Package 'linprog'

February 20, 2015

Version 0.9-2

Date 2012/10/17

Title Linear Programming / Optimization

Author Arne Henningsen

Maintainer Arne Henningsen <arne.henningsen@gmail.com>

Depends R (>= 2.4.0), lpSolve

Description This package can be used to solve Linear Programming / Linear Optimization problems by using the simplex algorithm.

License GPL (>= 2)

URL http://linprog.r-forge.r-project.org/

Repository CRAN

Date/Publication 2012-10-17 11:03:12

NeedsCompilation no

R topics documented:

print.solveLP *Print Objects of Class solveLP*

Description

This method prints the results of the Linear Programming algorithm.

Usage

```
   ## S3 method for class 'solveLP'
print( x, digits=6, ...)
```

Arguments

x	an object returned by solveLP.
digits	number of digits to print.
...	currently ignored.

Value

print.solveLP invisibly returns the object given in argument x.

Author(s)

Arne Henningsen

See Also

solveLP, summary.solveLP, readMps, writeMps

Examples

```
## example of Steinhauser, Langbehn and Peters (1992)
## Not run: library( linprog )

## Production activities
cvec <- c(1800, 600, 600)  # gross margins
names(cvec) <- c("Milk","Bulls","Pigs")

## Constraints (quasi-fix factors)
bvec <- c(40, 90, 2500)  # endowment
names(bvec) <- c("Land","Stable","Labor")

## Needs of Production activities
Amat <- rbind( c(  0.7,   0.35,   0 ),
               c(  1.5,   1,      3 ),
               c( 50,    12.5,   20 ) )
```

```
## Maximize the gross margin
res <- solveLP( cvec, bvec, Amat, TRUE )

## print the results
print( res )
```

---

readMps *Read MPS Files*

Description

This function reads MPS files - the standard format for Linear Programming problems.

Usage

```
readMps( file, solve=FALSE, maximum=FALSE )
```

Arguments

file	a character string naming the file to read.
solve	logical. Should the problem be solved after reading it from the file (using solveLP)?
maximum	logical. Should we maximize or minimize (the default)?

Details

Equality constraints and 'greater than'-bounds are not implemented yet.

Value

readMps returns a list containing following objects:

name	the name of the Linear Programming problem.
cvec	vector c.
bvec	vector b.
Amat	matrix A.
res	if solve is TRUE, it contains the results of the solving process (an object of class solveLP).

Author(s)

Arne Henningsen

See Also

solveLP, writeMps

Examples

```
## example of Steinhauser, Langbehn and Peters (1992)
## Production activities
cvec <- c(1800, 600, 600)  # gross margins
names(cvec) <- c("Cows","Bulls","Pigs")

## Constraints (quasi-fix factors)
bvec <- c(40, 90, 2500)  # endowment
names(bvec) <- c("Land","Stable","Labor")

## Needs of Production activities
Amat <- rbind( c(  0.7,   0.35,   0 ),
               c(  1.5,   1,      3 ),
               c( 50,    12.5,   20 ) )

## Write to MPS file
writeMps( "steinh.mps", cvec, bvec, Amat, "Steinhauser" )

## delete all LP objects
rm( cvec, bvec, Amat )

## Read LP data from MPS file and solve it.
lp <- readMps( "steinh.mps", TRUE, TRUE )

## Print the results
lp$res
```

`solveLP` *Solve Linear Programming / Optimization Problems*

Description

Minimizes (or maximizes) $c'x$, subject to $Ax <= b$ and $x >= 0$.

Note that the inequality signs `<=` of the individual linear constraints in $Ax <= b$ can be changed with argument `const.dir`.

Usage

```
solveLP( cvec, bvec, Amat, maximum = FALSE,
   const.dir = rep( "<=", length( bvec ) ),
   maxiter = 1000, zero = 1e-9, tol = 1e-6, dualtol = tol,
   lpSolve = FALSE, solve.dual = FALSE, verbose = 0 )
```

Arguments

cvec	vector c (containing n elements).
bvec	vector b (containing m elements).
Amat	matrix A (of dimension $m \times n$).
maximum	logical. Should we maximize or minimize (the default)?
const.dir	vector of character strings giving the directions of the constraints: each value should be one of "<," "<=," "=," "==," ">," or ">=". (In each pair the two values are identical.)
maxiter	maximum number of iterations.
zero	numbers smaller than this value (in absolute terms) are set to zero.
tol	if the constraints are violated by more than this number, the returned component status is set to 3.
dualtol	if the constraints in the dual problem are violated by more than this number, the returned status is non-zero.
lpSolve	logical. Should the package 'lpSolve' be used to solve the LP problem?
solve.dual	logical value indicating if the dual problem should also be solved.
verbose	an optional integer variable to indicate how many intermediate results should be printed (0 = no output; 4 = maximum output).

Details

This function uses the Simplex algorithm of George B. Dantzig (1947) and provides detailed results (e.g. dual prices, sensitivity analysis and stability analysis).
If the solution $x = 0$ is not feasible, a 2-phase procedure is applied.
Values of the simplex tableau that are actually zero might get small (positive or negative) numbers due to rounding errors, which might lead to artificial restrictions. Therefore, all values that are smaller (in absolute terms) than the value of zero (default is 1e-10) are set to 0.
Solving the Linear Programming problem by the package lpSolve (of course) requires the installation of this package, which is available on CRAN (http://cran.r-project.org/src/contrib/PACKAGES.html#lpSolve). Since the lpSolve package uses C-code and this (linprog) package is not optimized for speed, the former is much faster. However, this package provides more detailed results (e.g. dual values, stability and sensitivity analysis).
This function has not been tested extensively and might not solve all feasible problems (or might even lead to wrong results). However, you can export your LP to a standard MPS file via writeMps and check it with other software (e.g. lp_solve, see ftp://ftp.es.ele.tue.nl/pub/lp_solve).
Equality constraints are not implemented yet.

Value

solveLP returns a list of the class solveLP containing following objects:

opt	optimal value (minimum or maximum) of the objective function.
solution	vector of optimal values of the variables.
iter1	iterations of Simplex algorithm in phase 1.

iter2	iterations of Simplex algorithm in phase 2.
basvar	vector of basic (=non-zero) variables (at optimum).
con	matrix of results regarding the constraints: 1st column = maximum values (=vector b); 2nd column = actual values; 3rd column = differences between maximum and actual values; 4th column = dual prices (shadow prices); 5th column = valid region for dual prices.
allvar	matrix of results regarding all variables (including slack variables): 1st column = optimal values; 2nd column = values of vector c; 3rd column = minimum of vector c that does *not* change the solution; 4th column = maximum of vector c that does *not* change the solution; 5th column = derivatives to the objective function; 6th column = valid region for these derivatives.
status	numeric. Indicates if the optimization did succeed: 0 = success; 1 = lpSolve did not succeed; 2 = solving the dual problem did not succeed; 3 = constraints are violated at the solution (internal error or large rounding errors); 4 = simplex algorithm phase 1 did not find a solution within the number of iterations specified by argument `maxiter`; 5 = simplex algorithm phase 2 did not find the optimal solution within the number of iterations specified by argument `maxiter`.
lpStatus	numeric. Return code of `lp` (only if argument `lpSolve` is `TRUE`).
dualStatus	numeric. Return code from solving the dual problem (only if argument `solve.dual` is `TRUE`).
maximum	logical. Indicates whether the objective function was maximized or minimized.
Tab	final 'Tableau' of the Simplex algorith.
lpSolve	logical. Has the package 'lpSolve' been used to solve the LP problem.
solve.dual	logical. Argument `solve.dual`.
maxiter	numeric. Argument `maxiter`.

Author(s)

Arne Henningsen

References

Dantzig, George B. (1951), *Maximization of a linear function of variables subject to linear inequalities*, in Koopmans, T.C. (ed.), Activity analysis of production and allocation, John Wiley \& Sons, New York, p. 339-347.

Steinhauser, Hugo; Cay Langbehn and Uwe Peters (1992), Einfuehrung in die landwirtschaftliche Betriebslehre. Allgemeiner Teil, 5th ed., Ulmer, Stuttgart.

Witte, Thomas; Joerg-Frieder Deppe and Axel Born (1975), Lineare Programmierung. Einfuehrung fuer Wirtschaftswissenschaftler, Gabler-Verlag, Wiesbaden.

See Also

readMps and writeMps

Examples

```
## example of Steinhauser, Langbehn and Peters (1992)
## Production activities
cvec <- c(1800, 600, 600)  # gross margins
names(cvec) <- c("Cows","Bulls","Pigs")

## Constraints (quasi-fix factors)
bvec <- c(40, 90, 2500)  # endowment
names(bvec) <- c("Land","Stable","Labor")

## Needs of Production activities
Amat <- rbind( c(  0.7,   0.35,   0 ),
               c(  1.5,   1,      3 ),
               c( 50,    12.5,   20 ) )

## Maximize the gross margin
solveLP( cvec, bvec, Amat, TRUE )


## example 1.1.3 of Witte, Deppe and Born (1975)
## Two types of Feed
cvec <- c(2.5, 2 )  # prices of feed
names(cvec) <- c("Feed1","Feed2")

## Constraints (minimum (<0) and maximum (>0) contents)
bvec <- c(-10, -1.5, 12)
names(bvec) <- c("Protein","Fat","Fibre")

## Matrix A
Amat <- rbind( c( -1.6,  -2.4 ),
               c( -0.5,  -0.2 ),
               c(  2.0,   2.0 ) )

## Minimize the cost
solveLP( cvec, bvec, Amat )

# the same optimisation using argument const.dir
solveLP( cvec, abs( bvec ), abs( Amat ), const.dir = c( ">=", ">=", "<=" ) )


## There are also several other ways to put the data into the arrays, e.g.:
bvec <- c( Protein = -10.0,
           Fat     =  -1.5,
           Fibre   =  12.0 )
cvec <- c( Feed1 = 2.5,
           Feed2 = 2.0 )
Amat <- matrix( 0, length(bvec), length(cvec) )
```

```
rownames(Amat) <- names(bvec)
colnames(Amat) <- names(cvec)
Amat[ "Protein", "Feed1" ] <-  -1.6
Amat[ "Fat",     "Feed1" ] <-  -0.5
Amat[ "Fibre",   "Feed1" ] <-   2.0
Amat[ "Protein", "Feed2" ] <-  -2.4
Amat[ "Fat",     "Feed2" ] <-  -0.2
Amat[ "Fibre",   "Feed2" ] <-   2.0
solveLP( cvec, bvec, Amat )
```

`summary.solveLP` *Summary Results for Objects of Class solveLP*

Description

These methods prepare and print summary results of the Linear Programming algorithm.

Usage

```
   ## S3 method for class 'solveLP'
summary(object,...)
   ## S3 method for class 'summary.solveLP'
print(x,...)
```

Arguments

`object`	an object returned by `solveLP`.
`x`	an object returned by `summary.solveLP`.
`...`	currently ignored.

Value

`summary.solveLP` returns an object of class `summary.solveLP`. `print.summary.solveLP` invisibly returns the object given in argument `x`.

Author(s)

Arne Henningsen

See Also

`solveLP`, `print.solveLP`, `readMps`, `writeMps`

Examples

```
## example of Steinhauser, Langbehn and Peters (1992)
## Not run: library( linprog )

## Production activities
cvec <- c(1800, 600, 600)  # gross margins
names(cvec) <- c("Milk","Bulls","Pigs")

## Constraints (quasi-fix factors)
bvec <- c(40, 90, 2500)  # endowment
names(bvec) <- c("Land","Stable","Labor")

## Needs of Production activities
Amat <- rbind( c(  0.7,   0.35,   0 ),
               c(  1.5,   1,      3 ),
               c( 50,    12.5,   20 ) )

## Maximize the gross margin
res <- solveLP( cvec, bvec, Amat, TRUE )

## prepare and print the summary results
summary( res )
```

`writeMps` *Write MPS Files*

Description

This function writes MPS files - the standard format for Linear Programming problems.

Usage

```
writeMps( file, cvec, bvec, Amat, name="LP" )
```

Arguments

`file`	a character string naming the file to write.
`cvec`	vector c.
`bvec`	vector b.
`Amat`	matrix A.
`name`	an optional name for the Linear Programming problem.

Details

The exported LP can be solved by running other software on this MPS file (e.g. `lp_solve`, see ftp://ftp.es.ele.tue.nl/pub/lp_solve).

Author(s)

Arne Henningsen

See Also

solveLP, readMps

Examples

```
## example of Steinhauser, Langbehn and Peters (1992)
## Production activities
cvec <- c(1800, 600, 600)  # gross margins
names(cvec) <- c("Cows","Bulls","Pigs")

## Constraints (quasi-fix factors)
bvec <- c(40, 90, 2500)  # endowment
names(bvec) <- c("Land","Stable","Labor")

## Needs of Production activities
Amat <- rbind( c(  0.7,   0.35,   0 ),
               c(  1.5,   1,      3 ),
               c( 50,    12.5,   20 ) )

## Write to MPS file
writeMps( "steinh.mps", cvec, bvec, Amat, "Steinhauser" )
```

Index