

# Package ‘ROI.plugin.lpsolve’

May 7, 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, 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>

**NeedsCompilation** no

**Author** Florian Schwendinger [aut, cre]

**Maintainer** Florian Schwendinger <FlorianSchwendinger@gmx.at>

**Repository** CRAN

**Date/Publication** 2023-07-07 11:50:05 UTC

## Contents

control . . . . .	2
Example-1 . . . . .	3
read.lp . . . . .	4
write.lp . . . . .	5
<b>Index</b>	<b>6</b>

control

*Control Variables***Description**

The control variables are all optional, but can in some cases be used to improve the solving time and the solutions.

**Arguments**

- |                |   |
|----------------|---|
| basis          | <p>an optional named list used to set the initial basis of the LP, e.g.,<br/> <code>R&gt; control &lt;- list(basis=list(basis = c(1, 2, 3),<br/> + nonbasic = TRUE, default = TRUE))</code>.<br/> The elements of <code>basis</code> are passed to the <b>lpSolveAPI</b> function <a href="#">set.basis</a> as arguments.</p> <ul style="list-style-type: none"> <li>• <code>basis</code> a numeric vector giving the indices of the basis.</li> <li>• <code>nonbasic</code> an optional logical, if TRUE the nonbasic variables included in the basis as well.</li> <li>• <code>default</code> an optional logical, if TRUE the default basis is used and the arguments <code>basis</code> and <code>nonbasic</code> are ignored.</li> </ul> |
| branch.mode    | <p>an optional list used to set the branch and bound mode, e.g.,<br/> <code>R&gt; control &lt;- list(branch.mode=list(columns=c(1, 2, 3),<br/> + modes=c("ceiling", "auto", "floor"))</code> The elements of <code>branch.mode</code> are passed to <b>lpSolveAPI</b> function <a href="#">set.branch.mode</a> as arguments.</p> <ul style="list-style-type: none"> <li>• <code>columns</code> a vector of integer giving the column indices for which the mode is set.</li> <li>• <code>modes</code> a character vector giving the modes of the columns specified in <code>columns</code>.</li> </ul>  |
| branch.weights | <p>an optional numeric vector giving the weights for the decision variables. The length of the <code>branch.weights</code> must be equal to length of the objective function.</p>   |
| verbose        | <p>a character string (for more information see <a href="#">lp.control</a>).</p>  |
| anti.degen     | <p>a character vector (for more information see <a href="#">lp.control</a>).</p>  |
| basis.crash    | <p>a character string (for more information see <a href="#">lp.control</a>).</p>  |
| bb.depthlimit  | <p>a integer giving the maximum branch-and-bound depth (for more information see <a href="#">lp.control</a>).</p>   |
| bb.floorfirst  | <p>a character string (for more information see <a href="#">lp.control</a>).</p>  |
| bb.rule        | <p>a character vector giving the branch-and-bound rule (for more information see <a href="#">lp.control</a>).</p>   |
| break.at.first | <p>a logical controlling whether the branch-and-bound algorithm should be stopped at the first solution or the branch-and-bound algorithm continuous until an optimal solution is found (for more information see <a href="#">lp.control</a>).</p>  |
| break.at.value | <p>a numeric, if given the branch-and-bound algorithm stops when the objective function becomes smaller than the specified value. (for more information see <a href="#">lp.control</a>).</p>  |

epslevel	a character string giving the type of thresholds (for more information see <a href="#">lp.control</a> ).
epsb	a numeric giving the tolerance for the right-hand-side (for more information see <a href="#">lp.control</a> ).
epsd	a numeric (for more information see <a href="#">lp.control</a> ).
epsel	a numeric (for more information see <a href="#">lp.control</a> ).
epsint	a numeric (for more information see <a href="#">lp.control</a> ).
epsperturb	a numeric (for more information see <a href="#">lp.control</a> ).
epspivot	a numeric (for more information see <a href="#">lp.control</a> ).
improve	a character vector (for more information see <a href="#">lp.control</a> ).
infinite	a numeric (for more information see <a href="#">lp.control</a> ).
maxpivot	a integer (for more information see <a href="#">lp.control</a> ).
mip.gap	a numeric vector (for more information see <a href="#">lp.control</a> ).
negrange	a numeric (for more information see <a href="#">lp.control</a> ).
obj.in.bas	a logical (for more information see <a href="#">lp.control</a> ).
pivoting	a character vector (for more information see <a href="#">lp.control</a> ).
presolve	a character vector (for more information see <a href="#">lp.control</a> ).
scaletlimit	a numeric (for more information see <a href="#">lp.control</a> ).
scaling	a character vector (for more information see <a href="#">lp.control</a> ).
simplextype	a character vector which can take one of the following values c("primal"), c("dual"), c("primal", "dual") or c("dual", "primal") (for more information see <a href="#">lp.control</a> ).
timeout	a integer giving the number of seconds till a timeout occurs (for more information see <a href="#">lp.control</a> ).

---

 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

```

---

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.

**Details**

The optimization problems can be read from the three file formats "lp", "mps" and "freemps". Where it seems important to note that the "lp" format refers to lpsolve's native file format (<https://lpsolve.sourceforge.net/5.5/lp-format.htm>) and not to the CPLEX LP format.

**Examples**

```
## Not run:
op <- read.lp("optimization_problem.lp")
sol <- ROI_solve(op)
solution(sol)

## End(Not run)
```

---

write.lp

*Write Optimization Problem*


---

**Description**

Write an optimization problem to a file.

**Usage**

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

**Arguments**

x	an object of type OP.
file	a character giving the name of the file the optimization problem is written to.
type	a character giving the name of the file format used to store the optimization problem.

**Details**

The optimization problems can be written to the three file formats "lp", "mps" and "freemps". Where it seems important to note that the "lp" format refers to lpsolve's native file format (<https://lpsolve.sourceforge.net/5.5/lp-format.htm>) and not to the CPLEX LP format.

**Examples**

```
## Not run:
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)),
        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`, [2](#)

`Example-1`, [3](#)

`lp.control`, [2](#), [3](#)

`read.lp`, [4](#)

`set.basis`, [2](#)

`set.branch.mode`, [2](#)

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