

invsym() — Symmetric real matrix inversion

Description Diagnostics	Syntax Also see	Remarks and examples	Conformability
----------------------------	--------------------	----------------------	----------------

Description

`invsym(A)` returns a generalized inverse of real, symmetric, positive-semidefinite matrix A .

`invsym(A, order)` does the same but allows you to specify which columns are to be swept first.

`_invsym(A)` and `_invsym(A, order)` do the same thing as `invsym(A)` and `invsym(A, order)` except that A is replaced with the generalized inverse result rather than the result being returned.

`_invsym()` uses less memory than `invsym()`.

`invsym()` and `_invsym()` are the routines Stata uses for calculating inverses of symmetric matrices.

Also see [M-5] [luinv\(\)](#), [M-5] [qrinv\(\)](#), and [M-5] [pinv\(\)](#) for general matrix inversion.

Syntax

```

real matrix    invsym(real matrix A)
real matrix    invsym(real matrix A, real vector order)

void           _invsym(real matrix A)
void           _invsym(real matrix A, real vector order)

```

Remarks and examples

Remarks are presented under the following headings:

- Definition of generalized inverse*
- Specifying the order in which columns are omitted*
- Determining the rank, or counting the number of omitted columns*
- Extracting linear dependencies*

Definition of generalized inverse

When the matrix is full rank and positive semidefinite, the matrix A becomes positive definite and the generalized inverse equals the inverse, that is, assuming A is $n \times n$,

$$\text{invsym}(A) * A = A * \text{invsym}(A) = I(n)$$

or, at least the above restriction is true up to roundoff error. When A is not full rank, the generalized inverse `invsym()` satisfies (ignoring roundoff error)

$$A * \text{invsym}(A) * A = A$$

```
invsym(A)*A*invsym(A) = invsym(A)
```

In the generalized case, there are an infinite number of inverse matrices that can satisfy the above restrictions. The one `invsym()` chooses is one that sets entire columns (and therefore rows) to 0, thus treating A as if it were of reduced dimension. Which columns (rows) are selected is determined on the basis of minimizing roundoff error.

In the above we talk as if determining whether a matrix is of full rank is an easy calculation. That is not true. Because of the roundoff error in the manufacturing and recording of A itself, columns that ought to be perfectly collinear will not be and yet you will still want `invsym()` to behave as if they were. `invsym()` tolerates a little deviation from collinearity in making the perfectly collinear determination.

Specifying the order in which columns are omitted

Left to make the decision itself, `invsym()` will choose which columns to drop (to set to 0) to minimize the overall roundoff error of the generalized inverse calculation. If column 1 and column 3 are collinear, then `invsym()` will choose to drop column 1 or column 3.

There are occasions, however, when you would like to ensure that a particular column or set of columns are not omitted. Perhaps column 1 corresponds to the intercept of a regression model and you would much rather, if one of columns 1 and 3 has to be omitted, that it be column 3.

Order allows you to specify the columns of the matrix that you would prefer not be omitted in the generalized inverse calculation. In the above example, to prevent column 1 from being omitted, you could code

```
invsym(A, 1)
```

If you would like to keep columns 1, 5, and 10 from being omitted, you can code

```
invsym(A, (1,5,10))
```

Specifying columns not to be omitted does not guarantee that they will not be omitted because they still might be collinear with each other or they might equal constants. However, if any other column can be omitted to satisfy your desire, it will be.

Determining the rank, or counting the number of omitted columns

If a column is omitted, 0 will appear on the corresponding diagonal entry. Hence, the rank of the original matrix can be extracted after inversion by `invsym()`:

```
: Ainv = invsym(A)
: rank = rows(Ainv)-diag0cnt(Ainv)
```

See [M-5] `diag0cnt()`.

Extracting linear dependencies

The linear dependencies can be read from the rows of $A*\text{invsym}(A)$:

```

: A*invsym(A)
      1      2      3
1      0     -1      1
2      0      1  -2.22045e-16
3      0      0      1

```

The above is interpreted to mean

$$x_1 = -x_2 + x_3$$

$$x_2 = x_2$$

$$x_3 = x_3$$

ignoring roundoff error.

Conformability

`invsym(A)`, `invsym(A, order)`:

```

A:      n × n
order:  1 × k  or  k × 1, k ≤ n  (optional)
result:  n × n

```

`_invsym(A)`, `_invsym(A, order)`:

```

A:      n × n
order:  1 × k  or  k × 1, k ≤ n  (optional)
output:
A:      n × n

```

Diagnostics

`invsym(A)`, `invsym(A, order)`, `_invsym(A)`, and `_invsym(A, order)` assume that A is symmetric; they do not check. If A is nonsymmetric, they treat it as if it were symmetric and equal to its upper triangle.

`invsym()` and `_invsym()` return a result containing missing values if A contains missing values.

`_invsym()` aborts with error if A is a view. Both functions abort with argument-out-of-range error if `order` is specified and contains values less than 1, greater than `rows(A)`, or the same value more than once.

`invsym()` and `_invsym()` return a matrix of zeros if A is not positive semidefinite.

Also see

[M-5] **cholinv()** — Symmetric, positive-definite matrix inversion

[M-5] **diag0cnt()** — Count zeros on diagonal

[M-5] **luinv()** — Square matrix inversion

[M-5] **pinv()** — Moore–Penrose pseudoinverse

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

[M-4] **Matrix** — Matrix functions

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