lsglmsolve( ) — Solves a general Gauss–Markov linear model problem

Description Syntax Remarks and examples Conformability Diagnostics Also see

Description

`lsglmsolve(`A`,` B`,` d`,` x`,` y`)` finds the solution to a general Gauss–Markov linear model: $\min_x \|y\|_2$ subject to $d = A\,x + B\,y$ with no return value.

`_lsglmsolve(`A`,` B`,` d`,` x`,` y`)` does the same thing except that it returns `0` if a solution was found and `1` otherwise. If `1` is returned, x and y are overwritten with vectors of missing values.

`_lsglm_lapacke(`A`,` B`,` d`,` x`,` y`)` is the interface to the LAPACK routines that do the work. It returns `0` if a solution was found and `1` otherwise. Direct use of `_lsglm_lapacke()` is not recommended.

Note that these functions can be used only when `set lapack_mkl on` is in effect on Windows or Linux or when `set lapack_openblas on` is in effect on Mac; see [M-1] **LAPACK**.

Syntax

void	`lsglmsolve(`A`,` B`,` d`,` x`,` y`)`
real scalar	`_lsglmsolve(`A`,` B`,` d`,` x`,` y`)`
real scalar	`_lsglm_lapacke(`A`,` B`,` d`,` x`,` y`)`

where inputs are

A:	$n \times m$ *numeric matrix*
B:	$n \times p$ *numeric matrix*
d:	$n \times 1$ or $1 \times n$ *numeric vector*

and outputs are

x:	$m \times 1$ *numeric vector*
y:	$p \times 1$ *numeric vector*
result:	*real scalar*

where $m \le n \le m + p$, the rank of matrix A is m, and the rank of the following matrix is n:

$$\begin{bmatrix} A & B \\ n\times m & n\times p \end{bmatrix}$$

Remarks and examples

Remarks are presented under the following headings:

Introduction
Examples

Introduction

The above functions solve a general Gauss–Markov linear model: $\min_x \|y\|_2$ subject to $d = A\,x + B\,y$.

To obtain a unique solution, the functions require that

1. $m \le n \le m + p$,
2. the rank of matrix A is m, and
3. the rank of the following matrix is n:

$$\begin{bmatrix} \underset{n\times m}{A} & \underset{n\times p}{B} \end{bmatrix}$$

The solution is found with the underlying LAPACK routines using a generalized QR factorization of (A, B).

Examples

▷ Example 1: Least squares with equality constraints

Given A, B, and d, we can find x and y, satisfying $\min_x \|y\|_2$ subject to $A\,x + B\,y = d$ using

```
: A = (3, 2 \ 2, 10 \ 1, 1.5)
: B = (2, 5 \ 1, 6 \ 5, 2)
: d = (10 \ 2 \ 1)
: A
```

	1	2
1	3	2
2	2	10
3	1	1.5

```
: B
```

	1	2
1	2	5
2	1	6
3	5	2

```
: d
```

	1
1	10
2	2
3	1

```
: lsglmsolve(A, B, d, x = ., y = .)
```

```
: x
                  1
    +-------------+
  1 |   4.00221166|
  2 |  -.5527209235|
    +-------------+

: y
                  1
    +-------------+
  1 |  -.4315840528|
  2 |  -.0076050053|
    +-------------+
```

We can also use the `_lsglmsolve()` function to get the same solution as above and a return code of `0`:

```
: A = (3, 2 \ 2, 10 \ 1, 1.5)
: B = (2, 5 \ 1, 6 \ 5, 2)
: d = (10 \ 2 \ 1)
:
: _lsglmsolve(A, B, d, x = ., y = .)
  0
: x
                  1
    +-------------+
  1 |   4.00221166|
  2 |  -.5527209235|
    +-------------+

: y
                  1
    +-------------+
  1 |  -.4315840528|
  2 |  -.0076050053|
    +-------------+
```

◁

Conformability

`lsglmsolve(`A`, `B`, `d`, `x`, `y`)`:

input:

A:	$n \times m$
B:	$n \times p$
d:	$n \times 1$ or $1 \times n$

output:

x:	$m \times 1$
y:	$p \times 1$

`_lsglmsolve(`A`, `B`, `d`, `x`, `y`)`:

input:

A:	$n \times m$
B:	$n \times p$
d:	$n \times 1$ or $1 \times n$

output:

x:	$m \times 1$
y:	$p \times 1$
result:	1×1

`_lsglm_lapacke(`A`, `B`, `d`, `x`, `y`)` :

input:

A:	$n \times m$
B:	$n \times p$
d:	$n \times 1$ or $1 \times n$

output:

x:	$m \times 1$
y:	$p \times 1$
result:	1×1

Diagnostics

`lsglmsolve(`A`, `B`, ...)`, `_lsglmsolve(`A`, `B`, ...)`, and `_lsglm_lapacke(`A`, `B`, ...)` return a result containing missing if A, B, or d contains missing values. If the conditions in *Introduction* above are not satisfied, the functions will try to find a solution, which will either produce unstable results or abort with error. The functions abort with error if `set lapack_mkl on` is not in effect on Windows or Linux or when `set lapack_openblas on` is not in effect on Mac.

`_lsglmsolve(`A`, `B`, ...)` and `_lsglm_lapacke(`A`, `B`, ...)` abort with error if A, B, or d is a view.

`_lsglm_lapacke(`A`, `B`, ...)` aborts with error if A, B, and d are not all real or all complex. `_lsglm_lapacke(`A`, `B`, ...)` should not be used directly; use `_lsglmsolve()`.

Also see

[M-5] **cholsolve( )** — Solve AX=B for X using Cholesky decomposition

[M-5] **lsesolve( )** — Solve Ax=c for x with equality constraints using least-squares method

[M-5] **lssolve( )** — Solve AX=B for X using least-squares method

[M-5] **lusolve( )** — Solve AX=B for X using LU decomposition

[M-5] **qrsolve( )** — Solve AX=B for X using QR decomposition

[M-5] **_solvemat( )** — Solve AX=B for X

[M-5] **svsolve( )** — Solve AX=B for X using singular value decomposition

[M-4] **Matrix** — Matrix functions

[M-4] **Solvers** — Functions to solve AX=B and to obtain A inverse