Title

stata.com

rotate — Orthogonal and oblique rotations after factor and pca

Description

`rotate` performs a rotation of the loading matrix after `factor`, `factormat`, `pca`, or `pcamat`; see [MV] **factor** and [MV] **pca**. Many rotation criteria (such as varimax and oblimin) are available that can be applied with respect to the orthogonal and oblique class of rotations.

`rotate, clear` removes the rotation results from the estimation results.

If you want to rotate a given matrix, see [MV] **rotatemat**.

If you want a Procrustes rotation, which rotates variables optimally toward other variables, see [MV] **procrustes**.

Quick start

Orthogonal varimax rotation of loading matrix after `pca` or `factor`

```
rotate
```

As above, but apply the minimum entropy rotation criterion

```
rotate, entropy
```

As above, but apply oblique quartimin rotation criterion

```
rotate, oblique quartimin
```

As above, but rotate the Kaiser normalized matrix

```
rotate, oblique quartimin normalize
```

Menu

Statistics > Multivariate analysis > Factor and principal component analysis > Postestimation > Rotate loadings

Syntax

```
rotate [ , options ]

rotate, clear
```

options	Description
Main	
orthogonal	restrict to orthogonal rotations; the default, except with promax()
oblique	allow oblique rotations
rotation_methods	rotation criterion
normalize	rotate Kaiser normalized matrix
factors(#)	rotate # factors or components; default is to rotate all
components(#)	synonym for factors()
Reporting	
blanks(#)	display loadings as blanks when |loading| $<$ #; default is blanks(0)
detail	show rotatemat output; seldom used
format(%*fmt*)	display format for matrices; default is format(%9.5f)
noloading	suppress display of rotated loadings
norotation	suppress display of rotation matrix
Optimization	
optimize_options	control the maximization process; seldom used

rotation_methods	Description
* varimax	varimax (orthogonal only); the default
vgpf	varimax via the GPF algorithm (orthogonal only)
quartimax	quartimax (orthogonal only)
equamax	equamax (orthogonal only)
parsimax	parsimax (orthogonal only)
entropy	minimum entropy (orthogonal only)
tandem1	Comrey's tandem 1 principle (orthogonal only)
tandem2	Comrey's tandem 2 principle (orthogonal only)
* promax[(#)]	promax power # (implies oblique); default is promax(3)
oblimin[(#)]	oblimin with $\gamma = \#$; default is oblimin(0)
cf(#)	Crawford–Ferguson family with $\kappa = \#$, $0 \leq \# \leq 1$
bentler	Bentler's invariant pattern simplicity
oblimax	oblimax
quartimin	quartimin
target(*Tg*)	rotate toward matrix *Tg*
partial(*Tg W*)	rotate toward matrix *Tg*, weighted by matrix *W*

* varimax and promax ignore all *optimize_options*.

Options

Main

`orthogonal` specifies that an orthogonal rotation be applied. This is the default.

See *Rotation criteria* below for details on the *rotation_methods* available with `orthogonal`.

`oblique` specifies that an oblique rotation be applied. This often yields more interpretable factors with a simpler structure than that obtained with an orthogonal rotation. In many applications (for example, after `factor` and `pca`) the factors before rotation are orthogonal (uncorrelated), whereas the oblique rotated factors are correlated.

See *Rotation criteria* below for details on the *rotation_methods* available with `oblique`.

`clear` specifies that rotation results be cleared (removed) from the last estimation command. `clear` may not be combined with any other option.

`rotate` stores its results within the `e()` results of `pca` and `factor`, overwriting any previous rotation results. Postestimation commands such as `predict` operate on the last rotated results, if any, instead of the unrotated results, and allow you to specify `norotated` to use the unrotated results. The `clear` option of `rotate` allows you to remove the rotation results from `e()`, thus freeing you from having to specify `norotated` for the postestimation commands.

`normalize` requests that the rotation be applied to the Kaiser normalization (Horst 1965) of the matrix $\mathbf{A}$, so that the rowwise sums of squares equal 1. Kaiser normalization applies to the rotated columns only (see the `factors()` option below).

`factors(#)`, and synonym `components(#)`, specifies the number of factors or components (columns of the loading matrix) to be rotated, counted "from the left", that is, with the lowest column index. The other columns are left unrotated. All columns are rotated by default.

Reporting

`blanks(#)` shows blanks for loadings with absolute values smaller than #.

`detail` displays the `rotatemat` output; seldom used.

`format(%`*fmt*`)` specifies the display format for matrices. The default is `format(%9.5f)`.

`noloading` suppresses the display of the rotated loadings.

`norotation` suppresses the display of the optimal rotation matrix.

Optimization

optimize_options are seldom used; see [MV] **rotatemat**.

Rotation criteria

In the descriptions below, the matrix to be rotated is denoted as $\mathbf{A}$, p denotes the number of rows of $\mathbf{A}$, and f denotes the number of columns of $\mathbf{A}$ (factors or components). If $\mathbf{A}$ is a loading matrix from `factor` or `pca`, p is the number of variables, and f is the number of factors or components.

Criteria suitable only for orthogonal rotations

`varimax` and `vgpf` apply the orthogonal varimax rotation (Kaiser 1958). `varimax` maximizes the variance of the squared loadings within factors (columns of $\mathbf{A}$). It is equivalent to `cf(`*1/p*`)` and to `oblimin(1)`. `varimax`, the most popular rotation, is implemented with a dedicated fast algorithm and ignores all *optimize_options*. Specify `vgpf` to switch to the general GPF algorithm used for the other criteria.

`quartimax` uses the quartimax criterion (Harman 1976). `quartimax` maximizes the variance of the squared loadings within the variables (rows of $\mathbf{A}$). For orthogonal rotations, `quartimax` is equivalent to `cf(0)` and to `oblimax`.

`equamax` specifies the orthogonal equamax rotation. `equamax` maximizes a weighted sum of the `varimax` and `quartimax` criteria, reflecting a concern for simple structure within variables (rows of $\mathbf{A}$) as well as within factors (columns of $\mathbf{A}$). `equamax` is equivalent to `oblimin(`p`/2)` and `cf(#)`, where $\# = f/(2p)$.

`parsimax` specifies the orthogonal parsimax rotation. `parsimax` is equivalent to `cf(#)`, where $\# = (f-1)/(p+f-2)$.

`entropy` applies the minimum entropy rotation criterion (Jennrich 2004).

`tandem1` specifies that the first principle of Comrey's tandem be applied. According to Comrey (1967), this principle should be used to judge which "small" factors should be dropped.

`tandem2` specifies that the second principle of Comrey's tandem be applied. According to Comrey (1967), `tandem2` should be used for "polishing".

Criteria suitable only for oblique rotations

`promax`[`(#)`] specifies the oblique promax rotation. The optional argument specifies the `promax` power. Not specifying the argument is equivalent to specifying `promax(3)`. Values smaller than 4 are recommended, but the choice is yours. Larger `promax` powers simplify the loadings (generate numbers closer to zero and one) but at the cost of additional correlation between factors. Choosing a value is a matter of trial and error, but most sources find values in excess of 4 undesirable in practice. The power must be greater than 1 but is not restricted to integers.

Promax rotation is an oblique rotation method that was developed before the "analytical methods" (based on criterion optimization) became computationally feasible. Promax rotation comprises an oblique Procrustean rotation of the original loadings $\mathbf{A}$ toward the elementwise #-power of the orthogonal varimax rotation of $\mathbf{A}$.

Criteria suitable for orthogonal and oblique rotations

`oblimin`[`(#)`] specifies that the oblimin criterion with $\gamma = \#$ be used. When restricted to orthogonal transformations, the `oblimin()` family is equivalent to the orthomax criterion function. Special cases of `oblimin()` include

γ	Special case
0	quartimax / quartimin
1/2	biquartimax / biquartimin
1	varimax / covarimin
$p/2$	equamax

p = number of rows of $\mathbf{A}$.

γ defaults to zero. Jennrich (1979) recommends $\gamma \leq 0$ for oblique rotations. For $\gamma > 0$, it is possible that optimal oblique rotations do not exist; the iterative procedure used to compute the solution will wander off to a degenerate solution.

`cf(#)` specifies that a criterion from the Crawford–Ferguson (1970) family be used with $\kappa = \#$. `cf(`κ`)` can be seen as $(1-\kappa)\text{cf}_1(\mathbf{A}) + (\kappa)\text{cf}_2(\mathbf{A})$, where $\text{cf}_1(\mathbf{A})$ is a measure of row parsimony and $\text{cf}_2(\mathbf{A})$ is a measure of column parsimony. $\text{cf}_1(\mathbf{A})$ attains its greatest lower bound when no row of $\mathbf{A}$ has more than one nonzero element, whereas $\text{cf}_2(\mathbf{A})$ reaches zero if no column of $\mathbf{A}$ has more than one nonzero element.

For orthogonal rotations, the Crawford–Ferguson family is equivalent to the `oblimin()` family. For orthogonal rotations, special cases include the following:

κ	Special case
0	quartimax / quartimin
$1/p$	varimax / covarimin
$f/(2p)$	equamax
$(f-1)/(p+f-2)$	parsimax
1	factor parsimony

p = number of rows of $\mathbf{A}$.
f = number of columns of $\mathbf{A}$.

`bentler` specifies that the "invariant pattern simplicity" criterion (Bentler 1977) be used.

`oblimax` specifies the oblimax criterion. `oblimax` maximizes the number of high and low loadings. `oblimax` is equivalent to `quartimax` for orthogonal rotations.

`quartimin` specifies that the quartimin criterion be used. For orthogonal rotations, `quartimin` is equivalent to `quartimax`.

`target(`*Tg*`)` specifies that $\mathbf{A}$ be rotated as near as possible to the conformable matrix *Tg*. Nearness is expressed by the Frobenius matrix norm.

`partial(`*Tg W*`)` specifies that $\mathbf{A}$ be rotated as near as possible to the conformable matrix *Tg*. Nearness is expressed by a weighted (by *W*) Frobenius matrix norm. *W* should be nonnegative and usually is zero–one valued, with ones identifying the target values to be reproduced as closely as possible by the factor loadings, whereas zeros identify loadings to remain unrestricted.

Remarks and examples

stata.com

Remarks are presented under the following headings:

In this entry, we focus primarily on the rotation of factor loading matrices in factor analysis. `rotate` may also be used after `pca`, with the same syntax. We advise caution in the interpretation of rotated loadings in principal component analysis because some of the optimality properties of principal components are not preserved under rotation. See [MV] **pca postestimation** for more discussion of this point.

Orthogonal rotations

The interpretation of a factor analytical solution is not always easy—an understatement, many will agree. This is due partly to the standard way in which the inherent indeterminacy of factor analysis is resolved. Orthogonal transformations of the common factors and the associated factor loadings are possible without affecting the reconstructed (fitted) correlation matrix and preserving the property that common factors are uncorrelated. This gives considerable freedom in selecting an orthogonal rotation to facilitate the interpretation of the factor loadings. Thurstone (1935) offered criteria for a "simple structure" required for a psychologically meaningful factor solution. These informal criteria for interpretation were then formalized into formal rotation criteria, for example, Harman (1976) and Gorsuch (1983).

Example 1: Orthogonal varimax rotation

We illustrate `rotate` by using a factor analysis of the correlation matrix of eight physical variables (height, arm span, length of forearm, length of lower leg, weight, bitrochanteric diameter, chest girth, and chest width) of 305 girls.

```
. matrix input R = (1000  846  805  859  473  398  301  382 \
>                   846 1000  881  826  376  326  277  415 \
>                   805  881 1000  801  380  319  237  345 \
>                   859  826  801 1000  436  329  327  365 \
>                   473  376  380  436 1000  762  730  629 \
>                   398  326  319  329  762 1000  583  577 \
>                   301  277  237  327  730  583 1000  539 \
>                   382  415  345  365  629  577  539 1000)
. matrix R = R/1000
. matrix colnames R = height  arm_span  fore_arm  lower_leg
>                     weight  bitrod    ch_girth  ch_width
. matrix rownames R = height  arm_span  fore_arm  lower_leg
>                     weight  bitrod    ch_girth  ch_width
. matlist R, border format(%7.3f)
```

	height	arm_s~n	fore_~m	lower~g	weight	bitrod	ch_gi~h	ch_wi~h
height	1.000							
arm_span	0.846	1.000						
fore_arm	0.805	0.881	1.000					
lower_leg	0.859	0.826	0.801	1.000				
weight	0.473	0.376	0.380	0.436	1.000			
bitrod	0.398	0.326	0.319	0.329	0.762	1.000		
ch_girth	0.301	0.277	0.237	0.327	0.730	0.583	1.000	
ch_width	0.382	0.415	0.345	0.365	0.629	0.577	0.539	1.000

We extract two common factors with the iterated principal-factor method. See the description of `factormat` in [MV] **factor** for details on running a factor analysis on a Stata matrix rather than on a dataset.

```
. factormat R, n(305) fac(2) ipf
(obs=305)
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: (unrotated)                      Number of params =         15
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	4.44901	2.93878	0.7466	0.7466
Factor2	1.51023	1.40850	0.2534	1.0000
Factor3	0.10173	0.04705	0.0171	1.0171
Factor4	0.05468	0.03944	0.0092	1.0263
Factor5	0.01524	0.05228	0.0026	1.0288
Factor6	-0.03703	0.02321	-0.0062	1.0226
Factor7	-0.06025	0.01415	-0.0101	1.0125
Factor8	-0.07440	.	-0.0125	1.0000

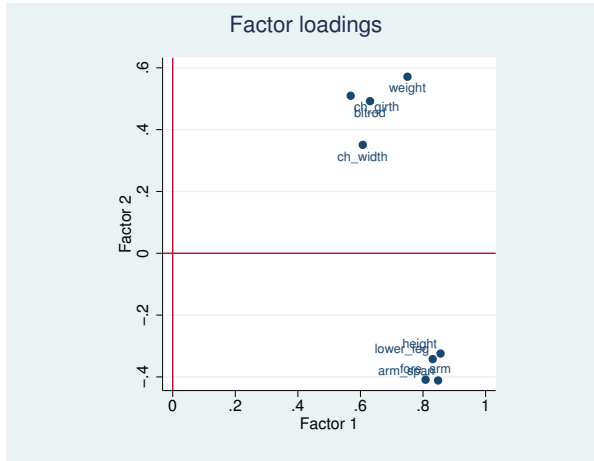
```
LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
```

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
height	0.8560	-0.3244	0.1620
arm_span	0.8482	-0.4115	0.1112
fore_arm	0.8082	-0.4090	0.1795
lower_leg	0.8309	-0.3424	0.1923
weight	0.7503	0.5712	0.1108
bitrod	0.6307	0.4922	0.3600
ch_girth	0.5687	0.5096	0.4169
ch_width	0.6074	0.3507	0.5081

The default factor solution is rather poor from the perspective of a "simple structure", namely, that variables should have high loadings on few (one) factors and factors should ideally have only low and high values. A plot of the loadings is illuminating.

```
. loadingplot, xlab(0(.2)1) ylab(-.4(.2).6) aspect(1) yline(0) xline(0)
```



There are two groups of variables. We would like to see one group of variables close to one axis and the other group of variables close to the other axis. Turning the plot by about 45 degrees counterclockwise should make this possible and offer a much "simpler" structure. This is what the `rotate` command accomplishes.

```
. rotate
Factor analysis/correlation                      Number of obs    =        305
    Method: iterated principal factors           Retained factors =          2
    Rotation: orthogonal varimax (Kaiser off)    Number of params =         15
```

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.39957	0.83989	0.5705	0.5705
Factor2	2.55968	.	0.4295	1.0000

LR test: independent vs. saturated: chi2(28) = 2092.68 Prob>chi2 = 0.0000

Rotated factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
height	0.8802	0.2514	0.1620
arm_span	0.9260	0.1770	0.1112
fore_arm	0.8924	0.1550	0.1795
lower_leg	0.8708	0.2220	0.1923
weight	0.2603	0.9064	0.1108
bitrod	0.2116	0.7715	0.3600
ch_girth	0.1515	0.7484	0.4169
ch_width	0.2774	0.6442	0.5081

Factor rotation matrix

	Factor1	Factor2
Factor1	0.8018	0.5976
Factor2	-0.5976	0.8018

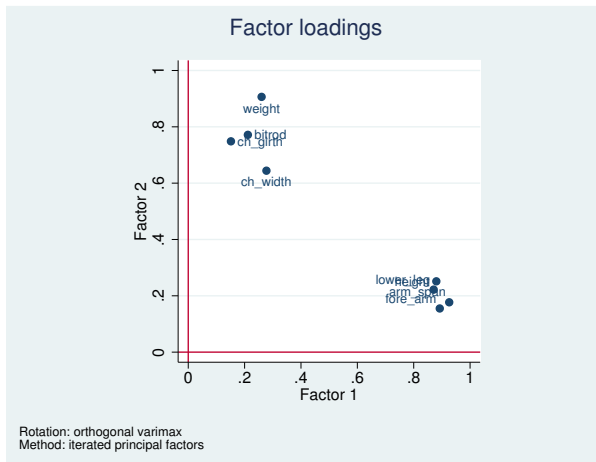
See [MV] **factor** for the interpretation of the first panel. Here we will focus on the second and third panel. The rotated factor loadings satisfy

$$\text{Factor1}_{\text{rotated}} = 0.8018 \times \text{Factor1}_{\text{unrotated}} - 0.5976 \times \text{Factor2}_{\text{unrotated}}$$

$$\text{Factor2}_{\text{rotated}} = 0.5976 \times \text{Factor1}_{\text{unrotated}} + 0.8018 \times \text{Factor2}_{\text{unrotated}}$$

The uniqueness—the variance of the specific factors—is not affected, because we are changing only the coordinates in common factor space. The purpose of rotation is to make factor loadings easier to interpret. The first factor loads high on the first four variables and low on the last four variables; for the second factor, the roles are reversed. This is really a simple structure according to Thurstone's criteria. This is clear in the plot of the factor loadings.

```
. loadingplot, xlab(0(.2)1) ylab(0(.2)1) aspect(1) yline(0) xline(0)
```



`rotate` provides several different rotations. You may make your intention clearer by typing the command as

```
. rotate, orthogonal varimax
```
(*output omitted*)

rotate defaults to orthogonal (angle and length preserving) rotations of the axes; thus, orthogonal may be omitted. The default rotation method is varimax, probably the most popular method. We warn that the varimax rotation is not appropriate if you expect a general factor contributing to all variables (see also Gorsuch 1983, chap. 9). In such a case you could, for instance, consider a quartimax rotation.

◁

▷ Example 2: Orthogonal varimax rotation with normalization

rotate has performed what is known as "raw varimax", rotating the axes to maximize the sum of the variance of the squared loadings in the columns—the variance in a column is large if it comprises small and large (in the absolute sense) values. In rotating the axes, rows with large initial loadings—that is, with high communalities—have more influence than rows with only small values. Kaiser suggested that in the computation of the optimal rotation, all rows should have the same weight. This is usually known as the Kaiser normalization and sometimes known as the Horst normalization (Horst 1965). The option normalize applies this normalization method for rotation.

```
. rotate, normalize
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: orthogonal varimax (Kaiser on)   Number of params =         15
```

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.31500	0.67075	0.5563	0.5563
Factor2	2.64425	.	0.4437	1.0000

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8724	0.2775	0.1620
arm_span	0.9203	0.2045	0.1112
fore_arm	0.8874	0.1815	0.1795
lower_leg	0.8638	0.2478	0.1923
weight	0.2332	0.9137	0.1108
bitrod	0.1885	0.7775	0.3600
ch_girth	0.1292	0.7526	0.4169
ch_width	0.2581	0.6522	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.7837	0.6212
Factor2	-0.6212	0.7837

Here the raw and normalized varimax rotated loadings are not much different.

◁

In the first example, loadingplot after rotate showed the rotated loadings, not the unrotated loadings. How can this be? Remember that Stata estimation commands store their results in e(), which we can list using ereturn list.

```
. ereturn list
scalars:
                  e(f) =  2
                  e(N) =  305
               e(df_m) =  15
               e(df_r) =  13
            e(chi2_i) =  2092.68137837692
              e(df_i) =  28
               e(p_i) =  0
             e(evsum) =  5.95922412962743
               e(r_f) =  2
macros:
    e(r_normalization) : "kaiser"
            e(r_class) : "orthogonal"
        e(r_criterion) : "varimax"
           e(r_ctitle) : "varimax"
            e(cmdline) : "factormat R, n(305) fac(2) ipf"
                e(cmd) : "factor"
       e(marginsnotok) : "_ALL"
         e(properties) : "nob noV eigen"
              e(title) : "Factor analysis"
            e(predict) : "factor_p"
          e(estat_cmd) : "factor_estat"
         e(rotate_cmd) : "factor_rotate"
           e(rngstate) : "XAA1055b80bcee95e83ca9e2d41adfb0f0806ad6e5dec1468.."
            e(factors) : "factors(2)"
             e(mtitle) : "iterated principal factors"
             e(method) : "ipf"
         e(matrixname) : "R"
matrices:
              e(r_Ev) :  1 x 2
             e(r_Phi) :  2 x 2
               e(r_T) :  2 x 2
               e(r_L) :  8 x 2
                 e(C) :  8 x 8
               e(Phi) :  2 x 2
                 e(L) :  8 x 2
               e(Psi) :  1 x 8
                e(Ev) :  1 x 8
functions:
            e(sample)
```

When you replay an estimation command, it simply knows where to look, so that it can redisplay the output. `rotate` does something that few other postestimation commands are allowed to do: it adds information to the estimation results computed by `factor` or `pca`. But to avoid confusion, it writes in `e()` fields with the prefix `r_`. For instance, the matrix `e(r_L)` contains the rotated loadings.

If you replay `factor` after `rotate`, `factor` will display the rotated results. And this is what all `factor` and `pca` postestimation commands do. For instance, if you `predict` after `rotate`, `predict` will use the rotated results. Of course, it is still possible to operate on the unrotated results. `factor, norotated` replays the unrotated results. `predict` with the `norotated` option computes the factor scores for the unrotated results.

`rotate` stores information only about the most recent rotation, overwriting any information from the previous rotation. If you need the previous results again, run `rotate` with the respective options again; you do not need to run `factor` again. It is also possible to use `estimates store` to store estimation results for different rotations, which you may later restore and replay at will. See [R] **estimates store** for details.

If you no longer need the rotation results, you may type

```
. rotate, clear
```

to clean up the rotation result and return the `factor` results back to their pristine state (as if `rotate` had never been called).

▷ Example 3: Orthogonal quartimax and orthogonal oblimin rotations

`rotate` provides many more orthogonal rotations. Previously we stated that the varimax rotation can be thought of as the rotation that maximizes the varimax criterion, namely, the variance of the squared loadings summed over the columns. A column of loadings with a high variance tends to contain a series of large values and a series of low values, achieving the simplicity aim of factor analytic interpretation. The other types of rotation simply maximize other concepts of simplicity. For instance, the `quartimax` rotation aims at rowwise simplicity—preferably, the loadings within variables fall into a grouping of a few large ones and a few small ones, using again the variance in squared loadings as the criterion to be maximized.

```
. rotate, quartimax normalize
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: orthogonal quartimax (Kaiser on) Number of params =         15

    Factor |   Variance   Difference    Proportion   Cumulative
-----------+----------------------------------------------------
   Factor1 |    3.32371     0.68818        0.5577       0.5577
   Factor2 |    2.63553           .        0.4423       1.0000

    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances

    Variable |  Factor1   Factor2  Uniqueness
-------------+--------------------------------
      height |   0.8732    0.2749      0.1620
    arm_span |   0.9210    0.2017      0.1112
    fore_arm |   0.8880    0.1788      0.1795
   lower_leg |   0.8646    0.2452      0.1923
      weight |   0.2360    0.9130      0.1108
      bitrod |   0.1909    0.7769      0.3600
    ch_girth |   0.1315    0.7522      0.4169
    ch_width |   0.2601    0.6514      0.5081

Factor rotation matrix

             |  Factor1   Factor2
-------------+--------------------
     Factor1 |   0.7855    0.6188
     Factor2 |  -0.6188    0.7855
```

Here the quartimax and the varimax rotated results are rather similar. This need not be the case—varimax focuses on simplicity within columns (factors) and quartimax within rows (variables). It is possible to compromise, rotating to strive for a weighted sum of row simplicity and column simplicity. This is known as the orthogonal oblimin criterion; in the orthogonal case, `oblimin()` is equivalent to the Crawford–Ferguson (option `cf()`) family and to the orthomax family. These are parameterized families of criteria with, for instance, the following special cases:

`oblimin(0)`	quartimax rotation
`oblimin(0.5)`	biquartimax rotation
`oblimin(1)`	varimax rotation

```
. rotate, oblimin(0.5) normalize
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: orthogonal oblimin (Kaiser on)   Number of params =         15
```

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.31854	0.67783	0.5569	0.5569
Factor2	2.64071	.	0.4431	1.0000

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8727	0.2764	0.1620
arm_span	0.9206	0.2033	0.1112
fore_arm	0.8877	0.1804	0.1795
lower_leg	0.8642	0.2468	0.1923
weight	0.2343	0.9134	0.1108
bitrod	0.1895	0.7772	0.3600
ch_girth	0.1301	0.7525	0.4169
ch_width	0.2589	0.6518	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.7844	0.6202
Factor2	-0.6202	0.7844

Because the varimax and orthomax rotation are relatively close, the factor loadings resulting from an optimal rotation of a compromise criterion are close as well.

The orthogonal quartimax rotation may be obtained in different ways, namely, directly or by the appropriate member of the `oblimin()` or `cf()` families:

```
. rotate, quartimax
(output omitted)
. rotate, oblimin(0)
(output omitted)
. rotate, cf(0)
(output omitted)
```

◁

❑ Technical note

The orthogonal varimax rotation also belongs to the oblimin and Crawford–Ferguson families.

```
. rotate, varimax
(output omitted)
. rotate, oblimin(1)
(output omitted)
. rotate, cf(0.125)
(output omitted)
```

(The $0.125 = 1/8$ above is 1 divided by the number of variables.) All three produce the orthogonal varimax rotation. (There is actually a fourth way, namely `rotate, vgpf`.) There is, however, a subtle difference in algorithms used. The varimax rotation as specified by the `varimax` option (which is also the default) is computed by the classic algorithm of cycling through rotations of two factors at a time. The other ways use the general "gradient projection" algorithm proposed by Jennrich; see [MV] **rotatemat** for more information.

❑

Oblique rotations

In addition to orthogonal rotations, oblique rotations are also available.

▷ Example 4: Oblique oblimin rotation

The rotation methods that we have discussed so far are all orthogonal: the angles between the axes are unchanged, so the rotated factors are uncorrelated.

Returning to our original factor analysis,

```
. factormat R, n(305) fac(2) ipf
```
(*output omitted*)

we examine the correlation matrix of the common factors,

```
. estat common
Correlation matrix of the common factors
```

Factors	Factor1	Factor2
Factor1	1	
Factor2	0	1

and see that they are uncorrelated.

The indeterminacy in the factor analytic model, however, allows us to consider other transformations of the common factors, namely, oblique rotations. These are rotations of the axes that preserve the norms of the rows of the loadings but not the angles between the axes or the angles between the rows. There are advantages and disadvantages of oblique rotations. See, for instance, Gorsuch (1983, chap. 9). In many substantive theories, there seems little reason to impose the restriction that the common factors be uncorrelated. The additional freedom in choosing the axes generally leads to more easily interpretable factors, sometimes to a great extent. However, although most researchers are willing to accept mildly correlated factors, they would prefer to use fewer of such factors.

`rotate` provides an extensive menu of oblique rotations; with a few exceptions, criteria suitable for orthogonal rotations are also suitable for oblique rotation. Again oblique rotation can be conceived of as maximizing some "simplicity" criterion. We illustrate with the oblimin oblique rotation.

```
. rotate, oblimin oblique normalize
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: oblique oblimin (Kaiser on)      Number of params =         15
```

Factor	Variance	Proportion	Rotated factors are correlated
Factor1	3.95010	0.6629	
Factor2	3.35832	0.5635	

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8831	0.0648	0.1620
arm_span	0.9560	-0.0288	0.1112
fore_arm	0.9262	-0.0450	0.1795
lower_leg	0.8819	0.0344	0.1923
weight	0.0047	0.9408	0.1108
bitrod	-0.0069	0.8032	0.3600
ch_girth	-0.0653	0.7923	0.4169
ch_width	0.1042	0.6462	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.9112	0.7930
Factor2	-0.4120	0.6092

The oblique rotation yields a much "simpler" structure in the Thurstone (1935) sense than that of the orthogonal rotations. This time, the common factors are moderately correlated.

```
. estat common
Correlation matrix of the oblimin(0) rotated common factors
```

Factors	Factor1	Factor2
Factor1	1	
Factor2	.4716	1

◁

❑ Technical note

The numerical maximization of a simplicity criterion with respect to the class of orthogonal or oblique rotations proceeds in a stepwise method, making small improvements from an initial guess, until no more small improvements are possible. Such a procedure is not guaranteed to converge to the global optimum but to a local optimum instead. In practice, we experience few such problems. To some extent, this is because we have a reasonable starting value using the unrotated factors or loadings. As a safeguard, Stata starts the improvement from multiple initial positions chosen at random from the classes of orthonormal and normal rotation matrices. If the maximization procedure converges to the same criterion value at each trial, we may be reasonably certain that we have found the global optimum. Let us illustrate.

```
. set seed 123
. rotate, oblimin oblique normalize protect(10)
Trial   1 : min criterion   .0181657
Trial   2 : min criterion   .0181657
Trial   3 : min criterion   .0181657
Trial   4 : min criterion   .0181657
Trial   5 : min criterion   .0181657
Trial   6 : min criterion   .0181657
Trial   7 : min criterion   .0181657
Trial   8 : min criterion   .0181657
Trial   9 : min criterion   458260.7
Trial  10 : min criterion   .0181657
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: oblique oblimin (Kaiser on)      Number of params =         15
```

Factor	Variance	Proportion	Rotated factors are correlated
Factor1	3.95010	0.6629	
Factor2	3.35832	0.5635	

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8831	0.0648	0.1620
arm_span	0.9560	-0.0288	0.1112
fore_arm	0.9262	-0.0450	0.1795
lower_leg	0.8819	0.0344	0.1923
weight	0.0047	0.9408	0.1108
bitrod	-0.0069	0.8032	0.3600
ch_girth	-0.0653	0.7923	0.4169
ch_width	0.1042	0.6462	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.9112	0.7930
Factor2	-0.4120	0.6092

Here three of the random trials converged to distinct rotations from the rest. Specifying options `log` and `trace` would demonstrate that in these cases, the initial configurations were so far off that no improvements could be found. In a real application, we would probably rerun `rotate` with more trials, say, `protect(50)`, for more reassurance.

❑

❑ Technical note

There is another but almost trivial source of nonuniqueness. All simplicity criteria supported by `rotate` and `rotatemat` are invariant with respect to permutations of the rows and of the columns. Also, the signs of rotated loadings are undefined. `rotatemat`, the computational engine of `rotate`, makes sure that all columns have a positive orientation, that is, have a positive sum. `rotate`, after `factor` and `pca`, also sorts the columns into decreasing order of explained variance.

❑

Other types of rotation

rotate supports a few rotation methods that do not fit into the scheme of "simplicity maximization". The first is known as the target rotation, which seeks to rotate the factor loading matrix to approximate as much as possible a target matrix of the same size as the factor loading matrix.

▷ Example 5: Rotation toward a target matrix

We continue with our same example. If we had expected a factor loading structure in which the first group of four variables would load especially high on the first factor and the second group of four variables on the second factor, we could have set up the following target matrix.

```
. matrix W = ( 1,0 \ 1,0 \ 1,0 \ 1,0 \ 0,1 \ 0,1 \ 0,1 \ 0,1 )
. matrix list W
W[8,2]
    c1  c2
r1   1   0
r2   1   0
r3   1   0
r4   1   0
r5   0   1
r6   0   1
r7   0   1
r8   0   1
```

It is also possible to request an orthogonal or oblique rotation toward the target $\mathbf{W}$.

```
. rotate, target(W) normalize
Factor analysis/correlation                    Number of obs    =        305
    Method: iterated principal factors         Retained factors =          2
    Rotation: orthogonal target (Kaiser on)    Number of params =         15
```

Factor	Variance	Difference	Proportion	Cumulative
Factor1	3.30616	0.65307	0.5548	0.5548
Factor2	2.65309	.	0.4452	1.0000

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8715	0.2802	0.1620
arm_span	0.9197	0.2073	0.1112
fore_arm	0.8869	0.1843	0.1795
lower_leg	0.8631	0.2505	0.1923
weight	0.2304	0.9144	0.1108
bitrod	0.1861	0.7780	0.3600
ch_girth	0.1268	0.7530	0.4169
ch_width	0.2561	0.6530	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.7817	0.6236
Factor2	-0.6236	0.7817

With this target matrix, the result is not far different from the varimax and other orthogonal rotations.

◁

▷ Example 6: Oblique promax rotation

For our last example, we return to the early days of factor analysis, the time before fast computing. Analytical methods for orthogonal rotation, such as varimax, were developed relatively early. Analogous methods for oblique rotations proved more complicated. Hendrickson and White (1964) proposed a computationally simple method to obtain an oblique rotation that comprises an oblique Procrustes rotation of the factor loadings toward a signed power of the varimax rotation of the factor loadings. The promax method has one parameter, the power to which the varimax loadings are raised. Larger promax powers simplify the factor loadings (that is, generate more zeros and ones) at the cost of more correlation between the common factors. Generally, we recommend that you keep the power in the range (1,4] and not restricted to integers. Specifying `promax` is equivalent to `promax(3)`.

```
. rotate, promax normalize
Factor analysis/correlation                       Number of obs    =        305
    Method: iterated principal factors            Retained factors =          2
    Rotation: oblique promax (Kaiser on)          Number of params =         15
```

Factor	Variance	Proportion	Rotated factors are correlated
Factor1	3.92727	0.6590	
Factor2	3.31295	0.5559	

```
    LR test: independent vs. saturated:  chi2(28) = 2092.68 Prob>chi2 = 0.0000
Rotated factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Uniqueness
height	0.8797	0.0744	0.1620
arm_span	0.9505	-0.0176	0.1112
fore_arm	0.9205	-0.0340	0.1795
lower_leg	0.8780	0.0443	0.1923
weight	0.0214	0.9332	0.1108
bitrod	0.0074	0.7966	0.3600
ch_girth	-0.0509	0.7851	0.4169
ch_width	0.1152	0.6422	0.5081

```
Factor rotation matrix
```

	Factor1	Factor2
Factor1	0.9069	0.7832
Factor2	-0.4214	0.6218

In this simple two-factor example, the promax solution is similar to the oblique oblimin solution.

◁

Stored results

`rotate` adds stored results named `e(r_name)` to the stored results that were already defined by `factor` or `pca`.

`rotate` adds to the following results:

Scalars

`e(r_f)`	number of factors/components in rotated solution
`e(r_fmin)`	rotation criterion value

Macros

`e(r_class)`	`orthogonal` or `oblique`
`e(r_criterion)`	rotation criterion
`e(r_ctitle)`	title for rotation
`e(r_normalization)`	`kaiser` or `none`

Matrices

`e(r_L)`	rotated loadings
`e(r_T)`	rotation
`e(r_Phi)`	correlations between common factors (after `factor` only)
`e(r_Ev)`	explained variance by common factors (`factor`) or rotated components (`pca`)

The factors/components in the rotated solution are in decreasing order of `e(r_Ev)`.

❑ Technical note

The rest of this section contains information of interest to programmers who want to provide `rotate` support to other estimation commands. Similar to other postestimation commands, such as `estat` and `predict`, `rotate` invokes a handler command. The name of this command is extracted from the field `e(rotate_cmd)`. The estimation command *cmd* should set this field appropriately. For instance, `pca` sets the macro `e(rotate_cmd)` to `pca_rotate`. The command `pca_rotate` implements rotation after `pca` and `pcamat`, using `rotatemat` as the computational engine. `pca_rotate` does not display output itself; it relies on `pca` to do so.

For consistent behavior for end users and programmers alike, we recommend that the estimation command *cmd*, the driver commands, and other postestimation commands adhere to the following guidelines:

Driver command

- The rotate driver command for *cmd* should be named *cmd*`_rotate`.
- *cmd*`_rotate` should be an e-class command, that is, returning in `e()`.
- Make sure that *cmd*`_rotate` is invoked after the correct estimation command (for example, `if "`‘`e(cmd)`’`" != "pca" ...`).
- Allow at least the option `detail` and any option available to `rotatemat`.
- Extract from `e()` the matrix you want to rotate; invoke `rotatemat` on the matrix; and run this command `quietly` (that is, suppress all output) unless the option `detail` was specified.
- Extract the `r()` objects returned by `rotatemat`; see *Methods and formulas* of [MV] **rotatemat** for details.
- Compute derived results needed for your estimator.
- Store in `e()` fields (macros, scalars, matrices) named `r_`*name*, adding to the existing `e()` fields.

  Store the macros returned by `rotatemat` under the same named prefixed with `r_`. In particular, the macro `e(r_criterion)` should be set to the name of the rotation criterion returned by `rotatemat` as `r(criterion)`. Other commands can check this field to find out whether rotation results are available.

We suggest that only the most recent rotation results be stored, overwriting any existing `e(r_*)` results. The programmer command `_rotate_clear` clears any existing `r_*` fields from `e()`.

- Display the rotation results by replaying *cmd*.

Estimation command cmd

- In *cmd*, define `e(rotate_cmd)` to *cmd*`_rotate`.
- *cmd* should be able to display the rotated results and should default to do so if rotated results are available. Include an option `noROTated` to display the unrotated results.
- You may use the programmer command `_rotate_text` to obtain a standard descriptive text for the rotation method.

Other postestimation commands

- Other postestimation commands after *cmd* should operate on the rotated results whenever they are appropriate and available, unless the option `noROTated` specifies otherwise.
- Mention that you operate on the unrotated results only if rotated results are available, but the user or you as the programmer decided not to use them.

❑

Methods and formulas

See *Methods and formulas* of [MV] **rotatemat**.

Henry Felix Kaiser (1927–1992) was born in Morristown, New Jersey, and educated in California, where he earned degrees at Berkeley in between periods of naval service during and after World War II. A specialist in psychological and educational statistics and measurement, Kaiser worked at the Universities of Illinois and Wisconsin before returning to Berkeley in 1968. He made several contributions to factor analysis, including varimax rotation (the subject of his PhD) and a measure for assessing sampling adequacy. Kaiser is remembered as an eccentric who spray-painted his shoes in unusual colors and listed ES (Eagle Scout) as his highest degree.

References

Akhtar-Danesh, N. 2018. qfactor: A command for Q-methodology analysis. *Stata Journal* 18: 432–446.

Bentler, P. M. 1977. Factor simplicity index and transformations. *Psychometrika* 42: 277–295.

Comrey, A. L. 1967. Tandem criteria for analytic rotation in factor analysis. *Psychometrika* 32: 277–295.

Crawford, C. B., and G. A. Ferguson. 1970. A general rotation criterion and its use in orthogonal rotation. *Psychometrika* 35: 321–332.

Gorsuch, R. L. 1983. *Factor Analysis*. 2nd ed. Hillsdale, NJ: Lawrence Erlbaum.

Harman, H. H. 1976. *Modern Factor Analysis*. 3rd ed. Chicago: University of Chicago Press.

Hendrickson, A. E., and P. O. White. 1964. Promax: A quick method for rotation to oblique simple structure. *British Journal of Statistical Psychology* 17: 65–70.

Horst, P. 1965. *Factor Analysis of Data Matrices*. New York: Holt, Rinehart & Winston.

Jennrich, R. I. 1979. Admissible values of γ in direct oblimin rotation. *Psychometrika* 44: 173–177.

——. 2004. Rotation to simple loadings using component loss functions: The orthogonal case. *Psychometrika* 69: 257–273.

Jensen, A. R., and M. Wilson. 1994. Henry Felix Kaiser (1927–1992). *American Psychologist* 49: 1085.

Kaiser, H. F. 1958. The varimax criterion for analytic rotation in factor analysis. *Psychometrika* 23: 187–200.

Mulaik, S. A. 1992. Henry Felix Kaiser 1927–1992. *Multivariate Behavioral Research* 27: 159–171.

Thurstone, L. L. 1935. *The Vectors of Mind: Multiple-Factor Analysis for the Isolation of Primary Traits*. Chicago: University of Chicago Press.

Also see *References* in [MV] **rotatemat**.

Also see

[MV] **factor** — Factor analysis

[MV] **factor postestimation** — Postestimation tools for factor and factormat

[MV] **pca** — Principal component analysis

[MV] **pca postestimation** — Postestimation tools for pca and pcamat

[MV] **procrustes** — Procrustes transformation

[MV] **rotatemat** — Orthogonal and oblique rotations of a Stata matrix