

qrsolve() — Solve $AX=B$ for X using QR decomposition

[Description](#) [Syntax](#) [Remarks and examples](#) [Conformability](#)
[Diagnostics](#) [Also see](#)

Description

`qrsolve(A, B, ...)` uses QR decomposition to solve $AX = B$ and returns X . When A is singular or nonsquare, `qrsolve()` computes a least-squares generalized solution. When *rank* is specified, in it is placed the rank of A .

`_qrsolve(A, B, ...)`, does the same thing, except that it destroys the contents of A and it overwrites B with the solution. Returned is the rank of A .

In both cases, *tol* specifies the tolerance for determining whether A is of full rank. *tol* is interpreted in the standard way—as a multiplier for the default if $tol > 0$ is specified and as an absolute quantity to use in place of the default if $tol \leq 0$ is specified; see [\[M-1\] tolerance](#).

Syntax

numeric matrix `qrsolve(A, B)`
numeric matrix `qrsolve(A, B, rank)`
numeric matrix `qrsolve(A, B, rank, tol)`

real scalar `_qrsolve(A, B)`
real scalar `_qrsolve(A, B, tol)`

where

A: *numeric matrix*
B: *numeric matrix*
rank: irrelevant; *real scalar* returned
tol: *real scalar*

Remarks and examples

stata.com

`qrsolve(A, B, ...)` is suitable for use with square and possibly rank-deficient matrix A , or when A has more rows than columns. When A is square and full rank, `qrsolve()` returns the same solution as `lusolve()` (see [\[M-5\] lusolve\(\)](#)), up to roundoff error. When A is singular, `qrsolve()` returns a generalized (least-squares) solution.

Remarks are presented under the following headings:

[Derivation](#)
[Relationship to inversion](#)
[Tolerance](#)

Derivation

We wish to solve for X

$$AX = B \tag{1}$$

Perform QR decomposition on A so that we have $A = QRP'$. Then (1) can be rewritten as

$$QRP'X = B$$

Premultiplying by Q' and remembering that $Q'Q = QQ' = I$, we have

$$RP'X = Q'B \tag{2}$$

Define

$$Z = P'X \tag{3}$$

Then (2) can be rewritten as

$$RZ = Q'B \tag{4}$$

It is easy to solve (4) for Z because R is upper triangular. Having Z , we can obtain X via (3), because $Z = P'X$, premultiplied by P (and if we remember that $PP' = I$), yields

$$X = PZ$$

For more information on QR decomposition, see [M-5] `qrd()`.

Relationship to inversion

For a general discussion, see *Relationship to inversion* in [M-5] `lusolve()`.

For an inverse based on QR decomposition, see [M-5] `qrinv()`. `qrinv(A)` amounts to `qrsolve(A, I(rows(A)))`, although it is not actually implemented that way.

Tolerance

The default tolerance used is

$$eta = 1e-13 * trace(abs(R))/rows(R)$$

where R is the upper-triangular matrix of the QR decomposition; see *Derivation* above. When A is less than full rank, by, say, d degrees of freedom, then R is also rank deficient by d degrees of freedom and the bottom d rows of R are essentially zero. If the i th diagonal element of R is less than or equal to eta , then the i th row of Z is set to zero. Thus if the matrix is singular, `qrsolve()` provides a generalized solution.

If you specify $tol > 0$, the value you specify is used to multiply eta . You may instead specify $tol \leq 0$, and then the negative of the value you specify is used in place of eta ; see [M-1] `tolerance`.

Conformability

`qrsolve(A, B, rank, tol)`:

input:

A: $m \times n$, $m \geq n$
B: $m \times k$
tol: 1×1 (optional)

output:

rank: 1×1 (optional)
result: $n \times k$

`_qrsolve(A, B, tol)`:

input:

A: $m \times n$, $m \geq n$
B: $m \times k$
tol: 1×1 (optional)

output:

A: 0×0
B: $n \times k$
result: 1×1

Diagnostics

`qrsolve(A, B, ...)` and `_qrsolve(A, B, ...)` return a result containing missing if A or B contain missing values.

`_qrsolve(A, B, ...)` aborts with error if A or B are views.

Also see

[M-5] [qrinv\(\)](#) — Generalized inverse of matrix via QR decomposition

[M-5] [qrd\(\)](#) — QR decomposition

[M-5] [solvelower\(\)](#) — Solve $AX=B$ for X , A triangular

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

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

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

[M-5] [solve_tol\(\)](#) — Tolerance used by solvers and inverters

[M-4] [matrix](#) — Matrix functions

[M-4] [solvers](#) — Functions to solve $AX=B$ and to obtain A inverse