dfactor — Dynamic-factor models

Description

`dfactor` estimates the parameters of dynamic-factor models by maximum likelihood. Dynamic-factor models are flexible models for multivariate time series in which unobserved factors have a vector autoregressive structure, exogenous covariates are permitted in both the equations for the latent factors and the equations for observable dependent variables, and the disturbances in the equations for the dependent variables may be autocorrelated.

Quick start

Dynamic-factor model with `y1` and `y2` a function of `x` and an unobserved factor that follows a third-order autoregressive process using `tsset` data

```
dfactor (y1 y2=x) (f=, ar(1/3))
```

Same as above, but with equations for the observed variables following an autoregressive process of order 1

```
dfactor (y1 y2=x, ar(1)) (f=, ar(1/3))
```

Same as above, but with an unstructured covariance matrix for the errors of `y1` and `y2`

```
dfactor (y1 y2=x, ar(1) covstructure(unstructured)) (f=, ar(1/3))
```

Menu

Statistics > Multivariate time series > Dynamic-factor models

Syntax

`dfactor` *obs_eq* [*fac_eq*] [*if*] [*in*] [, *options*]

obs_eq specifies the equation for the observed dependent variables, and it has the form

(*depvars* = [*exog_d*] [, *sopts*])

fac_eq specifies the equation for the unobserved factors, and it has the form

(*facvars* = [*exog_f*] [, *sopts*])

depvars are the observed dependent variables. *exog_d* are the exogenous variables that enter into the equations for the observed dependent variables. (All factors are automatically entered into the equations for the observed dependent variables.) *facvars* are the names for the unobserved factors in the model. You may specify the names of existing variables in *facvars*, but `dfactor` treats them only as names and takes no notice that they are also variables. *exog_f* are the exogenous variables that enter into the equations for the factors.

options	Description
Model	
`constraints(`*constraints*`)`	apply specified linear constraints
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `oim` or `robust`
Reporting	
`level(`#`)`	set confidence level; default is `level(95)`
`nocnsreport`	do not display constraints
display_options	control columns and column formats, row spacing, display of omitted variables and base and empty cells, and factor-variable labeling
Maximization	
maximize_options	control the maximization process; seldom used
`from(`*matname*`)`	specify initial values for the maximization process; seldom used
Advanced	
`method(`*method*`)`	specify the method for calculating the log likelihood; seldom used
`coeflegend`	display legend instead of statistics

sopts	Description
Model	
noconstant	suppress constant term from the equation; allowed only in *obs_eq*
ar(*numlist*)	autoregressive terms
arstructure(*arstructure*)	structure of autoregressive coefficient matrices
covstructure(*covstructure*)	covariance structure

arstructure	Description
diagonal	diagonal matrix; the default
ltriangular	lower triangular matrix
general	general matrix

covstructure	Description
identity	identity matrix
dscalar	diagonal scalar matrix
diagonal	diagonal matrix
unstructured	symmetric, positive-definite matrix

method	Description
hybrid	use the stationary Kalman filter and the De Jong diffuse Kalman filter; the default
dejong	use the stationary De Jong method and the De Jong diffuse Kalman filter

You must tsset your data before using dfactor; see [TS] **tsset**.
exog_d and *exog_f* may contain factor variables; see **[U] 11.4.3 Factor variables**.
depvars, *exog_d*, and *exog_f* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.
by, collect, fp, rolling, and statsby are allowed; see **[U] 11.1.10 Prefix commands**.
coeflegend does not appear in the dialog box.
See **[U] 20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

constraints(*constraints*) apply linear constraints. Some specifications require linear constraints for parameter identification.

noconstant suppresses the constant term.

ar(*numlist*) specifies the vector autoregressive lag structure in the equation. By default, no lags are included in either the observable or the factor equations.

arstructure(diagonal|ltriangular|general) specifies the structure of the matrices in the vector autoregressive lag structure.

`arstructure(diagonal)` specifies the matrices to be diagonal—separate parameters for each lag, but no cross-equation autocorrelations. `arstructure(diagonal)` is the default for both the observable and the factor equations.

`arstructure(ltriangular)` specifies the matrices to be lower triangular—parameterizes a recursive, or Wold causal, structure.

`arstructure(general)` specifies the matrices to be general matrices—separate parameters for each possible autocorrelation and cross-correlation.

`covstructure(identity | dscalar | diagonal | unstructured)` specifies the covariance structure of the errors.

`covstructure(identity)` specifies a covariance matrix equal to an identity matrix, and it is the default for the errors in the factor equations.

`covstructure(dscalar)` specifies a covariance matrix equal to σ^2 times an identity matrix.

`covstructure(diagonal)` specifies a diagonal covariance matrix, and it is the default for the errors in the observable variables.

`covstructure(unstructured)` specifies a symmetric, positive-definite covariance matrix with parameters for all variances and covariances.

SE/Robust

`vce(`*vcetype*`)` specifies the estimator for the variance–covariance matrix of the estimator.

`vce(oim)`, the default, causes `dfactor` to use the observed information matrix estimator.

`vce(robust)` causes `dfactor` to use the Huber/White/sandwich estimator.

Reporting

`level(#)`; see [R] **Estimation options**.

`nocnsreport`; see [R] **Estimation options**.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, and `sformat(%`*fmt*`)`; see [R] **Estimation options**.

Maximization

maximize_options: `difficult`, `technique(`*algorithm_spec*`)`, `iterate(#)`, [`no`]`log`, `trace`, `gradient`, `showstep`, `hessian`, `showtolerance`, `tolerance(#)`, `ltolerance(#)`, `nrtolerance(#)`, and `from(`*matname*`)`; see [R] **Maximize** for all options except `from()`, and see below for information on `from()`. These options are seldom used.

`from(`*matname*`)` specifies initial values for the maximization process. `from(b0)` causes `dfactor` to begin the maximization algorithm with the values in `b0`. `b0` must be a row vector; the number of columns must equal the number of parameters in the model; and the values in `b0` must be in the same order as the parameters in `e(b)`. This option is seldom used.

Advanced

`method(`*method*`)` specifies how to compute the log likelihood. `dfactor` writes the model in state-space form and uses `sspace` to estimate the parameters; see [TS] **sspace**. `method()` offers two methods for dealing with some of the technical aspects of the state-space likelihood. This option is seldom used.

`method(hybrid)`, the default, uses the Kalman filter with model-based initial values when the model is stationary and uses the De Jong (1988, 1991) diffuse Kalman filter when the model is nonstationary.

`method(dejong)` uses the De Jong (1988) method for estimating the initial values for the Kalman filter when the model is stationary and uses the De Jong (1988, 1991) diffuse Kalman filter when the model is nonstationary.

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

`coeflegend`; see [R] **Estimation options**.

Remarks and examples

Remarks are presented under the following headings:

An introduction to dynamic-factor models

`dfactor` estimates the parameters of dynamic-factor models by maximum likelihood (ML). Dynamic-factor models represent a vector of k endogenous variables as linear functions of $n_f < k$ unobserved factors and some exogenous covariates. The unobserved factors and the disturbances in the equations for the observed variables may follow vector autoregressive structures.

Dynamic-factor models have been developed and applied in macroeconomics; see Geweke (1977), Sargent and Sims (1977), Stock and Watson (1989, 1991), and Watson and Engle (1983).

Dynamic-factor models are very flexible; in a sense, they are too flexible. Constraints must be imposed to identify the parameters of dynamic-factor and static-factor models. The parameters in the default specifications in `dfactor` are identified, but other specifications require additional restrictions. The factors are identified only up to a sign, which means that the coefficients on the unobserved factors can flip signs and still produce the same predictions and the same log likelihood. The flexibility of the model sometimes produces convergence problems.

`dfactor` is designed to handle cases in which the number of modeled endogenous variables, k, is small. The ML estimator is implemented by writing the model in state-space form and by using the Kalman filter to derive and implement the log likelihood. As k grows, the number of parameters quickly exceeds the number that can be estimated.

A dynamic-factor model has the form

$$
\begin{aligned}
\mathbf{y}_t &= \mathbf{P}\mathbf{f}_t + \mathbf{Q}\mathbf{x}_t + \mathbf{u}_t \\
\mathbf{f}_t &= \mathbf{R}\mathbf{w}_t + \mathbf{A}_1\mathbf{f}_{t-1} + \mathbf{A}_2\mathbf{f}_{t-2} + \cdots + \mathbf{A}_{t-p}\mathbf{f}_{t-p} + \boldsymbol{\nu}_t \\
\mathbf{u}_t &= \mathbf{C}_1\mathbf{u}_{t-1} + \mathbf{C}_2\mathbf{u}_{t-2} + \cdots + \mathbf{C}_{t-q}\mathbf{u}_{t-q} + \boldsymbol{\epsilon}_t
\end{aligned}
$$

where the definitions are given in the following table:

Item	Dimension	Definition
$\mathbf{y}_t$	$k \times 1$	vector of dependent variables
$\mathbf{P}$	$k \times n_f$	matrix of parameters
$\mathbf{f}_t$	$n_f \times 1$	vector of unobservable factors
$\mathbf{Q}$	$k \times n_x$	matrix of parameters
$\mathbf{x}_t$	$n_x \times 1$	vector of exogenous variables
$\mathbf{u}_t$	$k \times 1$	vector of disturbances
$\mathbf{R}$	$n_f \times n_w$	matrix of parameters
$\mathbf{w}_t$	$n_w \times 1$	vector of exogenous variables
$\mathbf{A}_i$	$n_f \times n_f$	matrix of autocorrelation parameters for $i \in \{1, 2, \ldots, p\}$
$\boldsymbol{\nu}_t$	$n_f \times 1$	vector of disturbances
$\mathbf{C}_i$	$k \times k$	matrix of autocorrelation parameters for $i \in \{1, 2, \ldots, q\}$
$\boldsymbol{\epsilon}_t$	$k \times 1$	vector of disturbances

By selecting different numbers of factors and lags, the dynamic-factor model encompasses the six models in the table below:

Dynamic factors with vector autoregressive errors	$n_f > 0$	$p > 0$	$q > 0$
Dynamic factors	$n_f > 0$	$p > 0$	$q = 0$
Static factors with vector autoregressive errors	$n_f > 0$	$p = 0$	$q > 0$
Static factors	$n_f > 0$	$p = 0$	$q = 0$
Vector autoregressive errors	$n_f = 0$	$p = 0$	$q > 0$
Seemingly unrelated regression	$n_f = 0$	$p = 0$	$q = 0$

In addition to the time-series models, `dfactor` can estimate the parameters of static-factor models and seemingly unrelated regression. `dfactor` can place equality constraints on the disturbance covariances, which `sureg` and `var` do not allow.

Some examples

▷ Example 1: Dynamic-factor model

Stock and Watson (1989, 1991) wrote a simple macroeconomic model as a dynamic-factor model, estimated the parameters by ML, and extracted an economic indicator. In this example, we estimate the parameters of a dynamic-factor model. In [TS] **dfactor postestimation**, we extend this example and extract an economic indicator for the differenced series.

We have data on an industrial-production index, `ipman`; real disposable income, `income`; an aggregate weekly hours index, `hours`; and aggregate unemployment, `unemp`. We believe that these variables are first-difference stationary. We model their first differences as linear functions of an unobserved factor that follows a second-order autoregressive process.

```
. use https://www.stata-press.com/data/r19/dfex
(St. Louis Fed (FRED) macro data)

. dfactor (D.(ipman income hours unemp) = , noconstant) (f = , ar(1/2))
searching for initial values ....................
(setting technique to bhhh)
Iteration 0:  Log likelihood = -675.19824
Iteration 1:  Log likelihood = -666.74345
  (iteration log omitted)
Refining estimates:
Iteration 0:  Log likelihood = -662.09507
Iteration 1:  Log likelihood = -662.09507

Dynamic-factor model

Sample: 1972m2 thru 2008m11                        Number of obs =    442
                                                   Wald chi2(6)  = 751.95
Log likelihood = -662.09507                        Prob > chi2   = 0.0000

------------------------------------------------------------------------------
             | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
f            |
           f |
         L1. |   .2651932   .0568663     4.66   0.000     .1537372    .3766491
         L2. |   .4820398   .0624635     7.72   0.000     .3596136     .604466
-------------+----------------------------------------------------------------
D.ipman      |
           f |   .3502249   .0287389    12.19   0.000     .2938976    .4065522
-------------+----------------------------------------------------------------
D.income     |
           f |   .0746338   .0217319     3.43   0.001     .0320401    .1172276
-------------+----------------------------------------------------------------
D.hours      |
           f |   .2177469   .0186769    11.66   0.000     .1811407     .254353
-------------+----------------------------------------------------------------
D.unemp      |
           f |  -.0676016   .0071022    -9.52   0.000    -.0815217   -.0536816
-------------+----------------------------------------------------------------
/observable  |
 var(De.ipman)|  .1383158   .0167086     8.28   0.000     .1055675    .1710641
var(De.income)|  .2773808   .0188302    14.73   0.000     .2404743    .3142873
 var(De.hours)|  .0911446   .0080847    11.27   0.000     .0752988    .1069903
 var(De.unemp)|  .0237232   .0017932    13.23   0.000     .0202086    .0272378
------------------------------------------------------------------------------
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

For a discussion of the atypical iteration log, see example 1 in [TS] **sspace**.

The header in the output describes the estimation sample, reports the log-likelihood function at the maximum, and gives the results of a Wald test against the null hypothesis that the coefficients on the independent variables, the factors, and the autoregressive components are all zero. In this example, the null hypothesis that all parameters except for the variance parameters are zero is rejected at all conventional levels.

The results in the estimation table indicate that the unobserved factor is quite persistent and that it is a significant predictor for each of the observed variables.

`dfactor` writes the dynamic-factor model as a state-space model and uses the same methods as `sspace` to estimate the parameters. Example 5 in [TS] **sspace** writes the model considered here in state-space form and uses `sspace` to estimate the parameters.

◁

❑ Technical note

The signs of the coefficients on the unobserved factors are not identified. They are not identified because we can multiply the unobserved factors and the coefficients on the unobserved factors by negative one without changing the log likelihood or any of the model predictions.

Altering either the starting values for the maximization process, the maximization `technique()` used, or the platform on which the command is run can cause the signs of the estimated coefficients on the unobserved factors to change.

Changes in the signs of the estimated coefficients on the unobserved factors do not alter the implications of the model or the model predictions.

❑

▷ Example 2: Dynamic-factor model with covariates

Here we extend the previous example by allowing the errors in the equations for the observables to be autocorrelated. This extension yields a model with constrained vector autoregressive (VAR) errors and with an unobserved autocorrelated factor.

We estimate the parameters by typing

```
. dfactor (D.(ipman income hours unemp) = , noconstant ar(1)) (f = , ar(1/2))
searching for initial values ..............
(setting technique to bhhh)
```

(iteration log omitted)

```
Dynamic-factor model

Sample: 1972m2 thru 2008m11                          Number of obs =    442
                                                     Wald chi2(10) = 990.91
Log likelihood = -610.28846                          Prob > chi2   = 0.0000

──────────────┬────────────────────────────────────────────────────────────────
              │ Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
──────────────┼────────────────────────────────────────────────────────────────
f             │
            f │
          L1. │   .4058457   .0906183     4.48   0.000     .2282371    .5834544
          L2. │   .3663499   .0849584     4.31   0.000     .1998344    .5328654
──────────────┼────────────────────────────────────────────────────────────────
De.ipman      │
      e.ipman │
          LD. │  -.2772149    .068808    -4.03   0.000    -.4120761   -.1423538
──────────────┼────────────────────────────────────────────────────────────────
De.income     │
     e.income │
          LD. │  -.2213824   .0470578    -4.70   0.000    -.3136141   -.1291508
──────────────┼────────────────────────────────────────────────────────────────
De.hours      │
      e.hours │
          LD. │  -.3969317   .0504256    -7.87   0.000     -.495764   -.2980994
──────────────┼────────────────────────────────────────────────────────────────
De.unemp      │
      e.unemp │
          LD. │  -.1736835   .0532071    -3.26   0.001    -.2779675   -.0693995
──────────────┼────────────────────────────────────────────────────────────────
D.ipman       │
            f │   .3214972    .027982    11.49   0.000     .2666535    .3763408
──────────────┼────────────────────────────────────────────────────────────────
D.income      │
            f │   .0760412   .0173844     4.37   0.000     .0419684     .110114
──────────────┼────────────────────────────────────────────────────────────────
D.hours       │
            f │   .1933165   .0172969    11.18   0.000     .1594151    .2272179
──────────────┼────────────────────────────────────────────────────────────────
D.unemp       │
            f │  -.0711994   .0066553   -10.70   0.000    -.0842435   -.0581553
──────────────┼────────────────────────────────────────────────────────────────
/observable   │
 var(De.ipman)│   .1387909   .0154558     8.98   0.000     .1084981    .1690837
var(De.income)│   .2636239   .0179043    14.72   0.000     .2285322    .2987157
 var(De.hours)│   .0822919   .0071096    11.57   0.000     .0683574    .0962265
 var(De.unemp)│   .0218056   .0016658    13.09   0.000     .0185407    .0250704
──────────────┴────────────────────────────────────────────────────────────────
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

The autoregressive (AR) terms are displayed in error notation. `e.`*varname* stands for the error in the equation for *varname*. The estimate of the *p*th AR term from *y1* on *y2* is reported as `L`*p*`e.`*y1* in equation `e.`*y2*. In the above output, the estimated first-order AR term of `D.ipman` on `D.ipman` is −0.277 and is labeled as `LDe.ipman` in equation `De.ipman`.

◁

The previous two examples illustrate how to use `dfactor` to estimate the parameters of dynamic-factor models. Although the previous example indicates that the more general dynamic-factor model with VAR errors fits the data well, we use these data to illustrate how to estimate the parameters of more restrictive models.

▷ Example 3: A VAR model with constrained error variance

In this example, we use `dfactor` to estimate the parameters of a seemingly unrelated regression with constraints on the error-covariance matrix. The model is also a constrained VAR model with constraints on the error-covariance matrix, because we include the lags of two dependent variables as exogenous variables to model the dynamic structure of the data. Previous exploratory work suggested that we should drop the lag of `D.unemp` from the model.

```
. constraint 1 [/observable]cov(De.unemp,De.income) = 0
. dfactor (D.(ipman income unemp) = LD.(ipman income), noconstant
> covstructure(unstructured)), constraints(1)
searching for initial values ...........
(setting technique to bhhh)
Iteration 0:  Log likelihood = -569.34353
Iteration 1:  Log likelihood =  -548.7669
  (iteration log omitted)
Refining estimates:
Iteration 0:  Log likelihood = -535.12973
Iteration 1:  Log likelihood = -535.12973
Dynamic-factor model
Sample: 1972m3 thru 2008m11                        Number of obs =    441
                                                   Wald chi2(6)  =  88.32
Log likelihood = -535.12973                        Prob > chi2   = 0.0000
 ( 1)  [/observable]cov(De.income,De.unemp) = 0
------------------------------------------------------------------------------
              |  Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
--------------+---------------------------------------------------------------
D.ipman       |
        ipman |
          LD. |    .206276   .0471654     4.37   0.000     .1138335    .2987185
              |
       income |
          LD. |   .1867384   .0512139     3.65   0.000      .086361    .2871158
--------------+---------------------------------------------------------------
D.income      |
        ipman |
          LD. |   .1043733   .0434048     2.40   0.016     .0193015    .1894451
              |
       income |
          LD. |  -.1957893   .0471305    -4.15   0.000    -.2881634   -.1034153
--------------+---------------------------------------------------------------
D.unemp       |
        ipman |
          LD. |  -.0865823   .0140747    -6.15   0.000    -.1141681   -.0589964
              |
       income |
          LD. |  -.0200749   .0152828    -1.31   0.189    -.0500285    .0098788
--------------+---------------------------------------------------------------
/observable   |
 var(De.ipman)|   .3243902   .0218533    14.84   0.000     .2815584    .3672219
 cov(De.ipman,|
   De.income) |   .0445794    .013696     3.25   0.001     .0177358     .071423
 cov(De.ipman,|
    De.unemp) |  -.0298076   .0047755    -6.24   0.000    -.0391674   -.0204478
var(De.income)|   .2747234   .0185008    14.85   0.000     .2384624    .3109844
cov(De.income,|
    De.unemp) |          0  (constrained)
 var(De.unemp)|   .0288866   .0019453    14.85   0.000     .0250738    .0326994
------------------------------------------------------------------------------
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

The output indicates that the model fits well, except that the lag of first-differenced income is not a significant predictor of first-differenced unemployment.

◁

❑ Technical note

The previous example shows how to use `dfactor` to estimate the parameters of a seemingly unrelated regression with constraints on the error-covariance matrix. Neither `sureg` nor `var` allows for constraints on the error-covariance matrix. Without the constraints on the error-covariance matrix and including the lag of `D.unemp`,

```
. dfactor (D.(ipman income unemp) = LD.(ipman income unemp),
> noconstant covstructure(unstructured))
  (output omitted)

. var D.(ipman income unemp), lags(1) noconstant
  (output omitted)
```

and

```
. sureg (D.ipman  LD.(ipman income unemp), noconstant)
>       (D.income LD.(ipman income unemp), noconstant)
>       (D.unemp  LD.(ipman income unemp), noconstant)
  (output omitted)
```

produce the same estimates after allowing for small numerical differences.

❑

▷ Example 4: A lower-triangular VAR model with constrained error variance

The previous example estimated the parameters of a constrained VAR model with a constraint on the error-covariance matrix. This example makes two refinements on the previous one: we use an unconditional estimator instead of a conditional estimator, and we constrain the AR parameters to have a lower triangular structure. (See the next technical note for a discussion of conditional and unconditional estimators.) The results are

```
. constraint 1 [/observable]cov(De.unemp,De.income) = 0
. dfactor (D.(ipman income unemp) = , ar(1) arstructure(ltriangular) noconstant
> covstructure(unstructured)), constraints(1)
searching for initial values ...........
(setting technique to bhhh)
Iteration 0:  Log likelihood = -543.89917
Iteration 1:  Log likelihood = -541.47792
  (iteration log omitted)
Refining estimates:
Iteration 0:  Log likelihood = -540.36159
Iteration 1:  Log likelihood = -540.36159
Dynamic-factor model
Sample: 1972m2 thru 2008m11                         Number of obs =    442
                                                    Wald chi2(6)  =  75.48
Log likelihood = -540.36159                         Prob > chi2   = 0.0000
 ( 1)  [/observable]cov(De.income,De.unemp) = 0
------------------------------------------------------------------------------
             |  Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
De.ipman     |
      e.ipman|
         LD. |   .2297308   .0473147     4.86   0.000     .1369957    .3224659
-------------+----------------------------------------------------------------
De.income    |
      e.ipman|
         LD. |   .1075441   .0433357     2.48   0.013     .0226077    .1924805
             |
     e.income|
         LD. |  -.2209485    .047116    -4.69   0.000    -.3132943   -.1286028
-------------+----------------------------------------------------------------
De.unemp     |
      e.ipman|
         LD. |  -.0975759   .0151301    -6.45   0.000    -.1272304   -.0679215
             |
     e.income|
         LD. |  -.0000467   .0147848    -0.00   0.997    -.0290244    .0289309
             |
      e.unemp|
         LD. |  -.0795348   .0482213    -1.65   0.099    -.1740469    .0149773
-------------+----------------------------------------------------------------
/observable  |
 var(De.ipman)|   .3335286   .0224282    14.87   0.000     .2895702     .377487
 cov(De.ipman,|
   De.income)|   .0457804   .0139123     3.29   0.001     .0185127    .0730481
 cov(De.ipman,|
    De.unemp)|  -.0329438   .0051423    -6.41   0.000    -.0430226    -.022865
var(De.income)|   .2743375   .0184657    14.86   0.000     .2381454    .3105296
cov(De.income,|
    De.unemp)|          0  (constrained)
 var(De.unemp)|   .0292088     .00199    14.68   0.000     .0253083    .0331092
------------------------------------------------------------------------------
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

The estimated AR term of `D.income` on `D.unemp` is essentially 0. The estimated AR term for `D.unemp` on `D.unemp` is −0.0795, and it is not significant at the 1% or 5% levels. The estimated AR term of `D.ipman` on `D.income` is 0.1075 and is significant at the 5% level but not at the 1% level.

◁

❑ Technical note

We obtained the unconditional estimator in example 4 by specifying the ar() option instead of including the lags of the endogenous variables as exogenous variables, as we did in example 3. The unconditional estimator has an additional observation and is more efficient. This change is analogous to estimating an AR coefficient by arima instead of using regress on the lagged endogenous variable. For example, to obtain the unconditional estimator in a univariate model, typing

```
. arima D.ipman, ar(1) noconstant technique(nr)
  (output omitted)
```

will produce the same estimated AR coefficient as

```
. dfactor (D.ipman, ar(1) noconstant)
  (output omitted)
```

We obtain the conditional estimator by typing either

```
. regress D.ipman LD.ipman, noconstant
  (output omitted)
```

or

```
. dfactor (D.ipman = LD.ipman, noconstant)
  (output omitted)
```

❑

▷ Example 5: A static-factor model

In this example, we fit regional unemployment data to a static-factor model. We have data on the unemployment levels for the four regions in the US census: west for the West, south for the South, ne for the Northeast, and midwest for the Midwest. We treat the variables as first-difference stationary and model the first differences of these variables. Using dfactor yields

```
. use https://www.stata-press.com/data/r19/urate
(Monthly unemployment rates in US Census regions)

. dfactor (D.(west south ne midwest) = , noconstant ) (z = )
searching for initial values ............
(setting technique to bhhh)
Iteration 0:  Log likelihood =  872.71993
Iteration 1:  Log likelihood =  873.04786
  (iteration log omitted)
Refining estimates:
Iteration 0:  Log likelihood =   873.0755
Iteration 1:  Log likelihood =   873.0755

Dynamic-factor model

Sample: 1990m2 thru 2008m12                           Number of obs =    227
                                                      Wald chi2(4)  = 342.56
Log likelihood = 873.0755                             Prob > chi2   = 0.0000

------------------------------------------------------------------------------
             | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
D.west       |
           z |   .0978324   .0065644    14.90   0.000     .0849664    .1106983
-------------+----------------------------------------------------------------
D.south      |
           z |   .0859494   .0061762    13.92   0.000     .0738442    .0980546
-------------+----------------------------------------------------------------
D.ne         |
           z |   .0918607   .0072814    12.62   0.000     .0775893     .106132
-------------+----------------------------------------------------------------
D.midwest    |
           z |   .0861102   .0074652    11.53   0.000     .0714787    .1007417
-------------+----------------------------------------------------------------
/observable  |
  var(De.west)|   .0036887   .0005834     6.32   0.000     .0025453    .0048322
 var(De.south)|   .0038902   .0005228     7.44   0.000     .0028656    .0049149
    var(De.ne)|   .0064074   .0007558     8.48   0.000     .0049261    .0078887
var(De.midw~t)|   .0074749   .0008271     9.04   0.000     .0058538     .009096
------------------------------------------------------------------------------
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

The estimates indicate that we could reasonably suppose that the unobserved factor has the same effect on the changes in unemployment in all four regions. The output below shows that we cannot reject the null hypothesis that these coefficients are the same.

```
. test [D.west]z = [D.south]z = [D.ne]z = [D.midwest]z

 ( 1)  [D.west]z - [D.south]z = 0
 ( 2)  [D.west]z - [D.ne]z = 0
 ( 3)  [D.west]z - [D.midwest]z = 0

           chi2(  3) =    3.58
         Prob > chi2 =    0.3109
```

◁

▷ Example 6: A static-factor model with constraints

In this example, we impose the constraint that the unobserved factor has the same impact on changes in unemployment in all four regions. This constraint was suggested by the results of the previous example. The previous example did not allow for any dynamics in the variables, a problem we alleviate by allowing the disturbances in the equation for each observable to follow an AR(1) process.

```
. constraint 2 [D.west]z = [D.south]z
. constraint 3 [D.west]z = [D.ne]z
. constraint 4 [D.west]z = [D.midwest]z
. dfactor (D.(west south ne midwest) = , noconstant ar(1)) (z = ),
> constraints(2/4)
searching for initial values ............
(setting technique to bhhh)
Iteration 0:  Log likelihood =  827.97004
Iteration 1:  Log likelihood =  874.74471
  (iteration log omitted)
Refining estimates:
Iteration 0:  Log likelihood =  880.97488
Iteration 1:  Log likelihood =  880.97488
Dynamic-factor model
Sample: 1990m2 thru 2008m12                         Number of obs =    227
                                                    Wald chi2(5)  = 363.34
Log likelihood = 880.97488                          Prob > chi2   = 0.0000
 ( 1)  [D.west]z - [D.south]z = 0
 ( 2)  [D.west]z - [D.ne]z = 0
 ( 3)  [D.west]z - [D.midwest]z = 0

------------------------------------------------------------------------------
             |  Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
De.west      |
      e.west |
         LD. |   .1297198   .0992663     1.31   0.191    -.0648386    .3242781
-------------+----------------------------------------------------------------
De.south     |
     e.south |
         LD. |  -.2829014   .0909205    -3.11   0.002    -.4611023   -.1047004
-------------+----------------------------------------------------------------
De.ne        |
        e.ne |
         LD. |   .2866958   .0847851     3.38   0.001       .12052    .4528715
-------------+----------------------------------------------------------------
De.midwest   |
   e.midwest |
         LD. |   .0049427   .0782188     0.06   0.950    -.1483634    .1582488
-------------+----------------------------------------------------------------
D.west       |
           z |   .0904724   .0049326    18.34   0.000     .0808047    .1001401
-------------+----------------------------------------------------------------
D.south      |
           z |   .0904724   .0049326    18.34   0.000     .0808047    .1001401
-------------+----------------------------------------------------------------
D.ne         |
           z |   .0904724   .0049326    18.34   0.000     .0808047    .1001401
-------------+----------------------------------------------------------------
D.midwest    |
           z |   .0904724   .0049326    18.34   0.000     .0808047    .1001401
-------------+----------------------------------------------------------------
/observable  |
  var(De.west)|   .0038959   .0005111     7.62   0.000     .0028941    .0048977
 var(De.south)|   .0035518   .0005097     6.97   0.000     .0025528    .0045507
    var(De.ne)|   .0058173   .0006983     8.33   0.000     .0044488    .0071859
var(De.midw~t)|   .0075444   .0008268     9.12   0.000     .0059239     .009165
------------------------------------------------------------------------------
Note: Tests of variances against zero are one sided, and the two-sided
      confidence intervals are truncated at zero.
```

The results indicate that the model might not fit well. Two of the four AR coefficients are statistically insignificant, while the two significant coefficients have opposite signs and sum to about zero. We suspect that a dynamic-factors model might fit these data better than a static-factor model with autocorrelated disturbances.

◁

Stored results

dfactor stores the following in e():

Scalars

e(N)	number of observations
e(k)	number of parameters
e(k_eq)	number of equations in e(b)
e(k_eq_model)	number of equations in overall model test
e(k_dv)	number of dependent variables
e(k_obser)	number of observation equations
e(k_factor)	number of factors specified
e(o_ar_max)	number of AR terms for the disturbances
e(f_ar_max)	number of AR terms for the factors
e(df_m)	model degrees of freedom
e(ll)	log likelihood
e(chi2)	χ^2
e(p)	*p*-value for model test
e(tmin)	minimum time in sample
e(tmax)	maximum time in sample
e(stationary)	1 if the estimated parameters indicate a stationary model, 0 otherwise
e(rank)	rank of VCE
e(ic)	number of iterations
e(rc)	return code
e(converged)	1 if converged, 0 otherwise

Macros

e(cmd)	dfactor
e(cmdline)	command as typed
e(depvar)	unoperated names of dependent variables in observation equations
e(obser_deps)	names of dependent variables in observation equations
e(covariates)	list of covariates
e(factor_deps)	names of unobserved factors in model
e(tvar)	variable denoting time within groups
e(eqnames)	names of equations
e(model)	type of dynamic-factor model specified
e(title)	title in estimation output
e(tmins)	formatted minimum time
e(tmaxs)	formatted maximum time
e(o_ar)	list of AR terms for disturbances
e(f_ar)	list of AR terms for factors
e(observ_cov)	structure of observation-error covariance matrix
e(factor_cov)	structure of factor-error covariance matrix
e(chi2type)	Wald; type of model χ^2 test
e(vce)	*vcetype* specified in vce()
e(vcetype)	title used to label Std. err.
e(opt)	type of optimization
e(method)	likelihood method
e(initial_values)	type of initial values
e(technique)	maximization technique
e(tech_steps)	iterations taken in maximization technique(s)
e(datasignature)	the checksum

`e(datasignaturevars)`	variables used in calculation of checksum
`e(properties)`	`b V`
`e(estat_cmd)`	program used to implement `estat`
`e(predict)`	program used to implement `predict`
`e(marginsnotok)`	predictions disallowed by `margins`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`
Matrices	
`e(b)`	coefficient vector
`e(Cns)`	constraints matrix
`e(ilog)`	iteration log (up to 20 iterations)
`e(gradient)`	gradient vector
`e(V)`	variance–covariance matrix of the estimators
`e(V_modelbased)`	model-based variance
Functions	
`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices	
`r(table)`	matrix containing the coefficients with their standard errors, test statistics, *p*-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

`dfactor` writes the specified model as a state-space model and uses `sspace` to estimate the parameters by maximum likelihood. See Lütkepohl (2005, 619–621) for how to write the dynamic-factor model in state-space form. See [TS] **sspace** for the technical details.

References

De Jong, P. 1988. The likelihood for a state space model. *Biometrika* 75: 165–169. https://doi.org/10.2307/2336450.

———. 1991. The diffuse Kalman filter. *Annals of Statistics* 19: 1073–1083. https://doi.org/10.1214/aos/1176348139.

Geweke, J. 1977. “The dynamic factor analysis of economic time series models”. In *Latent Variables in Socioeconomic Models*, edited by D. J. Aigner and A. S. Goldberger, 365–383. Amsterdam: North-Holland.

Lütkepohl, H. 2005. *New Introduction to Multiple Time Series Analysis*. New York: Springer.

Sargent, T. J., and C. A. Sims. 1977. “Business cycle modeling without pretending to have too much a priori economic theory”. In *New Methods in Business Cycle Research: Proceedings from a Conference*, edited by C. A. Sims, 45–109. Minneapolis: Federal Reserve Bank of Minneapolis.

Stock, J. H., and M. W. Watson. 1989. “New indexes of coincident and leading economic indicators”. In *NBER Macroeconomics Annual 1989*, edited by O. J. Blanchard and S. Fischer, vol. 4: 351–394. Cambridge, MA: MIT Press.

———. 1991. “A probability model of the coincident economic indicators”. In *Leading Economic Indicators: New Approaches and Forecasting Records*, edited by K. Lahiri and G. H. Moore, 63–89. Cambridge: Cambridge University Press.

Watson, M. W., and R. F. Engle. 1983. Alternative algorithms for the estimation of dynamic factor, MIMIC and varying coefficient regression models. *Journal of Econometrics* 23: 385–400. https://doi.org/10.1016/0304-4076(83)90066-0.

Also see

[TS] **dfactor postestimation** — Postestimation tools for dfactor

[TS] **arima** — ARIMA, ARMAX, and other dynamic regression models

[TS] **sspace** — State-space models

[TS] **tsset** — Declare data to be time-series data

[TS] **var** — Vector autoregressive models

[R] **regress** — Linear regression

[R] **sureg** — Zellner's seemingly unrelated regression

[U] 20 Estimation and postestimation commands