#)

`lp.control.options`, [3–5](#)

`lp_control`, [2](#)

`read.lp`, [5](#)

`set.basis`, [3](#)

`set.branch.mode`, [3](#)

`write.lp`, [6](#)