Title

stata.com

svd( ) — Singular value decomposition

Description

`svd(A, U, s, Vt)` calculates the singular value decomposition of A: $m \times n$, $m \geq n$, returning the result in U, s, and Vt. Singular values returned in s are sorted from largest to smallest.

`svdsv(A)` returns the singular values of A: $m \times n$, $m \geq n$ or $m < n$ (that is, no restriction), in a column vector of length min(m,n). U and Vt are not calculated.

`_svd(A, s, Vt)` does the same as `svd()`, except that it returns U in A. Use of `_svd()` conserves memory.

`_svdsv(A)` does the same as `svdsv()`, except that, in the process, it destroys A. Use of `_svdsv()` conserves memory.

`_svd_la()` is the interface to the LAPACK SVD routines and is used in the implementation of the previous functions. There is no reason you should want to use it.

Syntax

void	`svd(`*numeric matrix A*`, U, s, Vt)`
real colvector	`svdsv(`*numeric matrix A*`)`
void	`_svd(`*numeric matrix A*`, s, Vt)`
real colvector	`_svdsv(`*numeric matrix A*`)`
real scalar	`_svd_la(`*numeric matrix A*`, s, Vt)`

Remarks and examples

stata.com

Remarks are presented under the following headings:

Documented here is the thin SVD, appropriate for use with A: $m \times n$, $m \geq n$. See [M-5] **fullsvd( )** for the full SVD, appropriate for use in all cases. The relationship between the two is discussed in *Relationship between the full and thin SVDs* in [M-5] **fullsvd( )**.

Use of the thin SVD—the functions documented here—is preferred when $m \geq n$.

Introduction

The SVD is used to compute accurate solutions to linear systems and least-squares problems, to compute the 2-norm, and to determine the numerical rank of a matrix.

The singular value decomposition (SVD) of A: $m \times n$, $m \geq n$, is given by

$$A = U * \text{diag}(s) * V'$$

where

U:	$m \times n$ and $U'U = \text{I}(n)$
s:	$n \times 1$
V:	$n \times n$ and orthogonal (unitary)

When A is complex, the transpose operator $'$ is understood to mean the conjugate transpose operator.

Vector s contains the singular values, and those values are real even when A is complex. s is ordered so that the largest singular value appears first, then the next largest, and so on.

Function `svd(A, U, s, Vt)` returns U, s, and $Vt = V'$.

Function `svdsv(A)` returns s, omitting the calculation of U and Vt. Also, whereas `svd()` is suitable for use only in the case $m \geq n$, `svdsv()` may be used in all cases.

Possibility of convergence problems

It is possible, although exceedingly unlikely, that the SVD routines could fail to converge. `svd()`, `svdsv()`, `_svd()`, and `_svdsv()` then return singular values in s equal to missing.

In coding, it is perfectly acceptable to ignore this possibility because (1) it is so unlikely and (2) even if the unlikely event occurs, the missing values will properly reflect the situation. If you do wish to check, in addition to checking `missing(s)>0` (see [M-5] **missing()**), you must also check `missing(A)==0` because that is the other reason s could contain missing values. Convergence was not achieved if `missing(s) > 0 & missing(A)==0`. If you are calling one of the destructive-of-A versions of SVD, remember to check `missing(A)==0` before extracting singular values.

Conformability

`svd(A, U, s, Vt)`:

input:

A:	$m \times n$, $m \geq n$

output:

U:	$m \times n$
s:	$n \times 1$
Vt:	$n \times n$

`svdsv(A)`:

A:	$m \times n, \quad m \geq n \quad \text{or} \quad m < n$
result:	$\min(m, n) \times 1$

`_svd(A, s, Vt)`:

input:

A:	$m \times n, \quad m \geq n$

output:

A:	$m \times n,$ contains U
s:	$n \times 1$
Vt:	$n \times n$

`_svdsv(A)`:

input:

A:	$m \times n, \quad m \geq n \quad \text{or} \quad m < n$

output:

A:	0×0
result:	$\min(m, n) \times 1$

`_svd_la(A, s, Vt)`:

input:

A:	$m \times n, \quad m \geq n$

output:

A:	$m \times n,$ contains U
s:	$n \times 1$
Vt:	$n \times n$
result:	1×1

Diagnostics

`svd(A, U, s, Vt)` and `_svd(A, s, Vt)` return missing results if *A* contains missing. In all other cases, the routines should work, but there is the unlikely possibility of convergence problems, in which case missing results will also be returned; see *Possibility of convergence problems* above.

`svdsv(A)` and `_svdsv(A)` return missing results if *A* contains missing values or if there are convergence problems.

`_svd()` and `_svdsv()` abort with error if *A* is a view.

Direct use of `_svd_la()` is not recommended.

Singular value decompositions have multiple roots, as is engagingly explained by Stewart (1993). Eugenio Beltrami (Italy, 1835–1900), Camille Jordan (France, 1838–1922), and James Joseph Sylvester (Britain, 1814–1897) came to them through what we now call linear algebra, while Erhard Schmidt (Germany, 1876–1959) and Hermann Klaus Hugo Weyl (Germany, 1885–1955) approached them from integral equations. Although none of them used the matrix terminology or notation that is familiar to modern workers, seeing the structure within sets of equations was a more familiar task to them than it was to us. The terminology "singular values" appears to come from the literature on integral equations. The use of SVDs as workhorses in modern numerical analysis owes most to Gene Howard Golub (United States, 1932–2007).

References

Stewart, G. W. 1993. On the early history of the singular value decomposition. *SIAM Review* 35: 551–566. https://doi.org/10.1137/1035134.

Trefethen, L. N. 2007. Obituary: Gene H. Golub (1932–2007). *Nature* 450: 962. https://doi.org/10.1038/450962a.

Also see

[M-5] **fullsvd( )** — Full singular value decomposition

[M-5] **norm( )** — Matrix and vector norms

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

[M-5] **rank( )** — Rank of matrix

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

[M-4] **Matrix** — Matrix functions