Package ‘lpSolve’

December 10, 2023

Version 5.6.20
Title Interface to ‘Lp_solve’ v. 5.5 to Solve Linear/Integer Programs
Author Michel Berkelaar and others
Maintainer Gábor Csárdi <csardi.gabor@gmail.com>
Description Lp_solve is freely available (under LGPL 2) software for solving linear, integer and mixed integer programs. In this implementation we supply a ``wrapper” function in C and some R functions that solve general linear/integer problems, assignment problems, and transportation problems. This version calls lp_solve version 5.5.
License LGPL-2
URL https://github.com/gaborcsardi/lpSolve
BugReports https://github.com/gaborcsardi/lpSolve/issues
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2023-12-10 13:40:02 UTC

R topics documented:

lp ......................................................... 2
lp.assign ............................................... 4
lp.object ............................................... 6
lp.transport .......................................... 7
make.q8 ................................................. 8
print.lp ................................................ 9

Index 11
Description

Interface to lp_solve linear/integer programming system

Usage

lp (direction = "min", objective.in, const.mat, const.dir, const.rhs,
transpose.constraints = TRUE, int.vec, presolve=0, compute.sens=0,
binary.vec, all.int=FALSE, all.bin=FALSE, scale = 196, dense.const,
num.bin.solns=1, use.rw=FALSE, timeout = 0L)

Arguments

direction Character string giving direction of optimization: "min" (default) or "max."
objective.in Numeric vector of coefficients of objective function
const.mat Matrix of numeric constraint coefficients, one row per constraint, one column
per variable (unless transpose.constraints = FALSE; see below).
const.dir Vector of character strings giving the direction of the constraint: each value
should be one of "<", "<=", "=", "==", ">", or ">=". (In each pair the two values
are identical.)
const.rhs Vector of numeric values for the right-hand sides of the constraints.
transpose.constraints By default each constraint occupies a row of const.mat, and that matrix needs to
be transposed before being passed to the optimizing code. For very large con-
straint matrices it may be wiser to construct the constraints in a matrix column-
by-column. In that case set transpose.constraints to FALSE.
int.vec Numeric vector giving the indices of variables that are required to be integer.
The length of this vector will therefore be the number of integer variables.
presolve Numeric: presolve? Default 0 (no); any non-zero value means "yes." Currently
ignored.
compute.sens Numeric: compute sensitivity? Default 0 (no); any non-zero value means "yes."
binary.vec Numeric vector like int.vec giving the indices of variables that are required to
be binary.
all.int Logical: should all variables be integer? Default: FALSE.
all.bin Logical: should all variables be binary? Default: FALSE.
scale Integer: value for lpSolve scaling. Details can be found in the lpSolve docu-
mentation. Set to 0 for no scaling. Default: 196
dense.const Three column dense constraint array. This is ignored if const.mat is supplied.
Otherwise the columns are constraint number, column number, and value; there
should be one row for each non-zero entry in the constraint matrix.
num.bin.solns  Integer: if all.bin=TRUE, the user can request up to num.bin.solns optimal solutions to be returned.

use.rw Logical: if TRUE and num.bin.solns > 1, write the lp out to a file and read it back in for each solution after the first. This is just to defeat a bug somewhere. Although the default is FALSE, we recommend you set this to TRUE if you need num.bin.solns > 1, until the bug is found.

timeout Integer: timeout variable in seconds, defaults to 0L which means no limit is set.

Details
This function calls the lp.solve 5.5 solver. That system has many options not supported here. The current version is maintained at https://lpsolve.sourceforge.net/5.5/

Note that every variable is assumed to be >= 0!

Value
An lp object. See lp.object for details.

Author(s)
Sam Buttrey, <buttrey@nps.edu>

See Also
lp.assign, lp.transport

Examples

```r
# Set up problem: maximize
# x1 + 9 x2 + x3 subject to
# x1 + 2 x2 + 3 x3 <= 9
# 3 x1 + 2 x2 + 2 x3 <= 15
#
f.obj <- c(1, 9, 1)
f.con <- matrix (c(1, 2, 3, 3, 2, 2), nrow=2, byrow=TRUE)
f.dir <- c("<=", "<=")
f.rhs <- c(9, 15)
#
# Now run.
#
lp ("max", f.obj, f.con, f.dir, f.rhs)
## Not run: Success: the objective function is 40.5
lp ("max", f.obj, f.con, f.dir, f.rhs)$solution
## Not run: [1] 0.0 4.5 0.0
#
# The same problem using the dense constraint approach:
#
f.con.d <- matrix (c(rep (1:2,each=3), rep (1:3, 2), t(f.con)), ncol=3)
lp ("max", f.obj, f.dir, f.rhs, dense.const=f.con.d)
## Not run: Success: the objective function is 40.5
```
# Get sensitivities

lp ("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.from
## Not run: [1] -1e+30 2e+00 -1e+30
lp ("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.to
## Not run: [1] 4.50e+00 1.00e+30 1.35e+01

# Right now the dual values for the constraints and the variables are
# combined, constraints coming first. So in this example...

lp ("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals
## Not run: [1] 4.5 0.0 -3.5 0.0 -10.5

# ...the duals of the constraints are 4.5 and 0, and of the variables,
# -3.5, 0.0, -10.5. Here are the lower and upper limits on these:

lp ("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.from
## Not run: [1] 0e+00 -1e+30 -1e+30 -1e+30 -6e+00
lp ("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.to
## Not run: [1] 1.5e+01 1.0e+30 3.0e+00 1.0e+30 3.0e+00

# Run again, this time requiring that all three variables be integer

lp ("max", f.obj, f.con, f.dir, f.rhs, int.vec=1:3)
## Not run: Success: the objective function is 37
lp ("max", f.obj, f.con, f.dir, f.rhs, int.vec=1:3)$solution
## Not run: [1] 1 4 0

# You can get sensitivities in the integer case, but they're harder to
# interpret.

lp ("max", f.obj, f.con, f.dir, f.rhs, int.vec=1:3, compute.sens=TRUE)$duals
## Not run: [1] 1 0 0 7 0

# Here's an example in which we want more than one solution to a problem
# in which all variables are binary: the 8-queens problem,
# with dense constraints.

chess.obj <- rep (1, 64)
q8 <- make.q8 ()
chess.dir <- rep (c("="", ","), c (16, 26))
chess.rhs <- rep (1, 42)
lp ('max', chess.obj, , chess.dir, chess.rhs, dense.const = q8,
    all.bin=TRUE, num.bin.solns=3)
Description

Interface to lp_solve linear/integer programming system specifically for solving assignment problems

Usage

lp.assign (cost.mat, direction = "min", presolve = 0, compute.sens = 0)

Arguments

cost.mat    Matrix of costs: the ij-th element is the cost of assigning source i to destination j.
direction   Character vector, length 1, containing either "min" (the default) or "max"
presolve    Numeric: presolve? Default 0 (no); any non-zero value means "yes." Currently ignored.
compute.sens Numeric: compute sensitivity? Default 0 (no); any non-zero value means "yes." In that case presolving is attempted.

Details

This is a particular integer programming problem. All the decision variables are assumed to be integers; each row has the constraint that its entries must add up to 1 (so that there is one 1 and the remaining entries are 0) and each column has the same constraint. This is assumed to be a minimization problem.

Value

An lp object. See documentation for details. The constraints are assumed (each row adds to 1, each column adds to 1, and no others) and are not returned.

Author(s)

Sam Buttrey, <buttrey@nps.edu>

See Also

lp, lp.transport

Examples

assign.costs <- matrix (c(2, 7, 7, 2, 7, 3, 2, 7, 2, 8, 10, 1, 9, 8, 2), 4, 4)
## Not run:
> assign.costs
[1,] 2 7 7 1
[2,] 7 7 2 9
[3,] 7 3 8 8
[4,] 2 2 10 2
lp.object

LP (linear programming) object

Description

Structure of lp object

Value

An lp.object is a list containing the following elements:

direction Optimization direction, as entered
x.count Number of variables in objective function
objective Vector of objective function coefficients, as entered
const.count Number of constraints entered
constraints Constraint matrix, as entered (not returned by lp.assign or lp.transport)
int.count Number of integer variables
int.vec Vector of integer variables' indices, as entered
objval Value of objective function at optimum
solution Vector of optimal coefficients
num.bin.solns Numeric indicator of number of solutions returned
status Numeric indicator: 0 = success, 2 = no feasible solution

Author(s)

Sam Buttrey, <buttrey@nps.edu>

See Also

lp, lp.assign, lp.transport
### Description

Interface to lp_solve linear/integer programming system specifically for solving transportation problems

### Usage

```r
lp.transport (cost.mat, direction="min", row.signs, row.rhs, col.signs, col.rhs, presolve=0, compute.sens=0, integers = 1:(nc*nr) )
```

### Arguments

- **cost.mat**: Matrix of costs; ij-th element is the cost of transporting one item from source i to destination j.
- **direction**: Character, length 1: "min" or "max"
- **row.signs**: Vector of character strings giving the direction of the row constraints: each value should be one of "\<", "\<=", "\="; "\==", ">", or "\>=." (In each pair the two values are identical.)
- **row.rhs**: Vector of numeric values for the right-hand sides of the row constraints.
- **col.signs**: Vector of character strings giving the direction of the column constraints: each value should be one of "\<", "\<=", "\="; "\==", ">", or "\>=."  
- **col.rhs**: Vector of numeric values for the right-hand sides of the column constraints.
- **presolve**: Numeric: presolve? Default 0 (no); any non-zero value means "yes." Currently ignored.
- **compute.sens**: Numeric: compute sensitivity? Default 0 (no); any non-zero value means "yes."
- **integers**: Vector of integers whose ith element gives the index of the ith integer variable. Its length will be the number of integer variables. Default: all variables are integer. Set to NULL to have no variables be integer.

### Details

This is a particular integer programming problem. All the decision variables are assumed to be integers, and there is one constraint per row and one per column (and no others). This is assumed to be a minimization problem.

### Value

An **lp** object. Constraints are implicit and not returned. See documentation for details.

### Author(s)

Sam Buttrey, <buttrey@nps.edu>
References


See Also

`lp.assign`, `lp.transport`

Examples

```r
# Transportation problem, Bronson, problem 9.1, p. 86
#
# Set up cost matrix
#
costs <- matrix(10000, 8, 5); costs[4,1] <- costs[-4,5] <- 0
costs[8,4] <- 10; costs[4,2:4] <- c(.7, 1.4, 2.1)
#
# Set up constraint signs and right-hand sides.
#
row.signs <- rep("<", 8)
row.rhs <- c(200, 300, 350, 200, 100, 50, 100, 150)
col.signs <- rep(">", 5)
col.rhs <- c(250, 100, 400, 500, 200)
#
# Run
#
lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)
## Not run: Success: the objective function is 7790
lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)$solution
## Not run:

[1,]  0  100  0  100  0
[2,]  0   0 300  0  0
[3,]  0   0  0 350  0
[4,] 200  0   0  0  0
[5,]  50  0   0  0 50
[6,]  0   0  0  0 50
[7,]  0   0 100  0  0
[8,]  0   0  0 50 100

## End(Not run)
```

`make.q8` Generate sparse constraint matrix for 8-queens problem
print.lp

Description
Generate sparse constraint matrix for 8-queens problem

Usage
make.q8()

Arguments
None.

Details
Sparse constraints come in a three-column matrix or data frame. Each row gives the row number, column number, and value of a particular non-zero entry in the constraint matrix. This function produces the sparse constraint matrix for the 8-queens problem (in which the object is to place eight queens on a chessboard with no two sharing a row, column or diagonal). The resulting sparse representation is 252 x 3, compared to 42 x 64 for the usual representation.

Value
A 252 x 3 numeric matrix. See lp for the complete example.

Author(s)
Sam Buttrey, <buttrey@nps.edu>

See Also
lp
Details

This function prints the objective function value, together with the word "Success" if the operation is successful, or an indication of the error if not. If multiple solutions have been produced (because this was an all-binary problem and lp was called with num.bin.solns > 1) the number of solutions is also displayed.

Value

None

Author(s)

Sam Buttrey. <buttrey@nps.edu>

See Also

lp, lp.assign, lp.transport
Index

* optimize
  lp. 2
  lp.assign. 4
  lp.object. 6
  lp.transport. 7
  make.q8. 8
  print.lp. 9

8-queens problem (make.q8). 8

lp. 2, 5–7, 9, 10
lp.assign. 3, 4, 6, 8–10
lp.object. 3, 6
lp.transport. 3, 5, 6, 7, 8–10

make.q8. 8

print.lp. 9