factor — Factor analysis

Description

factor and factormat perform a factor analysis of a correlation matrix. The commands produce principal factor, iterated principal factor, principal-component factor, and maximum-likelihood factor analyses. factor and factormat display the eigenvalues of the correlation matrix, the factor loadings, and the uniqueness of the variables.

factor expects data in the form of variables, allows weights, and can be run for subgroups. factormat is for use with a correlation or covariance matrix.

Quick start

Principal-factor analysis using variables v1 to v5

    factor v1 v2 v3 v4 v5

Same as above, but retain at most 3 factors

    factor v1-v5, factors(3)

Principal-component factor analysis using variables v1 to v5

    factor v1-v5, pcf

Maximum-likelihood factor analysis

    factor v1-v5, ml

Same as above, but perform 50 maximizations with different starting values

    factor v1-v5, ml protect(50)

Same as above, but set the seed for reproducibility

    factor v1-v5, ml protect(50) seed(349285)

Principal-factor analysis based on a correlation matrix cmat with a sample size of 800

    factormat cmat, n(800)

Same as above, retain only factors with eigenvalues greater than or equal to 1

    factormat cmat, n(800) mineigen(1)

Menu

factor

Statistics > Multivariate analysis > Factor and principal component analysis > Factor analysis

factormat

Statistics > Multivariate analysis > Factor and principal component analysis > Factor analysis of a correlation matrix

Syntax

Factor analysis of data

> <u>fac</u>tor *varlist* [*if*] [*in*] [*weight*] [, *method options*]

Factor analysis of a correlation matrix

> factormat *matname*, n(#) [*method options factormat_options*]

matname is a square Stata matrix or a vector containing the rowwise upper or lower triangle of the correlation or covariance matrix. If a covariance matrix is provided, it is transformed into a correlation matrix for the factor analysis.

method	Description
Model 2	
pf	principal factor; the default
pcf	principal-component factor
<u>ipf</u>	iterated principal factor
ml	maximum likelihood factor

options	Description
Model 2	
<u>factors</u>(#)	maximum number of factors to be retained
<u>mineigen</u>(#)	minimum value of eigenvalues to be retained
<u>citerate</u>(#)	communality reestimation iterations (ipf only)
Reporting	
<u>blanks</u>(#)	display loadings as blank when |loadings| < #
altdivisor	use trace of correlation matrix as the divisor for reported proportions
Maximization	
protect(#)	perform # optimizations and report the best solution (ml only)
random	use random starting values (ml only); seldom used
seed(*seed*)	random-number seed (ml with protect() or random only)
maximize_options	control the maximization process; seldom used (ml only)
<u>noro</u>tated	display unrotated solution, even if rotated results are available (replay only)

norotated does not appear in the dialog box.

factormat_options	Description
Model	
shape(<u>full</u>)	*matname* is a square symmetric matrix; the default
shape(<u>lower</u>)	*matname* is a vector with the rowwise lower triangle (with diagonal)
shape(<u>upper</u>)	*matname* is a vector with the rowwise upper triangle (with diagonal)
<u>names</u>(*namelist*)	variable names; required if *matname* is triangular
forcepsd	modifies *matname* to be positive semidefinite
* n(#)	number of observations
sds(*matname₂*)	vector with standard deviations of variables
means(*matname₃*)	vector with means of variables

* n(#) is required for factormat.

bootstrap, by, collect, jackknife, rolling, and statsby are allowed with factor; see [U] **11.1.10 Prefix commands**. However, bootstrap and jackknife results should be interpreted with caution; identification of the factor parameters involves data-dependent restrictions, possibly leading to badly biased and overdispersed estimates (Milan and Whittaker 1995).

Weights are not allowed with the bootstrap prefix; see [R] **bootstrap**.

aweights are not allowed with the jackknife prefix; see [R] **jackknife**.

aweights and fweights are allowed with factor; see [U] **11.1.6 weight**.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options for factor and factormat

    ⌐ Model 2 ⌐

pf, pcf, ipf, and ml indicate the type of estimation to be performed. The default is pf.

    pf specifies that the principal-factor method be used to analyze the correlation matrix. The factor loadings, sometimes called the factor patterns, are computed using the squared multiple correlations as estimates of the communality. pf is the default.

    pcf specifies that the principal-component factor method be used to analyze the correlation matrix. The communalities are assumed to be 1.

    ipf specifies that the iterated principal-factor method be used to analyze the correlation matrix. This reestimates the communalities iteratively.

    ml specifies the maximum-likelihood factor method, assuming multivariate normal observations. This estimation method is equivalent to Rao's canonical-factor method and maximizes the determinant of the partial correlation matrix. Hence, this solution is also meaningful as a descriptive method for nonnormal data. ml is not available for singular correlation matrices. At least three variables must be specified with method ml.

factors(#) and mineigen(#) specify the maximum number of factors to be retained. factors() specifies the number directly, and mineigen() specifies it indirectly, keeping all factors with eigenvalues greater than the indicated value. The options can be specified individually, together, or not at all.

    factors(#) sets the maximum number of factors to be retained for later use by the postestimation commands. factor always prints the full set of eigenvalues but prints the corresponding eigenvectors only for retained factors. Specifying a number larger than the number of variables in the *varlist* is equivalent to specifying the number of variables in the *varlist* and is the default.

mineigen(#) sets the minimum value of eigenvalues to be retained. The default for all methods except pcf is 5×10^{-6} (effectively zero), meaning that factors associated with negative eigenvalues will not be printed or retained. The default for pcf is 1. Many sources recommend mineigen(1), although the justification is complex and uncertain. If # is less than 5×10^{-6}, it is reset to 5×10^{-6}.

citerate(#) is used only with ipf and sets the number of iterations for reestimating the communalities. If citerate() is not specified, iterations continue until the change in the communalities is small. ipf with citerate(0) produces the same results that pf does.

> Reporting

blanks(#) specifies that factor loadings smaller than # (in absolute value) be displayed as blanks.

altdivisor specifies that reported proportions and cumulative proportions be computed using the trace of the correlation matrix, trace(e(C)), as the divisor. The default is to use the sum of all eigenvalues (even those that are negative) as the divisor.

> Maximization

protect(#) is used only with ml and requests that # optimizations with random starting values be performed along with squared multiple correlation coefficient starting values and that the best of the solutions be reported. The output also indicates whether all starting values converged to the same solution. When specified with a large number, such as protect(50), this provides reasonable assurance that the solution found is global and not just a local maximum. If trace is also specified (see [R] **Maximize**), the parameters and likelihoods of each maximization will be printed.

random is used only with ml and requests that random starting values be used. This option is rarely used and should be used only after protect() has shown the presence of multiple maximums.

seed(*seed*) is used only with ml when the random or protect() options are also specified. seed() specifies the random-number seed; see [R] **set seed**. If seed() is not specified, the random-number generator starts in whatever state it was last in.

maximize_options: iterate(#), [no]log, trace, tolerance(#), and ltolerance(#); see [R] **Maximize**. These options are seldom used.

The following option is available with factor but is not shown in the dialog box:

norotated specifies that the unrotated factor solution be displayed, even if a rotated factor solution is available. norotated is for use only with replaying results.

Options unique to factormat

> Model

shape(*shape*) specifies the shape (storage method) for the covariance or correlation matrix *matname*. The following shapes are supported:

full specifies that the correlation or covariance structure of k variables is a symmetric $k \times k$ matrix. This is the default.

lower specifies that the correlation or covariance structure of k variables is a vector with $k(k+1)/2$ elements in rowwise lower-triangular order,

$$C_{11}\ C_{21}\ C_{22}\ C_{31}\ C_{32}\ C_{33}\ \ldots\ C_{k1}\ C_{k2}\ \ldots\ C_{kk}$$

upper specifies that the correlation or covariance structure of k variables is a vector with $k(k+1)/2$ elements in rowwise upper-triangular order,

$$C_{11}\ C_{12}\ C_{13}\ \ldots\ C_{1k}\ C_{22}\ C_{23}\ \ldots C_{2k}\ \ldots\ C_{(k-1,k-1)}\ C_{(k-1,k)}\ C_{kk}$$

names(*namelist*) specifies a list of k different names to be used to document output and label estimation results and as variable names by predict. names() is required if the correlation or covariance matrix is in vectorized storage mode (that is, shape(lower) or shape(upper) is specified). By default, factormat verifies that the row and column names of *matname* and the column or row names of *matname*₂ and *matname*₃ from the sds() and means() options are in agreement. Using the names() option turns off this check.

forcepsd modifies the matrix *matname* to be positive semidefinite (psd) and so be a proper covariance matrix. If *matname* is not positive semidefinite, it will have negative eigenvalues. By setting negative eigenvalues to 0 and reconstructing, we obtain the least-squares positive-semidefinite approximation to *matname*. This approximation is a singular covariance matrix.

n(#), a required option, specifies the number of observations on which *matname* is based.

sds(*matname*₂) specifies a $k \times 1$ or $1 \times k$ matrix with the standard deviations of the variables. The row or column names should match the variable names, unless the names() option is specified. sds() may be specified only if *matname* is a correlation matrix. Specify sds() if you have variables in your dataset and want to use predict after factormat. sds() does not affect the computations of factormat but provides information so that predict does not assume that the standard deviations are one.

means(*matname*₃) specifies a $k \times 1$ or $1 \times k$ matrix with the means of the variables. The row or column names should match the variable names, unless the names() option is specified. Specify means() if you have variables in your dataset and want to use predict after factormat. means() does not affect the computations of factormat but provides information so that predict does not assume the means are zero.

Remarks and examples

Remarks are presented under the following headings:

> Introduction
> Factor analysis
> Factor analysis from a correlation matrix

Introduction

Factor analysis, in the sense of exploratory factor analysis, is a statistical technique for data reduction. It reduces the number of variables in an analysis by describing linear combinations of the variables that contain most of the information and that, we hope, admit meaningful interpretations.

Factor analysis originated with the work of Spearman (1904), and has since witnessed an explosive growth, especially in the social sciences and, interestingly, in chemometrics. For an introduction, we refer to Kim and Mueller (1978b, 1978a), van Belle, Fisher, Heagerty, and Lumley (2004, chap. 14), Hamilton (2013, chap. 11), and Afifi et al. (2020). Intermediate-level treatments include Gorsuch (1983) and Harman (1976). For mathematically more advanced discussions, see Mulaik (2010), Mardia, Kent, and Taylor (2024, chap. 10), and Fuller (1987).

Structural equation modeling provides a more general framework for performing factor analysis, including confirmatory factor analysis; see [SEM] **Intro 5**, [SEM] **Example 1**, and [SEM] **Example 3**.

Also see Kolenikov (2009) for another implementation of confirmatory factor analysis.

Factor analysis

Factor analysis finds a few common factors (say, q of them) that linearly reconstruct the p original variables

$$y_{ij} = z_{i1}b_{1j} + z_{i2}b_{2j} + \cdots + z_{iq}b_{qj} + e_{ij}$$

where y_{ij} is the value of the ith observation on the jth variable, z_{ik} is the ith observation on the kth common factor, b_{kj} is the set of linear coefficients called the factor loadings, and e_{ij} is similar to a residual but is known as the jth variable's unique factor. Everything except the left-hand-side variable is to be estimated, so the model has an infinite number of solutions. Various constraints are introduced to make the model determinate.

"Reconstruction" is typically defined in terms of prediction of the correlation matrix of the original variables, unlike principal components (see [MV] **pca**), where reconstruction means minimum residual variance summed across all equations (variables).

Once the factors and their loadings have been estimated, they are interpreted—an admittedly subjective process. Interpretation typically means examining the b_{kj}'s and assigning names to each factor. Because of the indeterminacy of the factor solution, we are not limited to examining solely the b_{kj}'s. The loadings could be rotated. Rotations come in two forms—orthogonal and oblique. If we restrict to orthogonal rotations, the rotated b_{kj}s, despite appearing different, are every bit as good as (and no better than) the original loadings. Oblique rotations are often desired but do not retain some important properties of the original solution; see example 3. Because there are an infinite number of potential rotations, different rotations could lead to different interpretations of the same data. These are not to be viewed as conflicting, but instead as two different ways of looking at the same thing. See [MV] **factor postestimation** and [MV] **rotate** for more information on rotation.

▷ Example 1: A simple factor analysis on six questions

We wish to analyze physicians' attitudes toward cost. Six questions about cost were asked of 568 physicians in the Medical Outcomes Study from Tarlov et al. (1989). We do not have the original data, so we used corr2data to create a dataset with the same correlation matrix. Factor analysis is often used to validate a combination of questions that looks meaningful at first glance. Here we wish to create a variable that summarizes the information on each physician's attitude toward cost.

Each response is coded on a five-point scale, where 1 means "agree" and 5 means "disagree":

```
. use https://www.stata-press.com/data/r19/bg2
(Physician-cost data)

. describe

Contains data from https://www.stata-press.com/data/r19/bg2.dta
 Observations:            568                   Physician-cost data
    Variables:              7                   11 Feb 2024 21:54
                                                (_dta has notes)
```

Variable name	Storage type	Display format	Value label	Variable label
clinid	int	%9.0g		Physician identifier
bg2cost1	float	%9.0g		Best health care is expensive
bg2cost2	float	%9.0g		Cost is a major consideration
bg2cost3	float	%9.0g		Determine cost of tests first
bg2cost4	float	%9.0g		Monitor likely complications only
bg2cost5	float	%9.0g		Use all means regardless of cost
bg2cost6	float	%9.0g		Prefer unnecessary tests to missing tests

```
Sorted by: clinid
```

We perform the factorization on bg2cost1, bg2cost2, ..., bg2cost6.

```
. factor bg2cost1-bg2cost6
(obs=568)
Factor analysis/correlation                  Number of obs    =        568
    Method: principal factors                Retained factors =          3
    Rotation: (unrotated)                    Number of params =         15
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	0.85389	0.31282	1.0310	1.0310
Factor2	0.54107	0.51786	0.6533	1.6844
Factor3	0.02321	0.17288	0.0280	1.7124
Factor4	-0.14967	0.03951	-0.1807	1.5317
Factor5	-0.18918	0.06197	-0.2284	1.3033
Factor6	-0.25115	.	-0.3033	1.0000

```
    LR test: independent vs. saturated:  chi2(15) =  269.07 Prob>chi2 = 0.0000
Factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Factor3	Uniqueness
bg2cost1	0.2470	0.3670	-0.0446	0.8023
bg2cost2	-0.3374	0.3321	-0.0772	0.7699
bg2cost3	-0.3764	0.3756	0.0204	0.7169
bg2cost4	-0.3221	0.1942	0.1034	0.8479
bg2cost5	0.4550	0.2479	0.0641	0.7274
bg2cost6	0.4760	0.2364	-0.0068	0.7175

`factor` retained only the first three factors because the eigenvalues associated with the remaining factors are negative. According to the default `mineigen(0)` criterion, a factor must have an eigenvalue greater than zero to be retained. You can set this threshold higher by specifying `mineigen(#)`. Although `factor` elected to retain three factors, only the first two appear to be meaningful.

The first factor seems to describe the physician's average position on cost because it affects the responses to all the questions "positively", as shown by the signs in the first column of the factor-loading table. We say "positively" because, obviously, the signs on three of the loadings are negative. When we look back at the results of describe, however, we find that the direction of the responses on bg2cost2, bg2cost3, and bg2cost4 are reversed. If the physician feels that cost should not be a major influence on medical treatment, he or she is likely to disagree with these three items and to agree with the other three.

The second factor loads positively (absolutely, not logically) on all six items and could be interpreted as describing the physician's tendency to agree with any good-sounding idea put forth. Psychologists refer to this as the "positive response set". On statistical grounds, we would probably keep this second factor, although on substantive grounds, we would be tempted to drop it.

We finally point to the column with the header "uniqueness". *Uniqueness* is the percentage of variance for the variable that is not explained by the common factors. The quantity "1 − *uniqueness*" is called *communality*. Uniqueness could be pure measurement error, or it could represent something that is measured reliably in that particular variable, but not by any of the others. The greater the uniqueness, the more likely that it is more than just measurement error. Values more than 0.6 are usually considered high; all the variables in this problem are even higher—more than 0.71. If the uniqueness is high, then the variable is not well explained by the factors.

◁

▷ Example 2: A different divisor for proportions

The cumulative proportions of the eigenvalues exceeded 1.0 in our factor analysis because of the negative eigenvalues. By default, the proportion and cumulative proportion columns are computed using the sum of all eigenvalues as the divisor. The altdivisor option allows you to display the proportions and cumulative proportions by using the trace of the correlation matrix as the divisor. This option is allowed at estimation time or when replaying results. We demonstrate by replaying the results with this option.

```
. factor, altdivisor
```

Factor analysis/correlation   Number of obs   =   568
    Method: principal factors   Retained factors =   3
    Rotation: (unrotated)   Number of params =   15

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	0.85389	0.31282	0.1423	0.1423
Factor2	0.54107	0.51786	0.0902	0.2325
Factor3	0.02321	0.17288	0.0039	0.2364
Factor4	-0.14967	0.03951	-0.0249	0.2114
Factor5	-0.18918	0.06197	-0.0315	0.1799
Factor6	-0.25115	.	-0.0419	0.1380

LR test: independent vs. saturated: chi2(15) = 269.07 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Factor3	Uniqueness
bg2cost1	0.2470	0.3670	-0.0446	0.8023
bg2cost2	-0.3374	0.3321	-0.0772	0.7699
bg2cost3	-0.3764	0.3756	0.0204	0.7169
bg2cost4	-0.3221	0.1942	0.1034	0.8479
bg2cost5	0.4550	0.2479	0.0641	0.7274
bg2cost6	0.4760	0.2364	-0.0068	0.7175

Among the sources we examined, there was not a consensus on which divisor is most appropriate. Therefore, both are available.

◁

▷ Example 3: Principal-component factors instead of principal factors

factor provides several alternative estimation strategies for the factor model. We specified no options on the factor command when we fit our first model, so we obtained the principal-factor solution. The *communalities* (defined as $1 - uniqueness$) were estimated using the squared multiple correlation coefficients.

We could have instead obtained the estimates from "principal-component factors", treating the communalities as all 1 — meaning that there are no unique factors — by specifying the pcf option:

```
. factor bg2cost1-bg2cost6, pcf
(obs=568)
```

Factor analysis/correlation
    Method: principal-component factors
    Rotation: (unrotated)

Number of obs   =   568
Retained factors =   2
Number of params =   11

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	1.70622	0.30334	0.2844	0.2844
Factor2	1.40288	0.49422	0.2338	0.5182
Factor3	0.90865	0.18567	0.1514	0.6696
Factor4	0.72298	0.05606	0.1205	0.7901
Factor5	0.66692	0.07456	0.1112	0.9013
Factor6	0.59236	.	0.0987	1.0000

LR test: independent vs. saturated: chi2(15) = 269.07 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
bg2cost1	0.3581	0.6279	0.4775
bg2cost2	-0.4850	0.5244	0.4898
bg2cost3	-0.5326	0.5725	0.3886
bg2cost4	-0.4919	0.3254	0.6521
bg2cost5	0.6238	0.3962	0.4539
bg2cost6	0.6543	0.3780	0.4290

Here we find that the principal-component factor model is inappropriate. It is based on the assumption that the uniquenesses are 0, but we find that there is considerable uniqueness—there is considerable variability left over after our two factors. We should use some other method.

◁

▷ Example 4: Iterated principal-factor analysis

We could have fit our model using iterated principal factors by specifying the ipf option. Here the initial estimates of the communalities would be the squared multiple correlation coefficients, but the solution would then be iterated to obtain different (better) estimates:

```
. factor bg2cost1-bg2cost6, ipf
(obs=568)
Factor analysis/correlation                    Number of obs    =       568
    Method: iterated principal factors         Retained factors =         5
    Rotation: (unrotated)                      Number of params =        15
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	1.08361	0.31752	0.5104	0.5104
Factor2	0.76609	0.53816	0.3608	0.8712
Factor3	0.22793	0.19469	0.1074	0.9786
Factor4	0.03324	0.02085	0.0157	0.9942
Factor5	0.01239	0.01256	0.0058	1.0001
Factor6	-0.00017	.	-0.0001	1.0000

```
    LR test: independent vs. saturated:  chi2(15) =  269.07 Prob>chi2 = 0.0000
Factor loadings (pattern matrix) and unique variances
```

Variable	Factor1	Factor2	Factor3	Factor4	Factor5	Uniqueness
bg2cost1	0.2471	0.4059	-0.1349	-0.1303	0.0288	0.7381
bg2cost2	-0.4040	0.3959	-0.2636	0.0349	0.0040	0.6093
bg2cost3	-0.4479	0.4570	0.1290	0.0137	-0.0564	0.5705
bg2cost4	-0.3327	0.1943	0.2655	0.0091	0.0810	0.7744
bg2cost5	0.5294	0.3338	0.2161	-0.0134	-0.0331	0.5604
bg2cost6	0.5174	0.2943	-0.0801	0.1208	0.0265	0.6240

Here we retained too many factors. Unlike in principal factors or principal-component factors, we cannot simply ignore the unnecessary factors because the uniquenesses are reestimated from the data and therefore depend on the number of retained factors. We need to reestimate. We use the opportunity to demonstrate the option blanks(#) for displaying "small loadings" as blanks for easier reading:

```
. factor bg2cost1-bg2cost6, ipf factors(2) blanks(.30)
(obs=568)
```

Factor analysis/correlation			Number of obs	=	568
Method: iterated principal factors			Retained factors =		2
Rotation: (unrotated)			Number of params =		11

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	1.03954	0.30810	0.5870	0.5870
Factor2	0.73144	0.60785	0.4130	1.0000
Factor3	0.12359	0.11571	0.0698	1.0698
Factor4	0.00788	0.03656	0.0045	1.0743
Factor5	-0.02867	0.07418	-0.0162	1.0581
Factor6	-0.10285	.	-0.0581	1.0000

LR test: independent vs. saturated: chi2(15) = 269.07 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
bg2cost1		0.3941	0.7937
bg2cost2	-0.3590		0.7827
bg2cost3	-0.5189	0.4935	0.4872
bg2cost4	-0.3230		0.8699
bg2cost5	0.4667	0.3286	0.6742
bg2cost6	0.5179	0.3325	0.6212

(blanks represent abs(loading)<.3)

It is instructive to compare the reported uniquenesses for this model and the previous one, where five factors were retained. Also, compared with the results we obtained from principal factors, these results do not differ much.

◁

▷ Example 5: Maximum-likelihood factor analysis

Finally, we could have fit our model using the maximum likelihood method by specifying the ml option. The maximum likelihood method assumes that the data are multivariate normal distributed. If the factor model provides an adequate approximation to the data, maximum likelihood estimates have favorable properties compared with the other estimation methods. Rao (1955) has shown that his canonical factor method is equivalent to the maximum likelihood method. This method seeks to maximize canonical correlations between the manifest variables and the common factors. Thus ml may be used descriptively, even if we are unwilling to assume multivariate normality.

As with ipf, if we do not specify the number of factors, Stata retains more than two factors (it retained three), and, as with ipf, we will need to reestimate with the number of factors that we really want. To save paper, we will start by retaining two factors:

```
. factor bg2cost1-bg2cost6, ml factors(2)
(obs=568)
Iteration 0:   Log likelihood = -28.702162
Iteration 1:   Log likelihood = -7.0065234
Iteration 2:   Log likelihood = -6.8513798
Iteration 3:   Log likelihood = -6.8429502
Iteration 4:   Log likelihood = -6.8424747
Iteration 5:   Log likelihood = -6.8424491
Iteration 6:   Log likelihood = -6.8424477
```

```
Factor analysis/correlation                Number of obs    =       568
    Method: maximum likelihood             Retained factors =         2
    Rotation: (unrotated)                  Number of params =        11
                                           Schwarz's BIC    =   83.4482
    Log likelihood = -6.842448             (Akaike's) AIC   =   35.6849
```

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	1.02766	0.28115	0.5792	0.5792
Factor2	0.74651	.	0.4208	1.0000

```
LR test: independent vs. saturated:  chi2(15) =  269.07 Prob>chi2 = 0.0000
LR test:   2 factors vs. saturated:  chi2(4)  =   13.58 Prob>chi2 = 0.0087
```

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Factor2	Uniqueness
bg2cost1	-0.1371	0.4235	0.8018
bg2cost2	0.4140	0.1994	0.7888
bg2cost3	0.6199	0.3692	0.4794
bg2cost4	0.3577	0.0909	0.8638
bg2cost5	-0.3752	0.4355	0.6695
bg2cost6	-0.4295	0.4395	0.6224

`factor` displays a likelihood-ratio test of independence against the saturated model with each estimation method. Because we are factor analyzing a correlation matrix, independence implies sphericity. Passing this test is necessary for a factor analysis to be meaningful.

In addition to the "standard" output, when you use the `ml` option, Stata reports a likelihood-ratio test of the number of factors in the model against the saturated model. This test is only approximately χ^2, and we have used the correction recommended by Bartlett (1951). There are many variations on this test in use by different statistical packages.

The following comments were made by the analyst looking at these results: "There is, in my opinion, weak evidence of more than two factors. The χ^2 test for more than two factors is really a test of how well you are fitting the correlation matrix. It is not surprising that the model does not fit it perfectly. The significance of 1%, however, suggests to me that there might be a third factor. As for the loadings, they yield a similar interpretation to other factor models we fit, although there are some noteworthy differences." When we challenged the analyst on this last statement, he added that he would want to rotate the resulting factors before committing himself further.

◁

❏ Technical note

Stata will sometimes comment, "Note: test formally not valid because a Heywood case was encountered". The approximations used in computing the χ^2 value and degrees of freedom are mathematically justified on the assumption that an *interior* solution to the factor maximum likelihood was found. This is the case in our example above, but that will not always be so.

Boundary solutions, called Heywood solutions, often produce uniquenesses of 0, and then at least at a formal level, the test cannot be justified. Nevertheless, we believe that the reported tests are useful, even in such circumstances, provided that they are interpreted cautiously. The maximum likelihood method seems to be particularly prone to producing Heywood solutions.

This message is also printed when, in principle, there are enough free parameters to completely fit the correlation matrix, another sort of boundary solution. We say "in principle" because the correlation matrix often cannot be fit perfectly, so you will see a positive χ^2 with zero degrees of freedom. This warning note is printed because the geometric assumptions underlying the likelihood-ratio test are violated.

❏

❏ Technical note

In a factor analysis with factors estimated with the maximum likelihood method, there may possibly be more than one local maximum, and you may want assurances that the maximum reported is the global maximum. Multiple maximums are especially likely when there is more than one group of variables, the groups are reasonably uncorrelated, and you attempt to fit a model with too few factors.

When you specify the protect(#) option, Stata performs # optimizations of the likelihood function, beginning each with random starting values, before continuing with the squared multiple correlations–initialized solution. Stata then selects the maximum of the maximums and reports it, along with a note informing you if other local maximums were found. protect(50) provides considerable assurance.

If you then wish to explore any of the nonglobal maximums, include the random option. This option, which is never specified with protect(), uses random starting values and reports the solution to which those random values converge. For multiple maximums, giving the command repeatedly will eventually report all local maximums. You are advised to set the random-number seed to ensure that your results are reproducible; see [R] **set seed**.

❏

Factor analysis from a correlation matrix

You may want to perform a factor analysis directly from a correlation matrix rather than from variables in a dataset. You may not have access to the dataset, or you may have used another method of estimating a correlation matrix—for example, as a matrix of tetrachoric correlations; see [R] **tetrachoric**. You can provide either a correlation or a covariance matrix—factormat will translate a covariance matrix into a correlation matrix.

▷ Example 6: Factor analysis of a correlation matrix

We illustrate with a small example with three variables on respondent's senses (visual, hearing, and taste), with a correlation matrix.

```
. matrix C = (1.000, 0.943,  0.771 \
>             0.943, 1.000,  0.605 \
>             0.771, 0.605,  1.000)
```

Elements within a row are separated by a comma, whereas rows are separated by a backslash, \. We now use `factormat` to analyze C. There are two required options here. First, the option `n(979)` specifies that the sample size is 979. Second, `factormat` has to have labels for the variables. It is possible to define row and column names for C. We did not explicitly set the names of C, so Stata has generated default row and columns names—r1 r2 r3 for the rows, and c1 c2 c3 for the columns. This will confuse `factormat`: why does a symmetric correlation matrix have different names for the rows and for the columns? `factormat` would complain about the problem and stop. We could set the row and column names of C to be the same and invoke `factormat` again. We can also specify the `names()` option with the variable names to be used.

```
. factormat C, n(979) names(visual hearing taste) fac(1) ipf
(obs=979)
```

Factor analysis/correlation	Number of obs =	979
Method: iterated principal factors	Retained factors =	1
Rotation: (unrotated)	Number of params =	3

Warning: Solution is a Heywood case; that is, invalid or boundary values of
         uniqueness.

Factor	Eigenvalue	Difference	Proportion	Cumulative
Factor1	2.43622	2.43609	1.0000	1.0000
Factor2	0.00013	0.00028	0.0001	1.0001
Factor3	-0.00015	.	-0.0001	1.0000

LR test: independent vs. saturated: chi2(3) = 3425.87 Prob>chi2 = 0.0000

Factor loadings (pattern matrix) and unique variances

Variable	Factor1	Uniqueness
visual	1.0961	-0.2014
hearing	0.8603	0.2599
taste	0.7034	0.5053

If we have the correlation matrix already in electronic form, this is a fine method. But if we have to enter a correlation matrix by hand, we may rather want to exploit its symmetry to enter just the upper triangle or lower triangle. This is not an issue with our small three-variable example, but what about a correlation matrix of 25 variables? However, there is an advantage to entering the correlation matrix in full symmetric form: redundancy offers some protection against making data entry errors; `factormat` will complain if the matrix is not symmetric.

`factormat` allows us to enter just one of the triangles of the correlation matrix as a vector, that is, a matrix with one row or column. We enter the upper triangle, including the diagonal,

```
. matrix Cup = (1.000, 0.943, 0.771,
>                      1.000, 0.605,
>                             1.000)
```

All elements are separated by a comma; indentation and the use of three lines are done for readability. We could have typed, all the numbers "in a row".

```
. matrix Cup = (1.000, 0.943, 0.771, 1.000, 0.605, 1.000)
```

We have to specify the option `shape(upper)` to inform `factormat` that the elements in the vector Cup are the upper triangle in rowwise order.

```
. factormat Cup, n(979) shape(upper) fac(2) names(visual hearing taste)
  (output omitted)
```

If we had entered the lower triangle of C, a vector Clow, it would have been defined as

```
. matrix Clow = (1.000, 0.943, 1.000, 0.771, 0.605, 1.000)
```

The features of `factormat` and `factor` are the same for estimation. Postestimation facilities are also the same—except that `predict` will not work after `factormat`, unless variables corresponding to the `names()` option exist in the dataset; see [MV] **factor postestimation**.

◁

Stored results

`factor` and `factormat` store the following in e():

Scalars

e(N)	number of observations
e(f)	number of retained factors
e(evsum)	sum of all eigenvalues
e(df_m)	model degrees of freedom
e(df_r)	residual degrees of freedom
e(chi2_i)	likelihood-ratio test of "independence vs. saturated"
e(df_i)	degrees of freedom of test of "independence vs. saturated"
e(p_i)	p-value of "independence vs. saturated"
e(ll_0)	log likelihood of null model (ml only)
e(ll)	log likelihood (ml only)
e(aic)	Akaike's AIC (ml only)
e(bic)	Schwarz's BIC (ml only)
e(chi2_1)	likelihood-ratio test of "# factors vs. saturated" (ml only)
e(df_1)	degrees of freedom of test of "# factors vs. saturated" (ml only)

Macros

e(cmd)	factor
e(cmdline)	command as typed
e(method)	pf, pcf, ipf, or ml
e(wtype)	weight type (factor only)
e(wexp)	weight expression (factor only)
e(title)	Factor analysis
e(mtitle)	description of method (for example, principal factors)
e(heywood)	Heywood case (when encountered)
e(matrixname)	input matrix (factormat only)
e(mineigen)	specified mineigen() option
e(factors)	specified factors() option
e(rngstate)	random-number state used (seed() option only)
e(properties)	nob noV eigen
e(rotate_cmd)	factor_rotate
e(estat_cmd)	factor_estat
e(predict)	factor_p
e(marginsnotok)	predictions disallowed by margins

Matrices

e(sds)	standard deviations of analyzed variables
e(means)	means of analyzed variables
e(C)	analyzed correlation matrix
e(Phi)	variance matrix common factors

e(L)	factor loadings
e(Psi)	uniqueness (variance of specific factors)
e(Ev)	eigenvalues

Functions
    e(sample)   marks estimation sample (factor only)

rotate after factor and factormat stores items in e() along with the estimation command. See *Stored results* of [MV] **factor postestimation** and [MV] **rotate** for details.

Methods and formulas

This section describes the statistical factor model. Suppose that there are p variables and q factors. Let $\mathbf{\Psi}$ represent the $p \times p$ diagonal matrix of uniquenesses, and let $\mathbf{\Lambda}$ represent the $p \times q$ factor loading matrix. Let $\mathbf{f}$ be a $1 \times q$ matrix of factors. The standardized (mean 0, variance 1) vector of observed variables $\mathbf{x}$ $(1 \times p)$ is given by the system of regression equations

$$\mathbf{x} = \mathbf{f}\mathbf{\Lambda}' + \mathbf{e}$$

where $\mathbf{e}$ is a $1 \times p$ vector of errors with diagonal covariance equal to the uniqueness matrix $\mathbf{\Psi}$. The common factors $\mathbf{f}$ and the specific factors $\mathbf{e}$ are assumed to be uncorrelated.

Under the factor model, the correlation matrix of $\mathbf{x}$, called $\mathbf{\Sigma}$, is decomposed by factor analysis as

$$\mathbf{\Sigma} = \mathbf{\Lambda}\mathbf{\Phi}\mathbf{\Lambda}' + \mathbf{\Psi}$$

There is an obvious freedom in reexpressing a given decomposition of $\mathbf{\Sigma}$. The default and unrotated form assumes uncorrelated common factors, $\mathbf{\Phi} = \mathbf{I}$. Stata performs this decomposition by an eigenvector calculation. First, an estimate is found for the uniqueness $\mathbf{\Psi}$, and then the columns of $\mathbf{\Lambda}$ are computed as the q leading eigenvectors, scaled by the square root of the appropriate eigenvalue.

See Harman (1976); Mardia, Kent, and Taylor (2024); Rencher (1998, chap. 10); and Rencher and Christensen (2012, chap. 13) for discussions of estimation methods in factor analysis. Basilevsky (1994) places factor analysis in a wider statistical context and details many interesting examples and links to other methods. For details about maximum likelihood estimation, see also Lawley and Maxwell (1971) and Clarke (1970).

References

Afifi, A. A., S. May, R. A. Donatello, and V. A. Clark. 2020. *Practical Multivariate Analysis*. 6th ed. Boca Raton, FL: CRC Press.

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

Baldwin, S. 2019. *Psychological Statistics and Psychometrics Using Stata*. College Station, TX: Stata Press.

Bartlett, M. S. 1951. The effect of standardization on a χ^2 approximation in factor analysis. *Biometrika* 38: 337–344. https://doi.org/10.2307/2332580.

Basilevsky, A. T. 1994. *Statistical Factor Analysis and Related Methods: Theory and Applications*. New York: Wiley. https://doi.org/10.1002/9780470316894.

Clarke, M. R. B. 1970. A rapidly convergent method for maximum-likelihood factor analysis. *British Journal of Mathematical and Statistical Psychology* 23: 43–52. https://doi.org/10.1111/j.2044-8317.1970.tb00434.x.

Dinno, A. 2009. Implementing Horn's parallel analysis for principal component analysis and factor analysis. *Stata Journal* 9: 291–298.

Fuller, W. A. 1987. *Measurement Error Models*. New York: Wiley. https://doi.org/10.1002/9780470316665.

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

Hamilton, L. C. 2013. *Statistics with Stata: Updated for Version 12*. 8th ed. Boston: Brooks/Cole.

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

Kim, J.-O., and C. W. Mueller. 1978a. *Factor Analysis: Statistical Methods and Practical Issues*. Thousand Oaks, CA: Sage. https://doi.org/10.4135/9781412984256.

———. 1978b. *Introduction to Factor Analysis. What It Is and How to Do It*. Thousand Oaks, CA: Sage. https://doi.org/10.4135/9781412984652.

Kolenikov, S. 2009. Confirmatory factor analysis using confa. *Stata Journal* 9: 329–373.

Lawley, D. N., and A. E. Maxwell. 1971. *Factor Analysis as a Statistical Method*. 2nd ed. London: Butterworths.

Mardia, K. V., J. T. Kent, and C. C. Taylor. 2024. *Multivariate Analysis*. 2nd ed. Hoboken, NJ: Wiley.

Milan, L., and J. C. Whittaker. 1995. Application of the parametric bootstrap to models that incorporate a singular value decomposition. *Journal of the Royal Statistical Society*, C ser., 44: 31–49. https://doi.org/10.2307/2986193.

Mulaik, S. A. 2010. *Foundations of Factor Analysis*. 2nd ed. Boca Raton, FL: Chapman and Hall/CRC. https://doi.org/10.1201/b15851.

Rao, C. R. 1955. Estimation and tests of significance in factor analysis. *Psychometrika* 20: 93–111. https://doi.org/10.1007/BF02288983.

Rencher, A. C. 1998. *Multivariate Statistical Inference and Applications*. New York: Wiley.

Rencher, A. C., and W. F. Christensen. 2012. *Methods of Multivariate Analysis*. 3rd ed. Hoboken, NJ: Wiley. https://doi.org/10.1002/9781118391686.

Spearman, C. E. 1904. The proof and measurement of association between two things. *American Journal of Psychology* 15: 72–101. https://doi.org/10.2307/1412159.

Tarlov, A. R., J. E. Ware, Jr., S. Greenfield, E. C. Nelson, E. Perrin, and M. Zubkoff. 1989. The medical outcomes study. An application of methods for monitoring the results of medical care. *Journal of the American Medical Association* 262: 925–930. https://doi.org/10.1001/jama.1989.03430070073033.

van Belle, G., L. D. Fisher, P. J. Heagerty, and T. S. Lumley. 2004. *Biostatistics: A Methodology for the Health Sciences*. 2nd ed. New York: Wiley.

Also see

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

[MV] **alpha** — Compute interitem correlations (covariances) and Cronbach's alpha

[MV] **canon** — Canonical correlations

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

[SEM] **Intro 5** — Tour of models

[SEM] **Example 1** — Single-factor measurement model

[SEM] **Example 3** — Two-factor measurement model

[U] **20 Estimation and postestimation commands**