Package ‘lsei’

October 23, 2017

Title  Solving Least Squares or Quadratic Programming Problems under Equality/Inequality Constraints

Version  1.2-0

Date  2017-10-22

Description  It contains functions that solve least squares linear regression problems under linear equality/inequality constraints. Functions for solving quadratic programming problems are also available, which transform such problems into least squares ones first. It is developed based on the ‘Fortran’ program of Lawson and Hanson (1974, 1995), which is public domain and available at <http://www.netlib.org/lawson-hanson>.

License  GPL (>= 2)

URL  https://www.stat.auckland.ac.nz/~yongwang

NeedsCompilation  yes

Author  Yong Wang [aut, cre],
        Charles L. Lawson [aut],
        Richard J. Hanson [aut]

Maintainer  Yong Wang <yongwang@auckland.ac.nz>

Repository  CRAN

Date/Publication  2017-10-23 07:22:58 UTC

R topics documented:

hti ................................................................. 2
indx ............................................................... 3
lsei ............................................................... 4
matMaxs ......................................................... 6
nnls ............................................................. 7

Index  10
hfti  

Least Squares Solution using Householder Transformation

Description

Solves the least squares problem using Householder forward triangulation with column interchanges. It is an R interface to the HFTI function that is described in Lawson and Hanson (1974, 1995). Its Fortran implementation is public domain and is available at http://www.netlib.org/lawson-hanson.

Usage

hfti(a, b, tol=1e-7)

Arguments

a  Design matrix.

b  Response vector or matrix.

tol  Tolerance for determining the pseudorank.

Details

Given matrix a and vector b, hfti solves the least squares problem:

\[
\text{minimize } \|ax - b\|.
\]

Value

b  first krank elements contains the solution

krank  psuedo-rank

rnorm  Euclidean norm of the residual vector.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

lsei, nnls.
**indx**

**Examples**

```r
da = matrix(rnorm(10*4), nrow=10)
b = a * c(0,1,-1,1) + rnorm(10)
hfti(a, b)
```

---

**Index-finding in a Sorted Vector**

**Description**

For each of given values, `indx` finds the index of the value in a vector sorted in ascending order that the given value is barely greater than or equal to.

**Usage**

```r
indx(x, v)
```

**Arguments**

- `x`: vector of numeric values, the indices of which are to be found.
- `v`: vector of numeric values sorted in ascending order.

**Details**

For each `x[i]`, the function returns integer `j` such that

\[ v_j \leq x_i < v_{j+1} \]

where \( v_0 = -\infty \) and \( v_{n+1} = \infty \).

**Value**

Returns a vector of integers, that are indices of `x`-values in vector `v`.

**Author(s)**

Yong Wang <yongwang@auckland.ac.nz>

**Examples**

```r
indx(0:6, c(1:5,5))
indx(sort(rnorm(5)), -2:2)
```
Least Squares Solution under Equality and Inequality Constraints

Description

The `lsei` function solves a least squares problem under both equality and inequality constraints and implements the LSEI algorithm described in Lawson and Hanson (1974, 1995).

The `lsi` function solves a least squares problem under inequality constraints and implements the LSI algorithm described in Lawson and Hanson (1974, 1995).

The `ldp` function solves a least distance programming problem under inequality constraints and implements the LDP algorithm described in Lawson and Hanson (1974, 1995).

The `qp` function solves a quadratic programming problem, by transforming the problem into a least squares one under the same equality and inequality constraints, which is then solved by function `lsei`.

The NNLS Fortran implementation used internally is downloaded from http://www.netlib.org/lawson-hanson.

Usage

```
lsei(a, b, c=NULL, d=NULL, e=NULL, f=NULL, lower=-Inf, upper=Inf)
lsi(a, b, e=NULL, f=NULL, lower=-Inf, upper=Inf)
ldp(e, f, tol=1e-15)
qp(q, p, c=NULL, d=NULL, e=NULL, f=NULL, lower=-Inf, upper=Inf, tol=1e-15)
```

Arguments

- `a`  Design matrix.
- `b`  Response vector.
- `c`  Matrix of numeric coefficients on the left-hand sides of equality constraints. If it is NULL, `c` and `d` are ignored.
- `d`  Vector of numeric values on the right-hand sides of equality constraints.
- `e`  Matrix of numeric coefficients on the left-hand sides of inequality constraints. If it is NULL, `e` and `f` are ignored.
- `f`  Vector of numeric values on the right-hand sides of inequality constraints.
- `q`  Matrix of numeric values for the quadratic term of a quadratic programming problem.
- `p`  Vector of numeric values for the linear term of a quadratic programming problem.
- `lower, upper`  Bounds on the solutions, as a way to specify such simple inequality constraints.
- `tol`  Tolerance, for checking compatibility of inequalities in `lsi` and for calculating pseudo-rank in `qp`. 
Details

Given matrices a, c and e, and vectors b, d and f, function `lsei` solves the least squares problem under both equality and inequality constraints:

\[
\text{minimize } ||ax - b||, \\
\text{subject to } cx = d, ex \geq f.
\]

Function `lsei` solves the least squares problem under inequality constraints:

\[
\text{minimize } ||ax - b||, \\
\text{subject to } ex \geq f.
\]

Function `ldp` solves the least distance programming problem under inequality constraints:

\[
\text{minimize } ||x||, \\
\text{subject to } ex \geq f.
\]

Function `qp` solves the quadratic programming problem:

\[
\text{minimize } \frac{1}{2}x^Tqx + p^Tx, \\
\text{subject to } cx = d, ex \geq f.
\]

Value

A vector of the solution values

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

`nnls`, `hfti`. 
Examples

```r
beta = c(rnorm(2), 1)
beta[beta<0] = 0
beta = beta / sum(beta)
a = matrix(rnorm(18), ncol=3)
b = a %*% beta + rnorm(3, sd=.1)
c = t(rep(1, 3))
d = 1
e = diag(1,3)
f = rep(0,3)
lsis(a, b) # under no constraint
lsis(a, b, c, d) # under eq. constraints
lsis(a, b, e=e, f=f) # under ineq. constraints
lsis(a, b, c, d, e, f) # under eq. and ineq. constraints
lsis(a, b, rep(1,3), 1, lower=0) # same solution
q = crossprod(a)
p = -drop(crossprod(b, a))
qp(q, p, rep(1,3), 1, lower=0) # same solution
```

## example from Lawson and Hanson (1974), p.170

```r
a = cbind(c(.25,.5,.5,.8),rep(1,4))
b = c(.5,.6,.7,1.2)
e = cbind(c(1,0,-1),c(0,1,-1))
f = c(0,0,-1)
lsis(a, b, e, f) # Solution: 0.6213152 0.3786848
```

## Example from Lawson and Hanson (1974), p.171:

```r
e = cbind(c(-.207,-.392,.599), c(2.558, -1.351, -1.206))
f = c(-1.3,-.084,.384)
ldp(e, f) # Solution: 0.1268538 -0.2554018
```

---

**matMaxs**

*Row or Column Maximum Values of a Matrix*

**Description**

Finds either row or column maximum values of a matrix.

**Usage**

```r
matMaxs(x, dim=1)
```

**Arguments**

- `x` numeric matrix.
- `dim` =1, for row maximum values; =2, for column maximum values.

**Details**

Matrix x may contain Inf or -Inf, but not NA or NaN.
Value

Returns a numeric vector with row or column maximum values. The function is very much the same as using apply(x, 1, max) or apply(x, 2, max), but faster.

Author(s)

Yong Wang <yongwang@auckland.ac.nz>

Examples

```r
x = cbind(c(1:4,Inf), 5:1)
matMaxs(x)
matMaxs(x, 2)
```

Description

`nnls` solves the least squares problem under nonnegativity (NN) constraints. It is an R interface to the NNLS function that is described in Lawson and Hanson (1974, 1995). Its Fortran implementation is public domain and available at http://www.netlib.org/lawson-hanson.

`pnnls` solves the least squares problem when only part of the coefficients are subject to nonnegativity constraints. It also allows the NN-restricted coefficients to be further restricted to have a fixed positive sum.

`pnnqp` solves the quadratic programming problem when solution values are partly subject to nonnegativity constraints. It also allows the NN-restricted coefficients to be further restricted to have a fixed positive sum.

These functions are particularly useful for finding zeros exactly.

Usage

```r
nnls(a, b)
pnnls(a, b, k=0, sum=NULL)
pnnqp(q, p, k=0, sum=NULL, tol=1e-20)
```

Arguments

- `a`: Design matrix.
- `b`: Response vector.
- `k`: Integer, meaning that the first k coefficients are not NN-restricted.
- `sum`:
  - = NULL, if NN-restricted coefficients are not further restricted to have a fixed sum;
  - = a positive value, if NN-restricted coefficients are further restricted to have a fixed positive sum.
\begin{itemize}
\item \textbf{q} \quad \text{Positive semidefinite matrix of numeric values for the quadratic term of a quadratic programming problem.}
\item \textbf{p} \quad \text{Vector of numeric values for the linear term of a quadratic programming problem.}
\item \textbf{tol} \quad \text{Tolerance used for calculating pseudo-rank of } q.\
\end{itemize}

\section*{Details}

Given matrix \(a\) and vector \(b\), \texttt{nnls} solves the nonnegativity least squares problem:

\[
\begin{align*}
\text{minimize} & \quad \|ax - b\|, \\
\text{subject to} & \quad x \geq 0.
\end{align*}
\]

Function \texttt{pnnls} also solves the above nonnegativity least squares problem when \(k=0\), but it may also leave the first \(k\) coefficients unrestricted. The output value of \(k\) can be different from the input, if \(a\) has linearly dependent columns. If \(\text{sum}\) is a positive value, \texttt{pnnls} solves the problem by further restricting that the NN-restricted coefficients must sum to the given value.

Function \texttt{pnnqp} solves the quadratic programming problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T q x + p^T x,
\end{align*}
\]

when only some or all coefficients are restricted by nonnegativity. The quadratic programming problem is solved by transforming the problem into a least squares one under the same constraints, which is then solved by function \texttt{pnnls}. Arguments \(k\) and \(\text{sum}\) have the same meanings as for \texttt{pnnls}.

Functions \texttt{nnls}, \texttt{pnnls} and \texttt{pnnqp} are able to return any zero-valued solution as 0 exactly. This differs from functions \texttt{lsei} and \texttt{qp}, which may produce very small values for exactly 0s, thanks to numerical errors.

\section*{Value}

\begin{itemize}
\item \textbf{x} \quad \text{Solution}
\item \textbf{r} \quad \text{The upper-triangular matrix } Q*a, \text{ pivoted by variables in the order of } \text{index}, \text{ when } \text{sum}=\text{NULL}. \text{ If } \text{sum} > 0, \text{ r is for the transformed } a.
\item \textbf{b} \quad \text{The vector } Q*b, \text{ pivoted by variables in the order of } \text{index}, \text{ when } \text{sum}=\text{NULL}. \text{ If } \text{sum} > 0, \text{ b is for the transformed } b.
\item \textbf{index} \quad \text{Indices of the columns of } r; \text{ those unrestricted and in the positive set are first given, and then those in the zero set.}
\item \textbf{rnorm} \quad \text{Euclidean norm of the residual vector.}
\item \textbf{mode} \quad \begin{align*}
= 1, & \text{ successful computation;} \\
= 2, & \text{ bad dimensions of the problem;} \\
= 3, & \text{ iteration count exceeded (more than 3 times the number of variables iterations).}
\end{align*}
\item \textbf{k} \quad \text{Number of the first few coefficients that are truly not NN-restricted.}
\end{itemize}
Author(s)

Yong Wang <yongwang@auckland.ac.nz>

References


See Also

lsei, hfti.

Examples

```r
a = matrix(rnorm(40), nrow=10)
b = drop(a %*% c(0,1,-1,1)) + rnorm(10)
nlsl(c, b)$x # constraint x > 0
pnlsl(c, b, k=0)$x # same as nlsl(c, b)
pnlsl(c, b, k=2)$x # first two coeffs are not NN-constrained
pnlsl(c, b, k=2, sum=1)$x # NN-constrained coeffs must sum to 1
pnlsl(c, b, k=2, sum=2)$x # NN-constrained coeffs must sum to 2
q = crossprod(a)
p = -drop(crossprod(b, a))
pnnqp(q, p, k=2, sum=2)$x # same solution
pnlsl(c, b, sum=1)$x # zeros found exactly
pnnqp(q, p, sum=1)$x # zeros found exactly
lsei(c, b, rep(1,4), 1, lower=0) # zeros not so exact
```
Index

*Topic algebra
  hfti, 2
  indx, 3
  lsei, 4
  matMaxs, 6
  nnls, 7
*Topic array
  hfti, 2
  indx, 3
  lsei, 4
  matMaxs, 6
  nnls, 7
  hfti, 2, 5, 9
  indx, 3
  ldp (lsei), 4
  lsei, 2, 4, 9
  lsi (lsei), 4
  matMaxs, 6
  nnls, 2, 5, 7
  pnnls (nnls), 7
  pnnqp (nnls), 7
  qp (lsei), 4