bayes: var — Bayesian vector autoregressive models

Description

`bayes: var` fits a Bayesian vector autoregressive (VAR) model—a multivariate time-series regression of each dependent variable on lags of itself and on lags of all the other dependent variables. `bayes: var` also fits a variant of Bayesian VAR models known as the Bayesian VARX model, which also includes exogenous variables. The command supports four classes of priors, which are specific to VAR, including the original and the conjugate Minnesota priors. See [BAYES] **bayes** and [TS] **var** for details.

Quick start

Bayesian VAR for three time series ($K = 3$) with default two lags ($p = 2$) and using the default conjugate Minnesota prior

```
bayes: var y1 y2 y3
```

Same as above, but with three lags and exogenous variable `x1` ($m = 1$)

```
bayes: var y1 y2 y3, lags(1/3) exog(x1)
```

Same as above, but with random seed for reproducibility and saving simulation results in dataset `bvarsim.dta`

```
bayes, rseed(17) saving(bvarsim): var y1 y2 y3, lags(1/3) exog(x1)
```

Customize the default conjugate Minnesota prior by changing the self-variables tightness parameter from 0.1 to 1, the lag decay from 1 to 0.5, and the exogenous variable tightness parameter from 100 to 1

```
bayes, minnconjprior(selftight(1) lagdecay(0.5) exogtight(1)): ///
  var y1 y2 y3, lags(1/3) exog(x1)
```

Report posterior summaries only for coefficients on lag 1 and lag 3 of variable `y1` in the first equation (`y1`), on lag 2 of variable `y3` in the second equation (`y2`), and on exogenous variable `x1` in the third equation (`y3`)

```
bayesstats summary {y1:L1.y1} {y1:L3.y1} {y2:L2.y3} {y3:x1}
```

Bayesian VAR for three time series with two lags using the original Minnesota prior with fixed AR error covariance

```
bayes, minnfixedcovprior: var y1 y2 y3
```

Same as above, but changing some of the default original Minnesota prior settings: self-variables tightness parameter from 0.1 to 0.5 and cross-variables tightness parameter from 0.5 to 0.1

```
bayes, minnfixedcovprior(selftight(0.5) crosstight(0.1)): var y1 y2 y3
```

Specify independent multivariate normal (MVN) prior for VAR coefficients and inverse-Wishart prior for error covariance

```
bayes, minniwishprior: var y1 y2 y3
```

Same as above, but specify a 3×1 zero mean vector for the MVN prior for self-variables first-lag coefficients (other coefficients are also set to 0 automatically) and a 3×3 identity scaling matrix for the inverse-Wishart prior for error covariance

```
matrix b0 = J(3,1,0)
matrix Omega0 = diag(J(3,1,1))
bayes, minniwishprior(mean(b0) scale(Omega0)): var y1 y2 y3
```

Specify independent MVN prior for coefficients and multivariate Jeffreys prior for error covariance

```
bayes, minnjeffprior: var y1 y2 y3
```

Same as above, but change the default MVN prior mean vector to a 21×1 zero mean vector and covariance matrix to a 21×21 identity matrix for all $21 = 3 \times (2 \times 3 + 1)$ coefficients

```
matrix b0 = J(21,1,0)
matrix S0 = I(21)
bayes, minniwishprior(mean(b0) cov(S0)): var y1 y2 y3
```

Also see *Quick start* in [BAYES] **bayes** and *Quick start* in [TS] **var**.

Menu

Statistics > Multivariate time series > Bayesian models > Vector autoregression (VAR)

Syntax

`bayes` [, *bayesopts*] : `var` *depvarlist* [*if*] [*in*] [, *options*]

options	Description
Model	
`noconstant`	suppress constant term
`lags(`*numlist*`)`	specify a list of lags for the VAR
`exog(`*varlist*`)`	specify exogenous variables
`level(#)`	set credible level; default is `level(95)`

You must `tsset` your data before using `bayes: var`; see [TS] **tsset**.

depvarlist and *varlist* may contain time-series operators; see **[U] 11.4.4 Time-series varlists**.

`bayes: var, level()` is equivalent to `bayes, clevel(): var`.

For a detailed description of *options*, see *Options* in [TS] **var**.

bayesopts	Description
Priors	
* minnconjprior[(*conjopts*)]	conjugate Minnesota prior for VAR coefficients and error covariance; the default
* minnfixedcovprior[(*fixcovopts*)]	original Minnesota prior with fixed error covariance
* minniwishprior[(*iwishopts*)]	Minnesota prior with inverse-Wishart prior for error covariance
* minnjeffprior[(*jeffopts*)]	Minnesota prior with multivariate Jeffreys prior for error covariance
dryrun	show model summary without estimation
Simulation	
nchains(#)	number of chains; default is to simulate one chain
mcmcsize(#)	MCMC sample size; default is mcmcsize(10000)
burnin(#)	burn-in period; default is burnin(2500)
thinning(#)	thinning interval; default is thinning(1)
rseed(#)	random-number seed
exclude(*paramref*)	specify model parameters to be excluded from the simulation results
Blocking	
blocksummary	display block summary
Initialization	
initial(*initspec*)	specify initial values for model parameters with a single chain
init#(*initspec*)	specify initial values for #th chain; requires nchains()
initall(*initspec*)	specify initial values for all chains; requires nchains()
nomleinitial	suppress the use of maximum likelihood estimates as starting values
initrandom	specify random initial values
initsummary	display initial values used for simulation
* noisily	display output from the estimation command during initialization
Reporting	
clevel(#)	set credible interval level; default is clevel(95)
hpd	display HPD credible intervals instead of the default equal-tailed credible intervals
eform[(*string*)]	report exponentiated coefficients and, optionally, label as *string*
batch(#)	specify length of block for batch-means calculations; default is batch(0)
saving(*filename*[, replace])	save simulation results to *filename*.dta
nomodelsummary	suppress model summary
chainsdetail	display detailed simulation summary for each chain
[no]dots	suppress dots or display dots every 100 iterations and iteration numbers every 1,000 iterations; default is nodots
dots(#[, every(#)])	display dots as simulation is performed
[no]show(*paramref*)	specify model parameters to be excluded from or included in the output
notable	suppress estimation table
noheader	suppress output header
title(*string*)	display *string* as title above the table of parameter estimates
display_options	control spacing, line width, and base and empty cells

Advanced

`search(`*search_options*`)`	control the search for feasible initial values
`corrlag(#)`	specify maximum autocorrelation lag; default varies
`corrtol(#)`	specify autocorrelation tolerance; default is `corrtol(0.01)`

* Starred options are specific to the `bayes` prefix; other options are common between `bayes` and `bayesmh`.

priorspec and *paramref* are defined in [BAYES] **bayesmh**.

paramref may contain factor variables; see **[U] 11.4.3 Factor variables**.

`collect` is allowed; see **[U] 11.1.10 Prefix commands**.

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

Model parameters are $K \times p$ outcome-specific regression coefficients for lagged outcome (dependent) variables plus a constant term unless `noconstant` is specified: {*depvar_k*:*Ldepvar_1 Ldepvar_2 … Ldepvar_K* `_cons`}, where *Ldepvar_k* denotes a list of lags for dependent variable *depvar_k* such as the default `L1.`*depvar_k* `L2.`*depvar_k*. If `exog(`*varlist*`)` is specified, regression coefficients also include $K \times m$ outcome-specific coefficients for exogenous variables: {*depvar_k*:*varlist*}. Use the `dryrun` option to see the definitions of model parameters prior to estimation.

Only one of options `minnconjprior()`, `minnfixedcovprior()`, `minniwishprior()`, or `minnjeffprior()` may be specified.

For a detailed description of *bayesopts*, see *Options* below.

conjopts	Description
`mean(...)`	mean vector for the MVN prior
`phi(`*matname*`)`	covariance product matrix Φ_0 for the MVN prior; default is diagonal autoregressive-structure matrix
`df(#)`	degrees of freedom for the inverse-Wishart prior; default is $K + 2$
`scale(`*matname*`)`	scale matrix for the inverse-Wishart prior; default is proportional to AR estimate of error covariance
minnopts	Minnesota prior options

fixcovopts	Description
`mean(...)`	mean vector for the MVN prior
minnopts	Minnesota prior options

iwishopts	Description
`mean(...)`	mean vector for the MVN prior
`cov(`*matname*`)`	covariance matrix for the MVN prior; default is diagonal autoregressive-structure matrix
`df(#)`	degrees of freedom for the inverse-Wishart prior; default is $K + 2$
`scale(`*matname*`)`	scale matrix for the inverse-Wishart prior; default is proportional to AR estimate of error covariance
minnopts	Minnesota prior options

jeffopts	Description
`mean(...)`	mean vector for the MVN prior
`cov(`*matname*`)`	covariance matrix for the MVN prior; default is diagonal autoregressive-structure matrix
minnopts	Minnesota prior options

meanopts	Description
`mean(`*matname*`)`	mean vector for the MVN prior for all $K(Kp+1+m)$ coefficients; default is to use 1s for K self-variables first-lag coefficients and 0s otherwise
`mean(`m_1`,...,`m_K`)`	mean values for the MVN prior for K self-variables first-lag coefficients; all other means are assumed to be zero

minnopts	Description
`selftight(#)`	self-variables tightness parameter for the Minnesota prior; default is `selftight(0.1)`
`crosstight(#)`	cross-variables tightness parameter for the Minnesota priors; default is `crosstight(0.5)`; not used with conjugate Minnesota prior
`lagdecay(#)`	lag decay parameter for the Minnesota prior; default is `lagdecay(1)`
`exogtight(#)`	exogenous variable tightness parameter for the Minnesota prior; default is `exogtight(100)`
`arcov`	use separate AR models to estimate error covariance
`varcov`	use VAR model to estimate error covariance

Options

`noconstant`, `lags(`*numlist*`)`, and `exog(`*varlist*`)`; see [TS] **var**.

Priors

`minnconjprior` and `minnconjprior(`*conjopts*`)` specify that a conjugate Minnesota prior be used for VAR coefficients and error covariance. `minnconjprior` is the default. The prior for VAR coefficients is MVN with mean and covariance based on the original Minnesota prior. The prior for the error covariance is an inverse-Wishart distribution. See *Conjugate Minnesota prior for VAR model with unknown error covariance* in *Methods and formulas*.

conjopts are `mean(`*meanspec*`)`, `phi(`*matname*`)`, `df(#)`, `scale(`*matname*`)`, and *minnopts*. *meanspec* is one of *matname*$_{K(Kp+1+m)}$, or *matname*$_K$, or $m_1, \ldots, m_K$.

`mean(`*matname*`)` specifies the mean vector (as a Stata matrix) of the MVN prior distribution for all $K(Kp+1+m)$ VAR coefficients. The default is to use ones for K self-variable first-lag coefficients and zeros otherwise.

`mean(`m_1`,...,`m_K` | `*matname*`)` specifies K mean values or mean vector *matname* of length K of prior means for the self-variable first-lag coefficients. The rest are set to zero.

`phi(`*matname*`)` specifies the covariance product matrix Φ_0 (as a Stata matrix) of the MVN prior distribution for the VAR coefficients. The default is the Minnesota factor covariance, a diagonal matrix that accounts for the autoregressive structure of the VAR model; see *Methods and formulas*.

`df(#)` specifies the degrees of freedom of the inverse-Wishart prior distribution for the error covariance. The default is $K+2$, and this is the minimum allowed value.

`scale(`*matname*`)` specifies the scale matrix of the inverse-Wishart prior distribution for the error covariance. The default is proportional to the diagonal matrix of K AR variance estimates, one for each VAR equation; see *Methods and formulas*.

`minnfixedcovprior` and `minnfixedcovprior(`*fixcovopts*`)` specify that the Minnesota prior with a fixed AR (or VAR if option `varcov` is specified) covariance be used for VAR coefficients. This is the original Minnesota prior for Bayesian VAR models. In this model formulation, the error covariance is considered fixed, thus decreasing the number of parameters needed to be simulated and speeding up computations. See *Original Minnesota prior with known (fixed) error covariance* in *Methods and formulas*.

fixcovopts are `mean(`*meanspec*`)` and *minnopts*.

`minniwishprior` and `minniwishprior(`*iwishopts*`)` specify that the MVN prior for VAR coefficients and an inverse-Wishart prior for the error covariance be used. The priors for VAR coefficients and error covariance are independent. The default MVN prior for coefficients uses the Minnesota prior mean vector and covariance matrix. See *MVN-inverse Wishart prior* in *Methods and formulas*.

iwishopts are `mean(`*meanspec*`)`, `cov(`*matname*`)`, `df(`#`)`, `scale(`*matname*`)`, and *minnopts*.

`cov(`*matname*`)` specifies the covariance matrix Ω_0 (as a Stata matrix) of the MVN prior distribution for the VAR coefficients. The default is a diagonal matrix that accounts for the autoregressive structure of the VAR model; see *Methods and formulas*.

`df(`#`)` specifies the degrees of freedom of the inverse-Wishart prior distribution for the error covariance. The default is $K + 2$, and this is the minimum allowed value.

`scale(`*matname*`)` specifies the scale matrix of the inverse-Wishart prior distribution for the error covariance. The default is proportional to the diagonal matrix of K AR variance estimates, one for each VAR equation; see *Methods and formulas*.

`minnjeffprior` and `minnjeffprior(`*jeffopts*`)` specify that the MVN prior for VAR coefficients and the Jeffreys prior for the error covariance be used. The priors for VAR coefficients and error covariance are independent. The default MVN prior for coefficients uses the Minnesota prior mean vector and covariance matrix. See *Multivariate normal-diffuse (normal-Jeffreys) prior* in *Methods and formulas*.

jeffopts are `mean(`*meanspec*`)`, `cov(`*matname*`)`, and *minnopts*.

`cov(`*matname*`)` specifies the covariance matrix Ω_0 (as a Stata matrix) of the MVN prior distribution for the VAR coefficients. The default is a diagonal matrix that accounts for the autoregressive structure of the VAR model; see *Methods and formulas*.

minnopts are `selftight(`#`)`, `crosstight(`#`)`, `lagdecay(`#`)`, `exogtight(`#`)`, `arcov`, and `varcov`.

`selftight(`#`)` specifies the self-variable tightness parameter, λ_1, for the Minnesota prior. The default is `selftight(0.1)`. The smaller this value, the more concentrated the prior distribution around the prior mean for self-variable lag coefficients. See *Methods and formulas*.

`crosstight(`#`)` specifies the cross-variable tightness parameter, λ_2, for the Minnesota prior. The default is `crosstight(0.5)`. The smaller this value, the more concentrated the prior distribution around the prior mean for cross-variable lag coefficients. `crosstight()` is not used with the conjugate Minnesota prior. See *Methods and formulas*.

`lagdecay(`#`)` specifies the lag-decay parameter, λ_3, for the Minnesota prior. This is a rate of lag-decay correction to the prior standard deviation of all endogenous variable lag coefficients. See *Methods and formulas*.

`exogtight(#)` specifies the exogenous variable tightness parameter, λ_4, for the Minnesota prior. This is a multiplicative factor to the prior standard deviation of exogenous variable coefficients. See *Methods and formulas*.

`arcov`, the default, specifies that the diagonal AR matrix estimate be used as an estimate of the error covariance matrix. This AR matrix has, on the diagonal, the estimates of error variances obtained from fitting a separate AR(p) model to each dependent variable. Only one of `arcov` or `varcov` may be specified.

`varcov` specifies that the VAR matrix estimate be used as an estimate of the error covariance matrix. The VAR matrix is an estimate of the error covariance obtained from fitting a VAR(p) model to the dependent variables.

`arcov` and `varcov` are used with all Minnesota priors. For the original prior with a fixed error covariance, these options specify which estimate will be used for the error covariance matrix. For other priors, these options specify which estimate will be used for the prior scale matrix of an inverse-Wishart prior for error covariance matrix.

See descriptions of other *bayesopts* in *Options* in [BAYES] **bayes**.

Remarks and examples

For a general introduction to Bayesian analysis, see [BAYES] **Intro**. For an introduction to VAR models, see [TS] **var intro**. For a general introduction to Bayesian estimation using Gibbs sampling, see [BAYES] **bayesmh**. For remarks and examples specific to the `bayes` prefix, see [BAYES] **bayes**. For details about the estimation command, see [TS] **var**.

For a simple example of the `bayes` prefix, see *Introductory example* in [BAYES] **bayes**.

Remarks are presented under the following headings:

Advantages of Bayesian VAR models
Introductory examples
US macroeconomic examples

Examples are presented under the following headings:

Default Bayesian VAR model
Bayesian VAR model with original Minnesota prior
MVN priors with unrestricted error covariances
Testing Bayesian VAR stability
Explaining the Minnesota prior
Choosing the number of lags of a VAR model
Bayesian VAR(4) model estimation
IRFs
Forecasting
One-step-ahead Bayesian predictions

Advantages of Bayesian VAR models

Since their introduction by Doan, Litterman, and Sims (1984), Bayesian VAR models have gained popularity for several reasons. As Bayesian models in general, they benefit from a unified and coherent approach of Bayesian inference; see [BAYES] **Intro**. Kadiyala and Karlsson (1997), Bańbura, Giannone, and Reichlin (2008), and Dieppe, Legrand, and van Roye (2016) describe advantages of Bayesian VARs. We summarize some below.

One of the major problems with traditional VAR models is overparameterization. The number of regression parameters in a VAR model is quadratic to the number of response variables and proportional to the number of lags. This leads to many parameters being estimated even for small models and thus to loss of degrees of freedom when maximum likelihood estimation is used. Overparameterized models also produce poor forecasts. The problem of overparameterization is exacerbated when VAR models are applied to small datasets, which is common in many economic applications.

In the Bayesian framework, VAR model parameters are considered random and are controlled by prior distributions. Prior selection, viewed as a limitation of Bayesian inference in the past, is now a powerful tool for flexible analysis and not purely a source of subjective inference. For example, it is easy to shrink higher-lag regression parameters through their priors and thus reduce the effective number of lags. One such example prior is the Minnesota prior (Litterman 1980). The Minnesota prior on regression coefficients and error covariance supports a wide range of models, from oversimplified to overparameterized ones. The Bayesian out-of-sample prediction errors, which can be obtained by simulation, provide a measure for choosing between oversimplified and overparameterized models (Litterman 1984). In cases of small or low-quality data, stronger priors based on existing expert knowledge can greatly enhance otherwise potentially unreliable VAR analysis.

VAR model specification requires choosing the number of lags. Within the Bayesian approach, we can use Bayes factors to compare models using different lags and choose the best one. We can use Bayes factors also for other decision-based inference such as selecting exogenous variables.

The availability of flexible priors, reliable lag-selection criteria, and efficient sampling algorithms capable of producing precise Bayesian estimates makes Bayesian VAR inference a useful alternative to the traditional VAR analysis.

Introductory examples

▷ Example 1: Default Bayesian VAR model

Let's revisit example 1 from [TS] **var**, which replicates a case from Lütkepohl (2005, 77–78). The example models the relationships between the first differences of the natural log of investment, `dln_inv`, of income, `dln_inc`, and of consumption, `dln_cons`, registered at each quarter of the years between 1960 and 1978 in West Germany.

```
. webuse lutkepohl2
(Quarterly SA West German macro data, Bil DM, from Lutkepohl 1993 Table E.1)
. tsset
Time variable: qtr, 1960q1 to 1982q4
        Delta: 1 quarter
```

The original VAR in example 1 considers all observations before 1979, has two lags, and is fit using the `var` command.

```
. var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)
Vector autoregression

Sample: 1960q4 thru 1978q4                      Number of obs     =         73
Log likelihood =    606.307                     AIC               =  -16.03581
FPE            =   2.18e-11                     HQIC              =  -15.77323
Det(Sigma_ml)  =   1.23e-11                     SBIC              =  -15.37691

Equation           Parms      RMSE     R-sq      chi2     P>chi2
------------------------------------------------------------------
dln_inv                7     .046148   0.1286   10.76961   0.0958
dln_inc                7     .011719   0.1142   9.410683   0.1518
dln_consump            7     .009445   0.2513   24.50031   0.0004
------------------------------------------------------------------
```

	Coefficient	Std. err.	z	P>|z|	[95% conf.	interval]
dln_inv						
dln_inv						
L1.	-.3196318	.1192898	-2.68	0.007	-.5534355	-.0858282
L2.	-.1605508	.118767	-1.35	0.176	-.39333	.0722283
dln_inc						
L1.	.1459851	.5188451	0.28	0.778	-.8709326	1.162903
L2.	.1146009	.508295	0.23	0.822	-.881639	1.110841
dln_consump						
L1.	.9612288	.6316557	1.52	0.128	-.2767936	2.199251
L2.	.9344001	.6324034	1.48	0.140	-.3050877	2.173888
_cons	-.0167221	.0163796	-1.02	0.307	-.0488257	.0153814
dln_inc						
dln_inv						
L1.	.0439309	.0302933	1.45	0.147	-.0154427	.1033046
L2.	.0500302	.0301605	1.66	0.097	-.0090833	.1091437
dln_inc						
L1.	-.1527311	.131759	-1.16	0.246	-.4109741	.1055118
L2.	.0191634	.1290799	0.15	0.882	-.2338285	.2721552
dln_consump						
L1.	.2884992	.1604069	1.80	0.072	-.0258926	.6028909
L2.	-.0102	.1605968	-0.06	0.949	-.3249639	.3045639
_cons	.0157672	.0041596	3.79	0.000	.0076146	.0239198
dln_consump						
dln_inv						
L1.	-.002423	.0244142	-0.10	0.921	-.050274	.045428
L2.	.0338806	.0243072	1.39	0.163	-.0137607	.0815219
dln_inc						
L1.	.2248134	.1061884	2.12	0.034	.0166879	.4329389
L2.	.3549135	.1040292	3.41	0.001	.1510199	.558807
dln_consump						
L1.	-.2639695	.1292766	-2.04	0.041	-.517347	-.010592
L2.	-.0222264	.1294296	-0.17	0.864	-.2759039	.231451
_cons	.0129258	.0033523	3.86	0.000	.0063554	.0194962

The output table reports summaries for 21 regression coefficients. But in VAR models, it is usually more instructive to analyze how shocks on a dependent variable affect other dependent variables and the variable itself over time. In this example, we focus on technical aspects of fitting Bayesian VAR models and the immediate impact on regression coefficients and covariances. Later in example 8, we demonstrate how to use more common impulse–response functions (IRFs) to interpret results.

Let us start by fitting the same VAR model using the `bayes: var` command with the default model prior—conjugate Minnesota prior for regression coefficients and error covariance. In addition to the `bayes` prefix, we specify the `rseed()` option for reproducibility and run three MCMC chains to compute a Gelman–Rubin convergence diagnostic; see *Convergence diagnostics using multiple chains*.

```
. bayes, rseed(17)  nchains(3):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)

Chain 1
  Burn-in ...
  Simulation ...

Chain 2
  Burn-in ...
  Simulation ...

Chain 3
  Burn-in ...
  Simulation ...

Model summary
------------------------------------------------------------------------------

Likelihood:
      dln_inv
      dln_inc
  dln_consump ~ mvnormal(3,xb_dln_inv,xb_dln_inc,xb_dln_consump,{Sigma,m})

Priors:
          {dln_inv:L(1 2).dln_inv}                                        (1)
          {dln_inv:L(1 2).dln_inc}                                        (1)
      {dln_inv:L(1 2).dln_consump}                                        (1)
                  {dln_inv:_cons}                                         (1)
          {dln_inc:L(1 2).dln_inv}                                        (2)
          {dln_inc:L(1 2).dln_inc}                                        (2)
      {dln_inc:L(1 2).dln_consump}                                        (2)
                  {dln_inc:_cons}                                         (2)
      {dln_consump:L(1 2).dln_inv}                                        (3)
      {dln_consump:L(1 2).dln_inc}                                        (3)
  {dln_consump:L(1 2).dln_consump}                                        (3)
              {dln_consump:_cons} ~ varconjugate(3,2,1,_b0,{Sigma,m},_Phi0)
                                    (3)
                        {Sigma,m} ~ iwishart(3,5,_Sigma0)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_dln_inv.
(2) Parameters are elements of the linear form xb_dln_inc.
(3) Parameters are elements of the linear form xb_dln_consump.
```

```
Bayesian vector autoregression                  Number of chains    =         3
Gibbs sampling                                  Per MCMC chain:
                                                    Iterations      =    12,500
                                                    Burn-in         =     2,500
                                                    Sample size     =    10,000
Sample: 1960q4 thru 1978q4                      Number of obs       =        73
                                                Avg acceptance rate =         1
                                                Avg efficiency: min =     .9755
                                                                avg =      .994
                                                                max =         1
Avg log marginal-likelihood =  483.43596        Max Gelman-Rubin Rc =         1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
dln_inv						
dln_inv						
L1.	.4808475	.103581	.000598	.480019	.2786408	.6837882
L2.	.0068788	.0627703	.000362	.0069781	-.1167651	.1290908
dln_inc						
L1.	.1026098	.4103202	.002369	.1044135	-.7070827	.8989748
L2.	.0320344	.2434573	.001406	.032674	-.4520181	.5169279
dln_consump						
L1.	-.0181305	.4774359	.002766	-.0166627	-.9574952	.9252084
L2.	.0297566	.2885481	.001687	.0288948	-.5385765	.5989857
_cons	.0063813	.0152462	.000088	.0063497	-.0235027	.0364822
dln_inc						
dln_inv						
L1.	.0148781	.0245811	.000142	.0149684	-.033299	.0630655
L2.	.001391	.0147206	.000086	.0013496	-.0272677	.0305663
dln_inc						
L1.	.5782111	.0966633	.000564	.5787724	.3873585	.7675225
L2.	.0130696	.0576478	.000333	.0131284	-.0985297	.126328
dln_consump						
L1.	-.0315052	.1143589	.000664	-.0311991	-.2557682	.1961411
L2.	-.0193878	.0681031	.000393	-.0194134	-.1543933	.1152181
_cons	.0087345	.0036292	.000021	.0087388	.0016601	.0158279
dln_consump						
dln_inv						
L1.	-.0183338	.0216079	.000125	-.0182276	-.0610608	.0238827
L2.	.0086858	.0131225	.000076	.0087476	-.0172135	.0344555
dln_inc						
L1.	-.0283731	.0857885	.000498	-.0287275	-.1961209	.1409767
L2.	.0344015	.0508225	.000297	.0344959	-.0658067	.1335025
dln_consump						
L1.	.5452017	.1011028	.000584	.5452941	.3461853	.7423402
L2.	.0528311	.0603558	.00035	.0523857	-.0640511	.1727009
_cons	.0078026	.0032046	.000019	.0077938	.0015249	.014078

Sigma_1_1	.0039149	.0006512	3.8e-06	.0038459	.002843	.0054003
Sigma_2_1	-.0000195	.0001079	6.2e-07	-.0000193	-.0002359	.0001924
Sigma_3_1	.0001329	.000097	5.6e-07	.0001291	-.0000493	.0003346
Sigma_2_2	.000219	.0000365	2.1e-07	.0002148	.0001587	.0003014
Sigma_3_2	.0000463	.0000232	1.3e-07	.0000451	4.14e-06	.000096
Sigma_3_3	.0001703	.0000282	1.6e-07	.0001673	.0001239	.0002344

Note: Default initial values are used for multiple chains.

The simulation is performed using Gibbs sampling, which provides high sampling efficiency, 0.99 on average. The maximum Gelman–Rubin Rc statistic is a perfect 1, which suggests no convergence issues. Because of this and to speed up computation, we will use only one chain in subsequent examples.

The model summary provides description of the model. We have an MVN likelihood for the error terms. Regression coefficients are assigned a conjugate Minnesota prior, which is labeled as `varconjugate(3,2,1,_b0,{Sigma,m},_Phi0)` in the output. The arguments are the number of dependent variables (3), number of lags (2), number of exogenous variables (0) plus a constant term per equation (1), default prior mean vector (`_b0`), error covariance matrix parameter (`{Sigma,m}`), and Minnesota factor covariance (`_Phi0`), which is a function of tightness parameters that control the concentration of the prior around its mean. The conjugate Minnesota prior for the coefficients is MVN with mean vector $\boldsymbol{\beta}_0$ and covariance $\Sigma \otimes \Phi_0$, where $\boldsymbol{\beta}_0$ and Φ_0 are defined in *Methods and formulas*. We discuss the Minnesota prior in detail in example 5. If you are not familiar with this prior, you may want to look at this example.

Error covariance `{Sigma,m}` is assigned an inverse-Wishart prior with default degrees of freedom $\text{df} = K + 2 = 5$ and scale matrix `_Scale0`$= (\text{df} - K - 1)$`_Sigma0`$=$`_Sigma0`, where `_Sigma0` is the diagonal AR covariance matrix, a diagonal matrix formed by error-variance estimates from fitting a separate AR model to each dependent variable; see *VAR model specification* in *Methods and formulas*.

The table of results contains three groups of regression parameters, one for each dependent variable, just like the output from the `var` command. `bayes: var` additionally displays the estimates of the error covariance `{Sigma,m}`. The output table reports standard Bayesian posterior summaries ([BAYES] **bayesstats summary**).

The prior mean vector `_b0` is 1 for the coefficients corresponding to the first own lags of dependent variables, which we also refer to as self-variables first-lag coefficients, and 0 otherwise. In the output table, these are labeled as `{dln_inv:L1.dln_inv}`, `{dln_inc:L1.dln_inc}`, and `{dln_consump:L1.dln_consump}`. As such, the prior is centered around each variable being a univariate random walk. The estimated posterior means for the coefficients reflect the strong prior assumptions in the model. For example, the estimated posterior mean of `{dln_inv:L1.dln_inv}` is 0.48 with a 95% CrI of $[0.28, 0.68]$ compared with the estimates from the `var` command of -0.32 with a 95% CI of $-0.55, -0.086$, which are quite different. Similarly, the posterior mean estimate for `{dln_inc:L1.dln_inc}` is 0.58 versus -0.15 and for `{dln_consump:L1.dln_consump}` is 0.55 versus -0.26. Continuing with the `dln_consump` equation, we see that the posterior mean estimates of cross-variable lag coefficients are small. The estimated posterior mean of the first lag of income, `{dln_consump:L1.dln_inc}`, is -0.03, and its 95% CrI includes 0. From the `var` results, `{dln_consump:L1.dln_inc}` is 0.22 and is statistically significantly different from 0 (with p-value $=$ 0.034).

All three self-variables first-lag coefficients have positive posterior estimates: means, medians, and 95% CrIs. Posterior estimates of remaining coefficients are close to 0. The results suggest a strong AR impact for each dependent variable and weak cross-correlations between the variables. The {Sigma,m} estimates show that there is some residual correlation in the error terms unexplained by the regression coefficients. The prior thus dominates the information about regression coefficients available in the data. This can be partially explained by the relatively small sample size of only 73 observations given the number of estimated parameters.

The results from the VAR models rely on the stability assumption. Thus, it is important to test this assumption, as we demonstrate in example 4. When the assumption is satisfied, as it is for these data, you may consider specifying priors for regression coefficients that are centered around zero; for instance, using these priors for our dataset produces results that are similar to those from var (example 2).

◁

▷ Example 2: Bayesian VAR model with original Minnesota prior

In early work on Bayesian VAR (Doan, Litterman, and Sims 1984 and Litterman 1986), researchers simplified the model prior by assuming a known, fixed-error covariance matrix. The covariance Σ in the MVN likelihood is replaced by an estimate $\widehat{\Sigma}$. A typical choice for $\widehat{\Sigma}$ is a diagonal matrix of variance estimates obtained by fitting a separate AR model to each dependent variable. The prior covariance for regression coefficients is then obtained from $\widehat{\Sigma}$ as described in *Original Minnesota prior with known (fixed) error covariance* in *Methods and formulas*. This prior specification is known as the original Minnesota prior. Also see example 5.

To fit a model with the original Minnesota prior, we specify the minnfixedcovprior option with bayes: var.

```
. bayes, minnfixedcovprior rseed(17):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
      dln_inv
      dln_inc
  dln_consump ~ mvnormal(3,xb_dln_inv,xb_dln_inc,xb_dln_consump,_Sigma0)
Priors:
          {dln_inv:L(1 2).dln_inv}                                        (1)
          {dln_inv:L(1 2).dln_inc}                                        (1)
      {dln_inv:L(1 2).dln_consump}                                        (1)
                 {dln_inv:_cons}                                          (1)
          {dln_inc:L(1 2).dln_inv}                                        (2)
          {dln_inc:L(1 2).dln_inc}                                        (2)
      {dln_inc:L(1 2).dln_consump}                                        (2)
                 {dln_inc:_cons}                                          (2)
      {dln_consump:L(1 2).dln_inv}                                        (3)
      {dln_consump:L(1 2).dln_inc}                                        (3)
  {dln_consump:L(1 2).dln_consump}                                        (3)
  {dln_consump:_cons} ~ minnesota(3,2,1,_b0,_Sigma0,.1,.5,1,100)          (3)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_dln_inv.
(2) Parameters are elements of the linear form xb_dln_inc.
(3) Parameters are elements of the linear form xb_dln_consump.
```

```
Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1960q4 thru 1978q4                      Number of obs    =         73
                                                Acceptance rate  =          1
                                                Efficiency:  min =       .946
                                                             avg =      .9957
Log marginal-likelihood =  478.02208                         max =          1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
dln_inv						
dln_inv						
L1.	.4836549	.0751107	.000751	.4825203	.3359447	.6314641
L2.	.0077444	.0458064	.000458	.0070614	-.0813891	.0984996
dln_inc						
L1.	.0370079	.1779866	.00178	.0381258	-.3085588	.3854122
L2.	.0090371	.0963583	.000964	.0081537	-.1793915	.1992546
dln_consump						
L1.	-.0028656	.2124749	.002125	-.0027569	-.4232734	.410695
L2.	.0094103	.1125252	.001125	.0087853	-.210387	.2303232
_cons	.0081521	.0082618	.000082	.0080767	-.0079144	.0244083
dln_inc						
dln_inv						
L1.	.0052036	.0117865	.000118	.0051966	-.0176583	.0288828
L2.	.0003523	.0063033	.000061	.0003332	-.011977	.012503
dln_inc						
L1.	.5758156	.0761506	.000776	.5767133	.4271662	.7249808
L2.	.0120131	.0457046	.000457	.0124684	-.0785969	.1006224
dln_consump						
L1.	-.0081978	.0543999	.000537	-.0080116	-.1156294	.0990992
L2.	-.0057737	.0288414	.000288	-.0056489	-.0627868	.0500214
_cons	.0082507	.0024756	.000025	.0082296	.0035138	.0131572
dln_consump						
dln_inv						
L1.	-.0068309	.0099134	.000102	-.0067224	-.0262589	.0127061
L2.	.002545	.0052876	.000053	.0025395	-.0079491	.0129162
dln_inc						
L1.	-.0091519	.0393528	.000394	-.0091127	-.0874506	.0692072
L2.	.0101553	.0207397	.000207	.0101049	-.0305975	.0503957
dln_consump						
L1.	.5358264	.0760533	.000752	.5362025	.3870167	.6834547
L2.	.0540704	.0459402	.000459	.0538069	-.0364033	.1445824
_cons	.007971	.0022349	.000022	.0079594	.0036197	.0123659

Compared with the default conjugate Minnesota prior from example 1, the error covariance matrix {Sigma,m} in the likelihood is replaced with a fixed matrix _Sigma0, a diagonal AR covariance estimate. The regression coefficients are assigned the minnesota(3,2,1,_b0,_Sigma0,.1,.5,1,100) prior. Most of the prior arguments are as we described in example 1, except the covariance matrix is now formed by _Sigma0 and tightness parameters (0.1, 0.5, 1, 100); see *Original Minnesota prior with known (fixed) error covariance*. Specifically, the default for the self-variables tightness parameter λ_1 is 0.1 (option selftight()), the default for the cross-variables tightness parameter λ_2 is 0.5 (option crosstight()), the default for the lag-decay parameter λ_3 is 1 (option lagdecay()), and the default for the exogenous variable tightness parameter is 100 (option exogtight()).

Like the default conjugate Minnesota prior, the original Minnesota prior places the same strong prior assumptions on regression coefficients: the prior mean vector _b0 contains 1 for self-variables first-lag coefficients and 0s for all other coefficients. The strength of the shrinkage toward the prior mean _b0 is controlled mainly by the tightness parameter λ_1, which can be reset using the Minnesota prior option selftight(). Coefficients of exogenous variables, including the constant terms, are shrunk toward 0 but are given wide prior variance controlled by the tightness parameter λ_4 and specified in the exogtight() option.

As expected, the results assuming the original Minnesota prior are closer to those assuming the default conjugate Minnesota prior than to those from the var command. In the absence of strong information about model parameters in the data, the Minnesota prior may introduce a stronger time dependence in the results. For example, the prior mean value for {dln_inv:L1.dln_inv} is 1 and the posterior mean estimate is 0.48, whereas the estimate from the var command is -0.32. It is completely acceptable to have a negative first-lag correlation in the change of investments at quarterly level. The Minnesota prior, however, expects an increase in investments to be followed by another increase in investment in the next time period. The question of whether this is a reasonable prior expectation is an empirical question. It is thus important to understand the behavior of the default Minnesota prior and use it carefully.

To relax the time-dependence assumption of the Minnesota prior, we can change the prior mean _b0 to be a zero vector and decrease the tightness of the prior by increasing the λ_1 parameter from the default of 0.1 to 1. The prior for the self-variables first-lag coefficients thus changes from $N(1, 0.01)$ to $N(0, 1)$ and those for the cross-variables first-lag coefficients from $N(0, 0.0025)$ to $N(0, 0.25)$.

We change the defaults by specifying the respective suboptions within the minnfixedcovprior() option. There are several ways to specify the prior mean values. We can provide a full 1×21 vector of mean values. Or, if we want to change the default values only for self-variables first-lag coefficients, we can specify a vector of lower dimension, 1×3 in our example. The remaining coefficients will be automatically set to zeros. Alternatively, for self-variables first-lag coefficients, we can list the values directly in the mean() suboption, that is, mean(0,0,0). We use the second approach below and specify a zero mean vector for self-variables first-lag coefficients.

```
. matrix b0 = J(1,3,0)
. bayes, minnfixedcovprior(mean(b0) selftight(1)) rseed(17):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)
Burn-in ...
Simulation ...
Model summary
------------------------------------------------------------------------------
Likelihood:
      dln_inv
      dln_inc
  dln_consump ~ mvnormal(3,xb_dln_inv,xb_dln_inc,xb_dln_consump,_Sigma0)
Priors:
          {dln_inv:L(1 2).dln_inv}                                        (1)
          {dln_inv:L(1 2).dln_inc}                                        (1)
      {dln_inv:L(1 2).dln_consump}                                        (1)
                  {dln_inv:_cons}                                         (1)
          {dln_inc:L(1 2).dln_inv}                                        (2)
          {dln_inc:L(1 2).dln_inc}                                        (2)
      {dln_inc:L(1 2).dln_consump}                                        (2)
                  {dln_inc:_cons}                                         (2)
      {dln_consump:L(1 2).dln_inv}                                        (3)
      {dln_consump:L(1 2).dln_inc}                                        (3)
  {dln_consump:L(1 2).dln_consump}                                        (3)
              {dln_consump:_cons} ~ minnesota(3,2,1,b0,_Sigma0,1,.5,1,100) (3)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_dln_inv.
(2) Parameters are elements of the linear form xb_dln_inc.
(3) Parameters are elements of the linear form xb_dln_consump.
Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in         =      2,500
                                                MCMC sample size =     10,000
Sample: 1960q4 thru 1978q4                      Number of obs    =         73
                                                Acceptance rate  =          1
                                                Efficiency:  min =       .946
                                                             avg =      .9946
Log marginal-likelihood =  539.71278                         max =          1
------------------------------------------------------------------------------
             |                                                 Equal-tailed
             |      Mean   Std. dev.     MCSE     Median  [95% cred. interval]
-------------+----------------------------------------------------------------
dln_inv      |
     dln_inv |
         L1. |  -.2987647   .1218328   .001218   -.300605  -.5383571  -.0590116
         L2. |  -.1415209   .1192125   .001192  -.1439763  -.3729553   .0926795
             |
     dln_inc |
         L1. |   .2014271   .5068694   .005069      .2036  -.7741228   1.196341
         L2. |   .1683548   .4567907   .004512   .1662872  -.7269593   1.058442
             |
 dln_consump |
         L1. |   .8313647   .6128113   .006128   .8291778  -.3631793   2.037414
         L2. |   .6988162    .554944   .005549   .6984739  -.3870548   1.810489
             |
       _cons |  -.0124058    .016376   .000161  -.0123152  -.0449067   .0197023
------------------------------------------------------------------------------
```

dln_inc						
dln_inv						
L1.	.0401237	.0307154	.000307	.0401054	-.0194542	.1018312
L2.	.0397051	.0285189	.000276	.0396625	-.0158618	.0955686
dln_inc						
L1.	-.1359873	.1350215	.001376	-.133623	-.4004453	.1259119
L2.	.0225672	.1266313	.001266	.0237303	-.2302853	.2689963
dln_consump						
L1.	.269855	.1601578	.001602	.2691198	-.0423728	.579863
L2.	-.003682	.1444577	.001445	-.001067	-.2915697	.276431
_cons	.0158543	.0041686	.000042	.0158808	.0075933	.0241443
dln_consump						
dln_inv						
L1.	-.0046515	.0259292	.000267	-.0043679	-.055467	.0464489
L2.	.0277595	.0239298	.000239	.0277642	-.0190999	.0748595
dln_inc						
L1.	.1971296	.1126233	.001126	.1963476	-.025344	.4199598
L2.	.273373	.100819	.001008	.2730468	.076632	.4698413
dln_consump						
L1.	-.2200755	.1392633	.001393	-.2176729	-.4890732	.0503819
L2.	.0383448	.1342118	.001342	.0392141	-.2271728	.2978496
_cons	.0132401	.0036414	.000036	.0132355	.0062362	.0203544

Now the posterior mean estimates of regression coefficients are similar to the estimates from the original `var` command. For example, the posterior mean estimate of `{dln_inv:L1.dln_inv}` is about −0.30 compared with `var`'s estimate of −0.32.

The original Minnesota prior always assumes no correlation between cross-equation error terms. The following two priors relax this assumption.

◁

▷ Example 3: MVN priors with unrestricted error covariances

What if we want to relax the assumption about the error covariance imposed by the original Minnesota prior? We can use a MVN-inverse-Wishart prior (option `minniwishprior`) or MVN-Jeffreys prior (option `minnjeffprior`). These priors use the same default MVN prior for the regression coefficients as the original Minnesota prior, but they assume an unrestricted error covariance and use the respective inverse-Wishart or Jeffreys prior for it.

Let's start with an MVN-inverse-Wishart prior. Continuing with example 2, we change the default prior means for the regression coefficients to be zeros by specifying zero values for the three self-variables first-lag coefficients in the `mean()` option. This specification automatically assigns zero prior means for all other coefficients. We also use the self-variables tightness parameter of 1 instead of the default 0.1 to loosen the prior variance tightness.

```
. bayes, minniwishprior(mean(0,0,0) selftight(1)) rseed(17):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)

Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
      dln_inv
      dln_inc
  dln_consump ~ mvnormal(3,xb_dln_inv,xb_dln_inc,xb_dln_consump,{Sigma,m})

Priors:
          {dln_inv:L(1 2).dln_inv}                                        (1)
          {dln_inv:L(1 2).dln_inc}                                        (1)
      {dln_inv:L(1 2).dln_consump}                                        (1)
                  {dln_inv:_cons}                                         (1)
          {dln_inc:L(1 2).dln_inv}                                        (2)
          {dln_inc:L(1 2).dln_inc}                                        (2)
      {dln_inc:L(1 2).dln_consump}                                        (2)
                  {dln_inc:_cons}                                         (2)
      {dln_consump:L(1 2).dln_inv}                                        (3)
      {dln_consump:L(1 2).dln_inc}                                        (3)
  {dln_consump:L(1 2).dln_consump}                                        (3)
              {dln_consump:_cons} ~ varmvnormal(3,2,1,(0,0,0),_Omega0)    (3)
                        {Sigma,m} ~ iwishart(3,5,_Sigma0)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_dln_inv.
(2) Parameters are elements of the linear form xb_dln_inc.
(3) Parameters are elements of the linear form xb_dln_consump.

Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1960q4 thru 1978q4                      Number of obs    =         73
                                                Acceptance rate  =          1
                                                Efficiency:  min =      .8113
                                                             avg =      .9438
Log marginal-likelihood =  527.12015                         max =          1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
dln_inv						
dln_inv						
L1.	-.2510453	.1176975	.001153	-.2528143	-.4765424	-.0235274
L2.	-.1063315	.116444	.001164	-.1059882	-.3344738	.1229004
dln_inc						
L1.	.2446635	.3498178	.003498	.2500731	-.4573745	.9259674
L2.	.095764	.219266	.002193	.0971031	-.3381891	.5251107
dln_consump						
L1.	.3645458	.3744485	.003811	.3680423	-.3646316	1.108422
L2.	.1400995	.2298	.002392	.1395888	-.310524	.5901507
_cons	.0074369	.0119759	.000123	.0073878	-.0153556	.0307578

dln_inc						
dln_inv						
L1.	.046923	.0316428	.000316	.0469129	-.0145534	.1098345
L2.	.0505845	.0310234	.000319	.0506224	-.0100996	.1122741
dln_inc						
L1.	-.1526888	.1310382	.00131	-.1527139	-.4111528	.106306
L2.	-.0118679	.1209026	.001225	-.0130416	-.2506488	.2254785
dln_consump						
L1.	.2586053	.1552061	.001552	.259664	-.0444987	.5662822
L2.	-.013651	.1354632	.001407	-.0124392	-.278631	.2532518
_cons	.0170262	.0041118	.000043	.0170605	.0090405	.0250246
dln_consump						
dln_inv						
L1.	.000902	.0253473	.000253	.0008	-.0484133	.0512205
L2.	.0365412	.025467	.000261	.0366538	-.0138045	.0861043
dln_inc						
L1.	.2124569	.1058319	.001058	.2127713	.0019291	.421758
L2.	.2993713	.0940064	.000973	.2987598	.1160611	.4852742
dln_consump						
L1.	-.2757223	.1279485	.001279	-.2756011	-.5238806	-.0256011
L2.	-.0293205	.1199357	.001199	-.0295581	-.2669714	.2049208
_cons	.0146112	.003401	.000035	.0145617	.0080611	.0213525
Sigma_1_1	.0021287	.0003691	3.9e-06	.0020854	.0015264	.0029714
Sigma_2_1	.0000718	.0000664	7.3e-07	.0000693	-.0000518	.0002108
Sigma_3_1	.0001215	.0000558	6.1e-07	.0001178	.0000212	.0002401
Sigma_2_2	.0001363	.0000239	2.6e-07	.0001338	.0000971	.0001908
Sigma_3_2	.0000601	.0000153	1.7e-07	.0000588	.0000341	.0000943
Sigma_3_3	.0000892	.0000155	1.7e-07	.0000875	.0000638	.0001243

In the model summary, the regression coefficients are assigned the `varmvnormal()` prior, in which the prior covariance matrix `_Omega0` is a function of tightness parameters, the same as with the original Minnesota prior.

The inverse-Wishart prior for the error covariance matrix is controlled by the degrees of freedom and the scaling matrix. The default degrees of freedom $\text{df} = K + 2 = 3 + 2 = 5$, and the default scale is `_Scale0`$= (\text{df} - K - 1)$`_Sigma0`=`_Sigma0`. The low degrees of freedom of the inverse-Wishart prior constrain the `{Sigma,m}` matrix parameter to be close to the scaling matrix `_Sigma0`.

The results are somewhat similar to those using the original Minnesota prior, but the error covariance matrix is now being estimated. Some of the covariance estimates are bounded away from zero based on their estimated CrIs, which suggests that the assumption of no correlation between the error terms imposed by the original Minnesota prior may not be appropriate for these data. Note that these results are closer to the results obtained from the `var` command.

Instead of assuming an inverse-Wishart prior for the error covariance, we can use the multivariate Jeffreys prior.

```
. bayes, minnjeffprior(mean(0,0,0) selftight(1)) rseed(17):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)

Burn-in ...
Simulation ...

Model summary
------------------------------------------------------------------------------
Likelihood:
      dln_inv
      dln_inc
  dln_consump ~ mvnormal(3,xb_dln_inv,xb_dln_inc,xb_dln_consump,{Sigma,m})

Priors:
          {dln_inv:L(1 2).dln_inv}                                        (1)
          {dln_inv:L(1 2).dln_inc}                                        (1)
      {dln_inv:L(1 2).dln_consump}                                        (1)
                  {dln_inv:_cons}                                         (1)
          {dln_inc:L(1 2).dln_inv}                                        (2)
          {dln_inc:L(1 2).dln_inc}                                        (2)
      {dln_inc:L(1 2).dln_consump}                                        (2)
                  {dln_inc:_cons}                                         (2)
      {dln_consump:L(1 2).dln_inv}                                        (3)
      {dln_consump:L(1 2).dln_inc}                                        (3)
  {dln_consump:L(1 2).dln_consump}                                        (3)
              {dln_consump:_cons} ~ varmvnormal(3,2,1,(0,0,0),_Omega0)    (3)
                        {Sigma,m} ~ jeffreys(3)
------------------------------------------------------------------------------
(1) Parameters are elements of the linear form xb_dln_inv.
(2) Parameters are elements of the linear form xb_dln_inc.
(3) Parameters are elements of the linear form xb_dln_consump.

Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1960q4 thru 1978q4                      Number of obs    =         73
                                                Acceptance rate  =          1
                                                Efficiency:  min =      .8186
                                                             avg =      .9489
Log marginal-likelihood =  535.28175                         max =          1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
dln_inv						
dln_inv						
L1.	-.2455836	.1221811	.001236	-.2462348	-.4869383	-.0050753
L2.	-.1025647	.1181023	.001181	-.102274	-.3362211	.127451
dln_inc						
L1.	.2298239	.3566309	.003566	.2342159	-.4802715	.9178026
L2.	.0920532	.2204407	.002259	.0922503	-.332021	.5241679
dln_consump						
L1.	.3544481	.3829504	.00383	.3546005	-.4036804	1.108751
L2.	.1308923	.230888	.002307	.130488	-.3293822	.5811411
_cons	.00804	.012356	.000125	.0080513	-.0161679	.0326142

dln_inc						
dln_inv						
L1.	.0467331	.0327457	.000331	.0468285	-.0167597	.1107341
L2.	.0501114	.0318974	.000319	.050056	-.0133188	.1128243
dln_inc						
L1.	-.1506219	.1354065	.001354	-.1523624	-.4147838	.1141846
L2.	-.0144403	.1264436	.001279	-.0141583	-.2584229	.2348877
dln_consump						
L1.	.2593289	.1596995	.001637	.2588021	-.0541279	.5715087
L2.	-.0130386	.1386775	.001409	-.0140483	-.2836371	.2634825
_cons	.0170224	.004309	.000044	.0170312	.0085191	.0255192
dln_consump						
dln_inv						
L1.	.0011214	.026178	.000262	.0010064	-.0490433	.0534858
L2.	.0364058	.0259021	.000259	.036759	-.0159044	.0879265
dln_inc						
L1.	.2110716	.1078844	.001117	.2120819	-.0014118	.4214028
L2.	.2979752	.0981546	.000982	.2974221	.1032472	.4875104
dln_consump						
L1.	-.2786814	.1301325	.001329	-.2805229	-.5309565	-.0213882
L2.	-.0292443	.1226758	.001257	-.0298197	-.270218	.2118989
_cons	.014751	.0035158	.000036	.0147164	.0078041	.021617
Sigma_1_1	.0022852	.000416	4.4e-06	.0022362	.0016076	.0032668
Sigma_2_1	.000077	.0000744	8.1e-07	.0000744	-.0000643	.0002339
Sigma_3_1	.0001311	.0000621	6.8e-07	.0001268	.000018	.0002672
Sigma_2_2	.0001475	.0000269	3.0e-07	.0001445	.0001042	.0002088
Sigma_3_2	.0000659	.0000175	1.9e-07	.0000642	.0000367	.0001061
Sigma_3_3	.0000961	.0000177	1.9e-07	.0000941	.0000676	.0001365

The results are similar to the MVN-inverse-Wishart prior results. The change in the prior for the error covariance did not change its estimates much, which again confirms that the data contribution to the posterior model is weak.

◁

▷ Example 4: Testing Bayesian VAR stability

A VAR model has meaningful interpretation in terms of IRFs and forecast-error variance decompositions only if the time-series process it represents is stable. The default Minnesota prior is based on the assumption that each dependent variable follows a univariate random walk, which is an unstable process. In the absence of strong information about model parameters in the data, the posterior is shrunk more toward the prior, so it is possible that Bayesian posterior estimates may not satisfy the stability assumption even when the frequentist estimates from the VAR model do. Thus, a stability check for a Bayesian VAR is particularly important.

In a frequentist setting, VAR stability can be checked by inspecting the eigenvalues of the companion matrix using the [TS] **varstable** command. In a Bayesian setting, the companion matrix and its eigenvalues are random, so we must inspect their posterior distributions. The Bayesian command for testing stationarity, `bayesvarstable`, reports posterior summaries for the eigenvalue moduli. Stability is declared when all eigenvalues are within the unit circle with high probability.

Let us refit the Bayesian VAR model from example 1 using the default prior options. In addition, we save simulation results in `bvarsex1.dta`, which is required by `bayesvarstable`. Because we already discussed the estimation results, we rerun the command quietly.

```
. quietly bayes, rseed(17) saving(bvarex1):
> var dln_inv dln_inc dln_consump if qtr<=tq(1978q4)
```

Now we call `bayesvarstable` to check the stability condition.

```
. bayesvarstable
Eigenvalue stability condition                  Companion matrix size =     6
                                                MCMC sample size      = 10000
```

Eigenvalue modulus	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
1	.7204885	.0946585	.000947	.7185141	.5401149	.911899
2	.5959965	.1036993	.001037	.6005058	.3817847	.7834288
3	.4271097	.1243872	.001244	.4243446	.2125586	.6563634
4	.2109317	.0790968	.000791	.1972916	.0886465	.3853979
5	.1357284	.0561101	.000561	.1322702	.0390514	.254025
6	.075227	.0499245	.000499	.0688643	.0033007	.1854592

```
Pr(eigenvalues lie inside the unit circle) = 0.9975
```

The companion matrix is of dimension 6 (3 dependent variables times 2 lags), so the output table reports posterior summaries for the moduli of 6 eigenvalues. The eigenvalues are displayed in decreasing order of their moduli. The largest eigenvalue modulus has a posterior mean of 0.72 and is within the unit circle. The command also reports the posterior probability that all eigenvalues lie in the unit circle, 0.9975. The high value of this probability provides confidence that the stability condition is satisfied.

◁

US macroeconomic examples

In the next set of examples, we will use `usmacro.dta`, quarterly macroeconomic data extracted from the Federal Reserve Economic Database that spans from 1954 to 2010.

```
. use https://www.stata-press.com/data/r19/usmacro
(Federal Reserve Economic Data - St. Louis Fed)
. describe
Contains data from https://www.stata-press.com/data/r19/usmacro.dta
 Observations:           226                  Federal Reserve Economic Data -
                                                St. Louis Fed
    Variables:             4                  4 Dec 2024 12:39
─────────────────────────────────────────────────────────────────────────────
Variable      Storage   Display    Value
    name         type    format    label      Variable label
─────────────────────────────────────────────────────────────────────────────
fedfunds        double  %10.0g                Federal funds rate
date            int     %tq                   Date (quarters)
inflation       float   %9.0g                 Annual rate of inflation
ogap            float   %9.0g                 GDP gap
─────────────────────────────────────────────────────────────────────────────
Sorted by: date
. tsset
Time variable: date, 1954q3 to 2010q4
        Delta: 1 quarter
```

Observed are three dependent variables: `fedfunds`, for federal funds rate, `inflation`, for annual rate of inflation, and `ogap`, for the GDP output gap, or the difference between actual and potential GDP. The `date` variable registers the quarterly periods.

▷ Example 5: Explaining the Minnesota prior

Consider the following simple VAR(2) model for `usmacro.dta` with dependent variables `ogap` and `inflation`:

$$\texttt{ogap} = a_{11}\texttt{L.ogap} + a_{12}\texttt{L2.ogap} + a_{21}\texttt{L.inflation} + a_{22}\texttt{L2.inflation} + a_0 + u_1$$

$$\texttt{inflation} = b_{11}\texttt{L.ogap} + b_{12}\texttt{L2.ogap} + b_{21}\texttt{L.inflation} + b_{22}\texttt{L2.inflation} + b_0 + u_2$$

In the specification of the original Minnesota prior, (u_1, u_2) is assumed to follow a bivariate normal distribution with 0 means and fixed error covariance $\boldsymbol{\Sigma}_0 = \text{diag}(\hat{\sigma}_1^2, \hat{\sigma}_2^2)$, which we define later.

Consider the vector $\boldsymbol{\beta}$ of 8 endogenous regression coefficients a_{ij}'s and b_{ij}'s and 2 exogenous constant terms a_0 and b_0. Specifically, we refer to a_{11}, a_{12}, b_{21}, and b_{22} as endogenous-self-variables lag coefficients (or simply self-variables coefficients); to a_{21}, a_{22}, b_{11}, and b_{12} as endogenous-cross-variables lag coefficients (or simply cross-variables coefficients); and to a_0 and b_0 as "exogenous variable" coefficients. We used quotes for a_0 and b_0 because, technically, these are constant terms that do not correspond to any exogenous variables. But in what follows, we will treat them as such. In the presence of exogenous variables, we would refer to their coefficients as exogenous variable coefficients. We also refer to a_{11} and b_{21} as self-variables first-lag coefficients, also know as first own lag coefficients.

The original Minnesota prior for $\boldsymbol{\beta}$ is MVN with 10×1 mean vector $\boldsymbol{\beta}_0$ and 10×10 covariance Ω_0, where $\boldsymbol{\beta}_0$ and Ω_0 are defined in *Original Minnesota prior with known (fixed) error covariance*. $\boldsymbol{\beta}_0$ contains 1 for all self-variables first-lag coefficients and 0 for all the other coefficients. Ω_0 is a diagonal matrix in which diagonals are functions of error variance estimates from individual AR models and tightness parameters. Because the covariance matrix Ω_0 is diagonal, all regression coefficients are assumed uncorrelated a priori.

In our example, the error variance estimates are the ordinary least-squares (OLS) residual variance estimates $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ obtained from fitting separately the following two AR models,

$$\texttt{ogap} = c_1\texttt{L.ogap} + c_2\texttt{L2.ogap} + c_3 + e_1$$

$$\texttt{inflation} = d_1\texttt{L.inflation} + d_2\texttt{L2.inflation} + d_3 + e_2$$

where $e_i \sim N(0, \sigma_i^2)$ for $i = 1, 2$.

The Minnesota prior has four control (tightness) parameters: λ_1, λ_2, λ_3, and λ_4, with default values of 0.1, 0.5, 1, and 100. These parameters can be reset using the `selftight()`, `crosstight()`, `lagdecay()`, and `exogtight()` Minnesota prior options, respectively.

Below, we show the default prior distributions for all coefficients. Let l denote the current lag.

Priors for endogenous-self-variables first-lag and second-lag coefficients are

$$a_{11}, b_{21} \sim N(1,\ 0.01)$$

$$a_{12}, b_{22} \sim N(0,\ 0.0025)$$

where $\lambda_1^2/l^{2\lambda_3} = \lambda_1^2 = 0.01$ for $l = 1$ and $\lambda_1^2/l^{2\lambda_3} = 0.0025$ for $l = 2$.

Priors for endogenous-cross-variables first-lag and second-lag coefficients are

$$a_{21} \sim N(0,\ 0.0025\frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2})$$

$$b_{11} \sim N(0,\ 0.0025\frac{\hat{\sigma}_2^2}{\hat{\sigma}_1^2})$$

$$a_{22} \sim N(0,\ 0.000625\frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2})$$

$$b_{12} \sim N(0,\ 0.000625\frac{\hat{\sigma}_2^2}{\hat{\sigma}_1^2})$$

where $(\lambda_1^2\lambda_2^2)/l^{2\lambda_3} = \lambda_1^2\lambda_2^2 = 0.0025$ for $l = 1$ and $(\lambda_1^2\lambda_2^2)/l^{2\lambda_3} = 0.000625$ for $l = 2$.

Priors for exogenous constant terms are

$$a_0 \sim N(0,\ 100\hat{\sigma}_1^2)$$

$$b_0 \sim N(0,\ 100\hat{\sigma}_2^2)$$

where $\lambda_1^2\lambda_4^2 = 100$.

The default prior variances for the coefficients of all endogenous variables are rather small. The Minnesota prior essentially assumes that we have two independent time series each representing a univariate random walk:

$$\texttt{ogap} = \texttt{L1.ogap} + \epsilon_1$$

$$\texttt{inflation} = \texttt{L1.inflation} + \epsilon_2$$

The prior variances shrink as the lag l increases as long as λ_3 is positive. Also, cross-variables variances shrink by a factor of λ_2^2 from self-variables variances. All variances are proportional to λ_1^2. If we increase λ_1 from 0.1, the default, to 1, all variances will increase by a factor of 100.

In the specification of the conjugate Minnesota prior, (u_1, u_2) is assumed to follow a bivariate normal with 0 means and an unknown error covariance $\boldsymbol{\Sigma}$.

The prior for $\boldsymbol{\beta}$ is conditional on $\boldsymbol{\Sigma}$. The prior mean stays the same, but the prior covariance matrix Ω_0 is replaced by $\boldsymbol{\Sigma} \otimes \Phi_0$, where Φ_0 has a structure similar to Ω_0 but of dimension 5 instead of 10,

$$\Phi_0 = \text{diag}\left(\frac{1}{\hat{\sigma}_1^2}0.01, \frac{1}{\hat{\sigma}_2^2}0.01, \frac{1}{\hat{\sigma}_1^2}0.0025, \frac{1}{\hat{\sigma}_2^2}0.0025, 100\right)$$

where $\lambda_1^2/l^{2\lambda_3} = \lambda_1^2 = 0.01$ for $l = 1$, $\lambda_1^2/l^{2\lambda_3} = 0.0025$ for $l = 2$, and $\lambda_1^2\lambda_4^2 = 100$.

In this case, the prior assumption on $\boldsymbol{\beta}$ implies that the multivariate process consists of two dependent random walks.

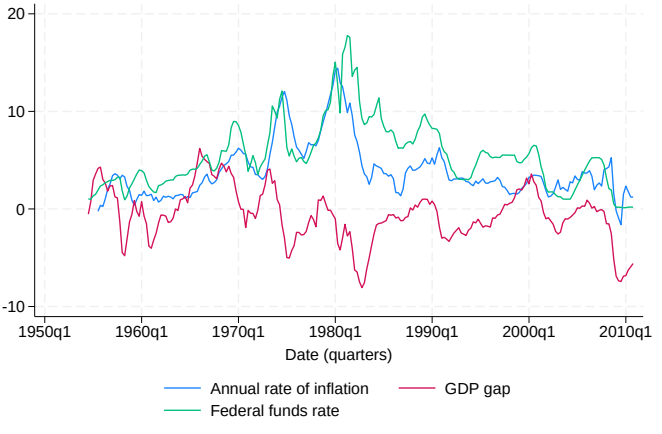
Error covariance $\boldsymbol{\Sigma}$ is assigned an inverse-Wishart prior with default degrees of freedom $K + 2 = 4$, and the default scale matrix S_0 is a diagonal matrix formed by the AR variance estimates. The effect of this prior can be interpreted as a lack of contemporaneous correlation among the error terms.

◁

▷ Example 6: Choosing the number of lags of a VAR model

Consider `usmacro.dta`. Let's look at time series of the three dependent variables.

```
. use https://www.stata-press.com/data/r19/usmacro
(Federal Reserve Economic Data - St. Louis Fed)
. tsline inflation ogap fedfunds
```



Time-series plots suggest a relationship between the three dependent variables that we would like to explore using a Bayesian VAR model.

Our goal is to model the dynamics of the three time series using VAR. We will use the `bayes: var` command to fit a Bayesian VAR model with the default conjugate Minnesota prior for the regression coefficients and error covariance. We will use all observations before the 1st quarter of 2004 to fit the model and leave out the later observations to test the forecasting ability of the model.

An important consideration in specifying the model is choosing the maximum number of lags. An expert in the field may have an optimal choice based on theoretical or empirical knowledge, but for us, it is not immediately clear whether we should use 2, 4, or more lags. In a classical setting, one can use the `varsoc` command to choose the maximum lag. It is not uncommon for `varsoc` to suggest too large of a lag length. For example, if we run `varsoc` on our data using up to 12 lags,

```
. varsoc inflation ogap fedfunds, maxlag(12)
Lag-order selection criteria
   Sample: 1958q3 thru 2010q4                              Number of obs = 210
```

Lag	LL	LR	df	p	FPE	AIC	HQIC	SBIC
0	-1488.6				296.509	14.2057	14.225	14.2535
1	-723.715	1529.8	9	0.000	.221616	7.00681	7.08413	7.19807
2	-689.089	69.252	9	0.000	.173634	6.76275	6.89806	7.09746*
3	-673.171	31.836	9	0.000	.162585	6.69686	6.89017	7.17502
4	-661.806	22.729	9	0.007	.159006	6.67434	6.92564	7.29595
5	-639.015	45.583	9	0.000	.139492	6.543	6.85228	7.30805
6	-619.85	38.329	9	0.000	.126698	6.44619	6.81346*	7.35469
7	-615.967	7.7663	9	0.558	.133135	6.49492	6.92019	7.54687
8	-610.886	10.161	9	0.338	.138349	6.53225	7.0155	7.72765
9	-587.182	47.409	9	0.000	.120437	6.39221	6.93345	7.73105
10	-581.902	10.559	9	0.307	.124996	6.42764	7.02688	7.90993
11	-567.442	28.921*	9	0.001	.118912*	6.37564*	7.03286	8.00137
12	-565.064	4.7562	9	0.855	.126973	6.4387	7.15392	8.20789

```
* optimal lag
Endogenous: inflation ogap fedfunds
 Exogenous: _cons
```

the AIC criterion suggests a maximum lag of 11. A VAR model with 11 lags for our data will have 102 coefficients, which is likely too many given the sample size of 190. The resulting imprecision in the estimates would lead to wide forecast intervals.

From a Bayesian viewpoint, an optimal way to solve this problem is to use Bayesian model comparison. First, we choose a reasonable set of possible lags, $1, \ldots, p_{\max}$. Then, for each lag p, we fit a Bayesian VAR(p) model. Finally, we compare the fitted models using their log-marginal likelihoods. Except the number of lags, all other model specifications, including the choice of priors, stay the same.

In this example, we consider six possible VAR models with lags ranging from 1 to 6. We specify two options with `bayes: var`: `rseed(17)`, for reproducibility, and `saving()` to save the simulation results. The latter is required by `estimates store` to store Bayesian model estimation results. We run the models quietly to suppress lengthy estimation output.

```
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/1)
. estimates store bvar1
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/2)
. estimates store bvar2
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/3)
. estimates store bvar3
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/4)
. estimates store bvar4
```

```
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/5)
. estimates store bvar5
. quietly bayes, rseed(17) saving(bvarsim,replace):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/6)
. estimates store bvar6
```

We compare the models using the `bayestest model` command. All six models are assumed equally probable a priori, as can be seen from the second column of the output table. The third column shows posterior model probabilities; the model with the highest probability is the best.

```
. bayestest model bvar1 bvar2 bvar3 bvar4 bvar5 bvar6
Bayesian model tests
```

	log(ML)	P(M)	P(M|y)
bvar1	-690.7037	0.1667	0.0000
bvar2	-680.1811	0.1667	0.0000
bvar3	-674.5212	0.1667	0.0065
bvar4	-670.3258	0.1667	0.4313
bvar5	-670.7045	0.1667	0.2953
bvar6	-670.8059	0.1667	0.2669

```
Note: Marginal likelihood (ML) is computed using
      Laplace-Metropolis approximation.
```

The model with four lags has the highest posterior probability, 0.43, and thus four is our choice for the number of lags. Incidentally or not, four lags corresponds to a period of one year.

◁

▷ Example 7: Bayesian VAR(4) model estimation

Continuing with example 6, we proceed with Bayesian estimation of the chosen VAR(4) model. We rerun the model but this time showing the MCMC summary and output tables. The model summary is suppressed for brevity, but as we mentioned in example 6, we use the default conjugate Minnesota prior.

```
. bayes, nomodelsummary rseed(17):
> var inflation ogap fedfunds if date < tq(2004q1), lags(1/4)

Burn-in ...
Simulation ...

Bayesian vector autoregression                  MCMC iterations  =     12,500
Gibbs sampling                                  Burn-in          =      2,500
                                                MCMC sample size =     10,000
Sample: 1956q3 thru 2003q4                      Number of obs    =        190
                                                Acceptance rate  =          1
                                                Efficiency:  min =      .9322
                                                             avg =       .993
Log marginal-likelihood = -670.32584                         max =          1
```

	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred. interval]	
inflation						
inflation						
L1.	1.107465	.0422849	.000423	1.106848	1.02544	1.192476
L2.	-.064825	.0417594	.000418	-.064536	-.1470882	.0176208
L3.	-.0358872	.0290815	.000291	-.0359867	-.092745	.0210088
L4.	-.0397985	.0215853	.000216	-.0397207	-.0821996	.002274
ogap						
L1.	.0646785	.0294384	.000305	.0644936	.0070243	.1229662
L2.	.0071294	.0267595	.000268	.0072498	-.0444994	.058461
L3.	-.002015	.0187035	.000192	-.0021291	-.038934	.0346847
L4.	-.0088532	.0142951	.000141	-.0089083	-.0366927	.0193774
fedfunds						
L1.	.0770026	.027543	.000275	.076643	.0237776	.1315991
L2.	-.0351476	.0243814	.000244	-.0351349	-.0831241	.0124089
L3.	-.0151671	.0173423	.000173	-.0154901	-.0487873	.0193082
L4.	-.0190271	.0134133	.000134	-.0191324	-.0456025	.0072003
_cons	.1225433	.0832813	.000833	.1225758	-.0433392	.2853939
ogap						
inflation						
L1.	-.068909	.0627925	.000628	-.0683572	-.1934463	.0524915
L2.	.0073091	.0617798	.000609	.0066414	-.1153963	.1276874
L3.	.0098226	.0437754	.000438	.0105327	-.0773841	.0942487
L4.	.0146217	.0325626	.000326	.0147658	-.0498018	.0778098
ogap						
L1.	1.030706	.0443351	.000436	1.030381	.9445329	1.117702
L2.	-.0533506	.0405626	.000406	-.0536868	-.1331328	.0269409
L3.	-.0463432	.028635	.000286	-.0468054	-.1021503	.0103083
L4.	-.0243524	.0215736	.000216	-.0246339	-.0671305	.0178628
fedfunds						
L1.	-.0080148	.0410321	.000404	-.0079538	-.0897528	.0726622
L2.	-.0513393	.0362004	.000362	-.0514847	-.1208578	.0196766
L3.	.0096443	.0264572	.000265	.0092495	-.0416928	.0618986
L4.	.0028706	.0200856	.000201	.002678	-.0362012	.0424353
_cons	.3851112	.1261445	.001261	.3836084	.1334414	.6333448

fedfunds						
inflation						
L1.	.0568126	.0719825	.00072	.0563617	-.0829406	.2008528
L2.	.0568927	.0706982	.000699	.0569514	-.0811303	.1967728
L3.	-.0026048	.0495878	.000474	-.0023296	-.1001453	.09392
L4.	-.0159998	.0369476	.000375	-.0163655	-.0877556	.0563861
ogap						
L1.	.1873653	.0495204	.000498	.1873384	.0899816	.2850046
L2.	-.0544593	.045749	.000465	-.055174	-.1438389	.035413
L3.	-.0485134	.0324919	.000335	-.048869	-.1120501	.0148947
L4.	-.0327431	.0245286	.000245	-.0324051	-.0807114	.0156984
fedfunds						
L1.	.9623752	.0472236	.000472	.9622282	.8696618	1.054146
L2.	-.0728725	.0414158	.000414	-.0731102	-.1538312	.0082934
L3.	.0146377	.0293537	.000294	.0143481	-.0419335	.072309
L4.	.0018861	.0228329	.000228	.0021041	-.0430797	.0462406
_cons	.1931161	.1433129	.001433	.1950842	-.0912408	.4717853
Sigma_1_1	.2873009	.0293728	.000297	.2853721	.2349716	.3493519
Sigma_2_1	.0281781	.0315254	.000315	.0276486	-.0345647	.0912571
Sigma_3_1	.1480748	.0372496	.000372	.1468518	.0777631	.2251876
Sigma_2_2	.6575456	.0671182	.000684	.6530136	.5395734	.8029292
Sigma_3_2	.2398338	.0559347	.000559	.238127	.1357633	.3561841
Sigma_3_3	.8371554	.0857785	.000858	.8298623	.6868505	1.024522

The Gibbs sampling used to simulate the posterior distribution has high efficiency of 99% on average and the perfect acceptance rate of 1. There is no indication of convergence problems.

There are 39 regression coefficients in the model, which would be difficult to interpret directly. The posterior estimates for the error covariance matrix {Sigma,m} suggest a positive correlation between fedfunds and inflation and fedfunds and ogdp; see the estimates for {Sigma_3_1} and {Sigma_3_2}.

Because we did not use the saving() option with bayes: var, the simulation results are saved in a temporary dataset. If you plan to use one of the postestimation commands such as bayesirf or bayesfcast, you need to save the simulation results to a permanent dataset. We can do this by using the saving() option on replay.

```
. bayes, saving(bvarex2)
note: file bvarex2.dta saved.
```

Before continuing with postestimation analysis, let's check the stability condition of the model using the `bayesvarstable` command.

```
. bayesvarstable
Eigenvalue stability condition                  Companion matrix size =    12
                                                MCMC sample size      = 10000
```

Eigenvalue modulus	Mean	Std. dev.	MCSE	Median	Equal-tailed [95% cred.	interval]
1	.9473457	.0199198	.000199	.9481282	.9057116	.9838371
2	.9417123	.0257058	.000257	.9453142	.877582	.9811621
3	.8184194	.0716288	.000716	.8274233	.6763741	.9322606
4	.5930213	.0930861	.000931	.5836551	.4256008	.7733104
5	.4859573	.0896516	.000897	.4866775	.330644	.6554575
6	.3659255	.0417669	.000418	.3635287	.291461	.459251
7	.3499339	.0365851	.000366	.3496959	.2767796	.4214287
8	.3155561	.0383687	.000384	.3173136	.2348504	.3856269
9	.3014183	.0396995	.000397	.3038818	.2177103	.3736035
10	.2670156	.0479518	.00048	.2717858	.1582521	.3475958
11	.2361436	.0556598	.000557	.2414199	.1135724	.329785
12	.1887299	.0805818	.000806	.2036124	.0151749	.3102756

```
Pr(eigenvalues lie inside the unit circle) = 0.9977
```

The command reports that the companion matrix of our model is of size 12 (three response variables times four lags) and thus reports posterior summaries for the moduli of 12 eigenvalues. The posterior probability that all eigenvalues lie in the unit circle is estimated to be essentially 1, so the stability condition is satisfied.

◁

The main postestimation tools for interpreting VAR models are IRFs and forecasting, which we illustrate in the following examples.

▷ Example 8: IRFs

IRFs are commonly used to summarize a VAR model. IRFs measure the effect of a shock in one variable, also called an impulse variable, on a given response variable. The effect of the shock on the response variable is traced out over a predefined number of future steps. We compute a number of IRF statistics associated with our model using the `bayesirf` command, whose syntax is similar to the frequentist `irf` command. For computational details, see *Methods and formulas* of [BAYES] **bayesirf create**.

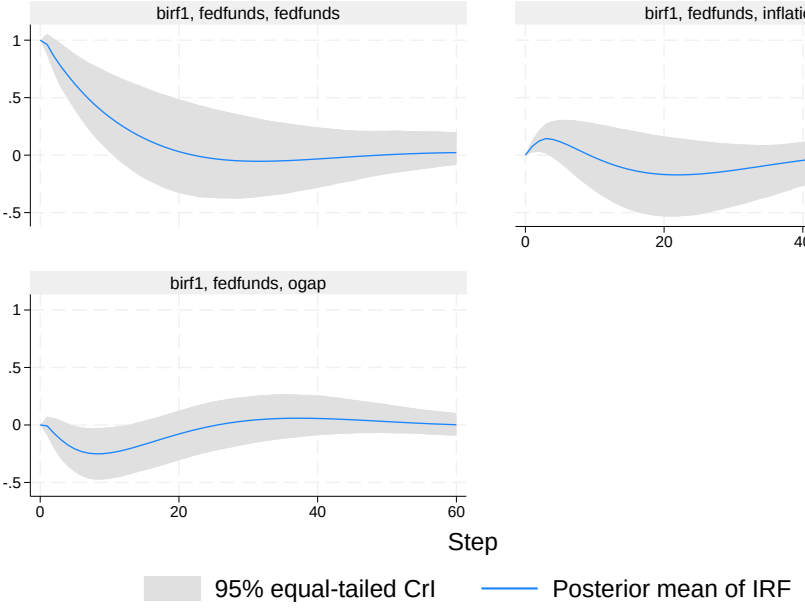
The `bayesirf create` command computes IRF results and stores them in a dataset with a special structure and with the `.irf` extension. One `.irf` dataset may contain several sets of IRF results.

Continuing with example 7, let's compute the effect of shocks for up to 15 years (60 quarter periods) into the future. We name the set of results `birf1` and save them in `birfex2.irf`.

```
. bayesirf create birf1, step(60) set(birfex2)
(file birfex2.irf created)
(file birfex2.irf now active)
(file birfex2.irf updated)
```

It is easier to visualize the effect of a shock in one response variable on all other response variables and itself by using the `bayesirf graph` command. The command draws the posterior mean estimates of IRF coefficients along with 95% CrIs. Let's inspect the effect of shock on `fedfunds`. Shocks of interests are specified using the `impulse()` option.

```
. bayesirf graph irf, impulse(fedfunds)
```



Graphs by irfname, impulse variable, and response variable

IRFs are obtained by setting the error vector in the likelihood model to (0,0,1) (1 for `fedfunds` and 0 otherwise) at step 0 and propagating this unit vector in time according to the VAR equations. For example, the response of `inflation` (second graph) starts from 0 at step 0, slightly increases during the first year, then slowly decreases during the next 4 years, and finally converges to a small positive value at the end of our 15-year period. According to the third graph, after a monetary shock from the Federal Reserve, the output gap decreases during the first two years, then slowly increases for the following eight years, and finally stabilizes at a small positive value. Notably, the shock effect on all response variables reach equilibrium after about 12 years.

We can examine IRF coefficients in more detail by listing them in a table using `bayesirf table`. For example, let's inspect how the output gap is responding to a shock in federal funds in the first two years. This particular choice is made using the `response()`, `impulse()`, and `step()` options.

```
. bayesirf table irf, response(ogap) impulse(fedfunds) step(7)
Results from birf1
```

Step	(1) irf	(1) Lower	(1) Upper
0	0	0	0
1	-.008015	-.089753	.072662
2	-.072428	-.205354	.059264
3	-.128667	-.296316	.039592
4	-.174391	-.361456	.009988
5	-.208873	-.409928	-.009742
6	-.232076	-.444489	-.021792
7	-.245563	-.466458	-.028681

```
Posterior means reported.
95% equal-tailed credible lower and upper bounds reported.
(1) irfname = birf1, impulse = fedfunds, and response = ogap.
```

The `bayesirf table` command reports posterior mean estimates (first column), lower 95% credible limits (second column), and upper 95% credible limits (third column). We see that a 1% increase in `fedfunds` leads to about a 0.01 units decrease in `ogap` after 1 quarter and to 0.25 units decrease in `ogap` after 8 quarters (2 years). That is, in the short term, an increase in federal spending increases the gap between real and potential GDP.

The regular IRF functions do not account for the fact that the shocks in different impulse variables are generally not independent. For example, in our case, shocks in federal funds and inflation are likely dependent. A better representation of the dynamics between variables is provided by the so-called orthogonalized IRFs (OIRFs), referred to as `oirf` in `bayesirf` commands. The latter depends on the preset causal ordering of the impulse variables, as specified using the `order()` option. The default order is the order in which the variables are listed in the `bayes: var` command specification.

For example, let's examine the following causal order: `inflation` → `fedfunds` → `ogap`. In other words, let's assume that `fedfunds` has no immediate effect on `inflation` and that `ogap` has no immediate effect on `inflation` and `fedfunds`.

```
. bayesirf create birf2, step(60) set(birfex2) order(inflation fedfunds ogap)
(file birfex2.irf now active)
(file birfex2.irf updated)
```

The new IRF statistics are saved as `birf2` in `birfex2.irf`.

We can now summarize `oirf` statistics of `ogap` response to impulse in the third equation, referred to as `fedfunds`, corresponding to the new order we have specified.

```
. bayesirf table oirf, irf(birf2) response(ogap) impulse(fedfunds) step(7)
Results from birf2
```

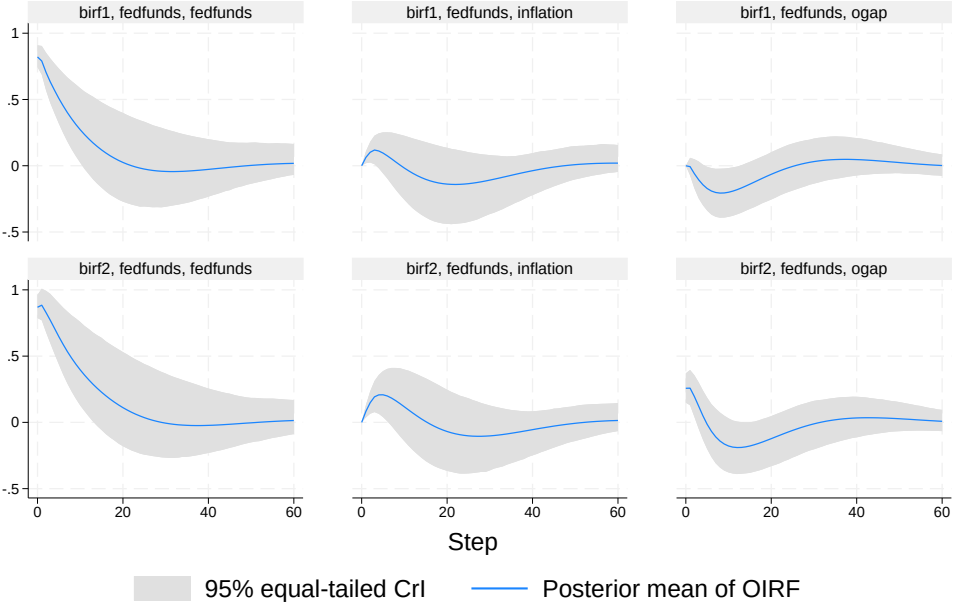
Step	(1) oirf	(1) Lower	(1) Upper
0	.257325	.148406	.370283
1	.258308	.128963	.395361
2	.195725	.041816	.35614
3	.123306	-.047116	.30244
4	.050238	-.130222	.241047
5	-.014137	-.201699	.186256
6	-.067159	-.261451	.13775
7	-.109004	-.309117	.103761

```
Posterior means reported.
95% equal-tailed credible lower and upper bounds reported.
(1) irfname = birf2, impulse = fedfunds, and response = ogap.
```

We observe notable differences between `oirf` and `irf` estimates reported above. A shock in the `fedfunds` equation now starts at 0.26 at step 0 and initially has little effect on closing the positive output gap, but after 4 steps (about a year), `ogap` becomes negative. As in the case with `irf` values, we see that a shock in federal funds has a negative effect on the output gap in a short term. OIRFs have the benefit of accounting for the correlation between `inflation` and `fedfunds`.

The interpretation of OIRFs very much depends on the causal order of response variables. Choosing an order can be difficult when there is no obvious choice based on expert knowledge. Next, for easier comparison, we show how to use `bayesirf graph` to plot the OIRFs from both `birf1` and `birf2`, which differ only in the response variable ordering.

```
. bayesirf graph oirf, impulse(fedfunds)
```

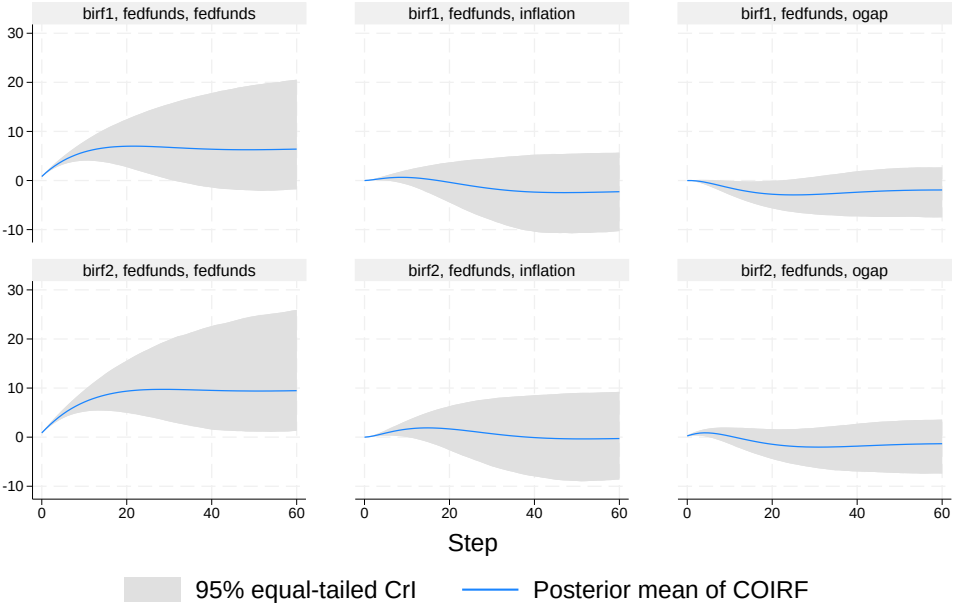


Graphs by irfname, impulse variable, and response variable

The first row shows OIRFs for the original order, `inflation` → `ogap` → `fedfunds`, and the second row shows results for the new order, `inflation` → `fedfunds` → `ogap`. As we remarked above, there are differences between the OIRF results corresponding to different orderings.

Another way to follow the dynamics in ogap is to inspect the cumulative OIRF, coirf. Cumulative IRF statistics accumulate the shock effects over time. The following graph compares the cumulative OIRFs of birf1 and birf2 to a shock in fedfunds.

```
. bayesirf graph coirf, impulse(fedfunds)
```



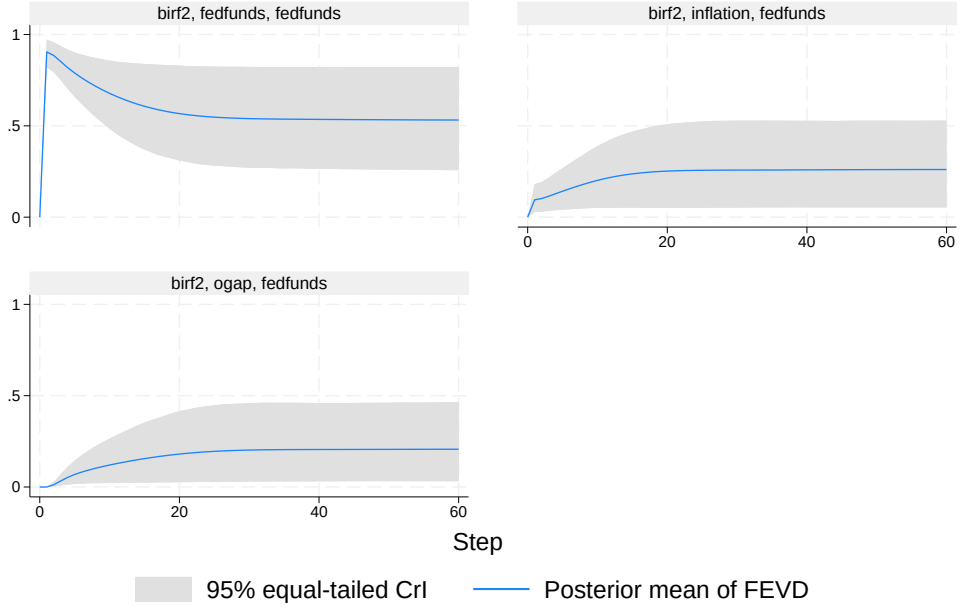
Graphs by irfname, impulse variable, and response variable

All two shock effects reach equilibrium after about 10 years. In the long term, a monetary shock reduces inflation and decreases the output gap.

Another set of IRFs that are useful for interpreting VAR models is the forecast error variance decompositions, or FEVDs. FEVDs measure the contribution, in terms of variability, of impulse variables to the forecast error in response variables. FEVDs, similar to OIRFs, depend on the causal ordering of the response variables.

For illustration, let's inspect the FEVDs of the response variable `fedfunds` for the `birf2` results corresponding to the `inflation` → `fedfunds` → `ogap` order. First, we show FEVD graphs.

```
. bayesirf graph fevd, irf(birf2) response(fedfunds)
```



Graphs by irfname, impulse variable, and response variable

In the long term, half the forecast error of `fedfunds` is contributed by `fedfunds` itself, whereas `inflation` and `ogap` contribute by a quarter each.

A table of FEVD estimates gives us more details.

```
. bayesirf table fevd, irf(birf2) response(fedfunds) step(7)
Results from birf2
```

Step	(1) fevd	(1) Lower	(1) Upper
0	0	0	0
1	.095083	.027875	.180163
2	.102093	.029869	.192633
3	.114495	.034531	.216095
4	.128495	.038329	.242878
5	.142093	.041094	.268065
6	.155334	.043475	.293326
7	.16808	.045986	.318506

Step	(2) fevd	(2) Lower	(2) Upper
0	0	0	0
1	.904917	.819837	.972125
2	.885277	.794215	.957829
3	.852453	.751346	.936377
4	.818721	.702946	.917391
5	.789353	.659076	.902321
6	.763024	.619474	.890185
7	.739026	.582641	.879954

Step	(3) fevd	(3) Lower	(3) Upper
0	0	0	0
1	0	0	0
2	.01263	.002624	.027988
3	.033052	.00823	.069683
4	.052784	.013287	.111021
5	.068554	.016546	.144177
6	.081642	.018741	.174098
7	.092895	.020159	.199897

```
Posterior means reported.
95% equal-tailed credible lower and upper bounds reported.
(1) irfname = birf2, impulse = inflation, and response = fedfunds.
(2) irfname = birf2, impulse = fedfunds, and response = fedfunds.
(3) irfname = birf2, impulse = ogap, and response = fedfunds.
```

The command output contains three tables, one for each impulse. At step 1, which corresponds to one-step-ahead predictions, FEVD posterior mean estimates are about 0.095 for `inflation`, 0.905 for `fedfunds`, and 0 for `ogap` due to the imposed order. The sum of FEVDs across impulses is 1. Most of the forecast error in `fedfunds` is because of the variability in `fedfunds` itself. At step 8, however, FEVD estimates become 0.18 for `inflation`, 0.72 for `fedfunds`, and 0.10 for `ogap`. The predominant effect of its own variability in FEVD estimates is an indirect effect of the Minnesota prior that shrinks self-variables first-lag coefficients to 1 and all others to 0.

◁

▷ Example 9: Forecasting

Bayesian dynamic forecasting is a special case of Bayesian predictions that uses posterior predictive distributions conditional on time to predict a response variable at multiple steps into the future; see *Methods and formulas* of [BAYES] **bayesfcast compute**.

`bayesfcast compute` is the Bayesian counterpart of the [TS] **fcast compute** command, which is used for Bayesian forecasting after the `bayes: var` command.

Let's compute dynamic forecasts starting with the first quarter of 2004 until the end of the observed time frame, or 28 quarter periods ahead.

```
. bayesfcast compute b_, step(28) dynamic(tq(2004q1))
```

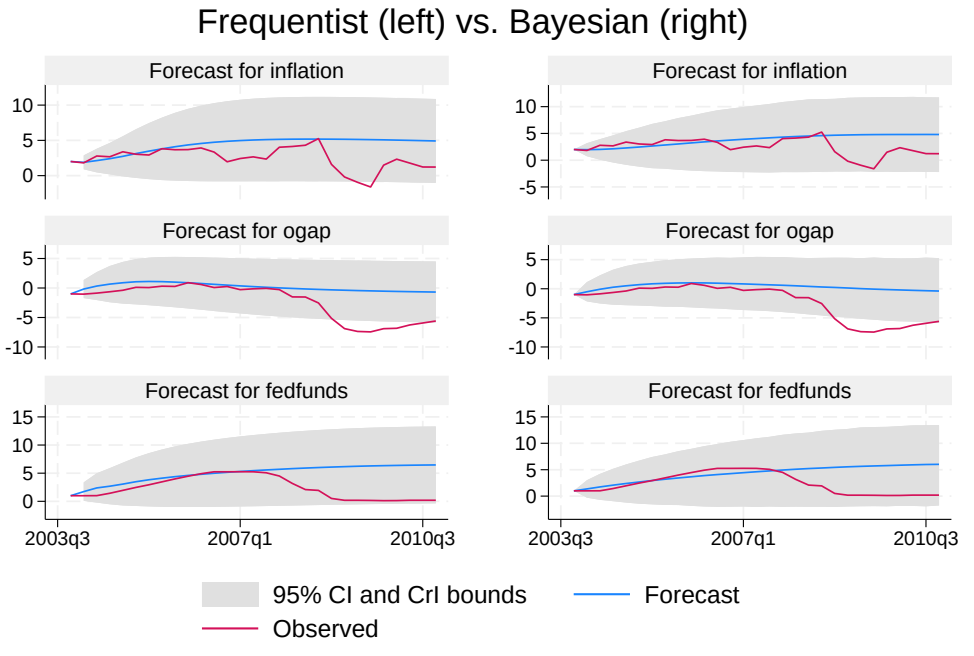
By default, `bayesfcast compute` computes and saves in the current dataset the posterior mean estimates of the predicted response variables along with the 95% credible intervals. The new variables are prefixed with `b_`.

It would be interesting to compare the Bayesian forecast results with the frequentist ones obtained by `fcast compute` after fitting the `var` command on the same model.

```
. quietly var inflation ogap fedfunds if date < tq(2004q1), lags(1/4)
. fcast compute f_, step(28) dynamic(tq(2004q1))
```

We can use the bayesfcast graph command to plot the observed and forecasted values along with their 95% CIs for the frequentist and 95% CrIs for the Bayesian results. Frequentist results are on the left, and Bayesian results are on the right.

```
. bayesfcast graph f_inflation b_inflation f_ogap b_ogap f_fedfunds b_fedfunds,
> observed byopts(rows(3) title("Frequentist (left) vs. Bayesian (right)"))
> legend(label(1 "95% CI and CrI bounds"))
```



In the forecast period before 2008, the Bayesian forecasts seem to fit the observed response variables slightly better. The 95% CrIs include the observed values most of the time, except for ogap at the second half of 2008 during the great recession. We should not expect a VAR model to forecast extreme events such as recessions.

◁

▷ Example 10: One-step-ahead Bayesian predictions

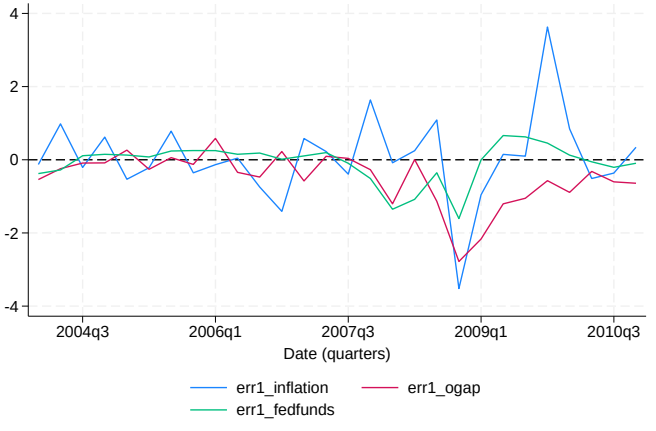
One-step-ahead Bayesian predictions are a special case of Bayesian forecasts, where current observed responses are used to make predictions for the next time period. In contrast to dynamic predictions, one-step-ahead predictions can be computed using the general postestimation command for Bayesian predictions, bayespredict.

For illustration, let's compute predicted posterior means for the three responses starting with the first quarter of 2004 and save the results as new variables `pr1_inflation`, `pr1_ogap`, and `pr1_fedfunds`.

```
. bayespredict pr1_inflation pr1_ogap pr1_fedfunds if date>=tq(2004q1), mean
Computing predictions ...
```

Because these are one-step-ahead predictions, we expect them to be fairly close to the observed responses, much more so than dynamic forecasts with multiple prediction steps ahead. To verify that, let's look at the prediction errors computed as the difference between the observed and predicted responses and plot them as time series.

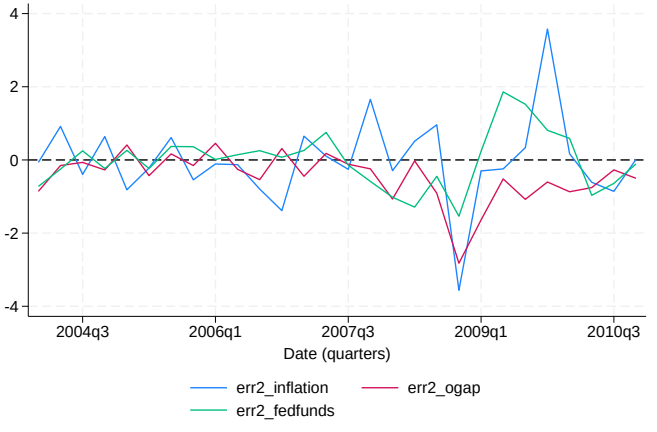
```
. generate err1_inflation = inflation - pr1_inflation
(198 missing values generated)
. generate err1_ogap = ogap - pr1_ogap
(198 missing values generated)
. generate err1_fedfunds = fedfunds - pr1_fedfunds
(198 missing values generated)
. tsline err1_inflation err1_ogap err1_fedfunds if date>=tq(2004q1), yline(0)
```



Recall that the measurement units in this example are percentage growth rates for inflation and federal funds and percentage deviation from trend for the output gap. We see that one-step-ahead predictions perform well right before the beginning of the great recession in 2008, within a margin of 1 unit; that is, prediction errors are within 1 percentage point of realized values. After that, all three errors become negative for a period of time, which means they commit overprediction, but then stabilize again at the end of 2009. The predictions for `inflation` are particularly off, overpredicting before the second quarter of 2009 and underpredicting after that until the end of 2009. A logical conclusion is that our model, fit on the data before the great recession, cannot capture the macroeconomic disruption of 2008 and 2009.

Finally, for those of you interested in comparing Bayesian and classical one-step-ahead predictions, we show the computation for the latter.

```
. quietly var inflation ogap fedfunds if date < tq(2004q1), lags(1/4)
. predict pr2_inflation, equation(inflation)
(option xb assumed; fitted values)
(8 missing values generated)
. predict pr2_ogap, equation(ogap)
(option xb assumed; fitted values)
(8 missing values generated)
. predict pr2_fedfunds, equation(fedfunds)
(option xb assumed; fitted values)
(8 missing values generated)
. generate err2_inflation = inflation - pr2_inflation
(8 missing values generated)
. generate err2_ogap = ogap - pr2_ogap
(8 missing values generated)
. generate err2_fedfunds = fedfunds - pr2_fedfunds
(8 missing values generated)
. tsline err2_inflation err2_ogap err2_fedfunds if date>=tq(2004q1), yline(0)
```



Bayesian and classical one-step-ahead predictions are similar, both failing to follow the dynamics of 2008–2009. However, Bayesian predictions for federal funds appear to be more precise.

◁

Stored results

See *Stored results* in [BAYES] **bayes**. In addition, `bayes: var` stores the following in `e()`:

Scalars

`e(tmin)`	first time period in sample
`e(tmax)`	maximum time
`e(mlag)`	highest lag in VAR
`e(selftight)`	self-variables tightness parameter in Minnesota priors
`e(crosstight)`	cross-variables tightness parameter in Minnesota priors (not with conjugate Minnesota prior)
`e(lagdecay)`	lag-decay parameter in Minnesota priors
`e(exogtight)`	exogenous variable tightness parameter in Minnesota priors
`e(dfcov)`	degrees of freedom of inverse-Wishart prior

Macros

`e(cmdname)`	`var`
`e(prefix)`	`bayes`
`e(command)`	`var` command specification
`e(varprior)`	prior model for VAR coefficients and error covariance
`e(endog)`	names of endogenous variables
`e(exog)`	names of exogenous variables, and their lags, if specified
`e(exogvars)`	names of exogenous variables, if specified
`e(lags)`	lags in model
`e(exlags)`	lags of exogenous variables in model, if specified
`e(timevar)`	time variable specified in `tsset`
`e(tsfmt)`	format for the current time variable

Matrices

`e(exlagsm)`	matrix mapping lags to exogenous variables (with `exog()`)
`e(phi)`	covariance product matrix Φ_0 for conjugate Minnesota prior
`e(arcov)`	AR covariance matrix (with `arcov`)
`e(varcov)`	VAR covariance matrix (with `varcov`)
`e(mvnmean)`	mean vector of MVN prior
`e(mvncov)`	covariance matrix of MVN prior (with `mvniwishprior()` or `mvnjeffprior()`)
`e(scalecov)`	scale matrix of inverse-Wishart prior

Methods and formulas

Methods and formulas are presented under the following headings:

VAR model specification
Original Minnesota prior with known (fixed) error covariance
Conjugate Minnesota prior for VAR model with unknown error covariance
MVN-inverse Wishart prior
MVN-diffuse (normal-Jeffreys) prior

VAR model specification

Let $\mathbf{y}_t$ be a $K \times 1$ vector of endogenous (dependent) variables at time t for $t = 1, \ldots, T$ and $\mathbf{x}_t$ be a $m \times 1$ vector of exogenous regressors including the constant terms.

A p-order VAR model, VAR(p), can be defined according to Lütkepohl (2005) as

$$\mathbf{y}_t = \mathbf{A}_1\mathbf{y}_{t-1} + \cdots + \mathbf{A}_p\mathbf{y}_{t-p} + \mathbf{C}\mathbf{x}_t + \mathbf{u}_t,\ \mathbf{u}_t \sim N(\mathbf{0}, \boldsymbol{\Sigma})$$

where p is the number of lags;

$\mathbf{A}_l = (a_{ij}^l)$ are $K \times K$ matrices of unknown endogenous variable lag coefficients ($l = 1, \ldots, p$);

$\mathbf{C} = (c_{is})$ is a $K \times m$ matrix of exogenous variable coefficients; and

$\mathbf{u}_t$ is a $K \times 1$ vector of error terms with a $K \times K$ covariance matrix $\boldsymbol{\Sigma}$.

A VAR(p) model can be written in a more compact form as

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{U}$$

where

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1' \\ \vdots \\ \mathbf{y}_T' \end{pmatrix}, \mathbf{X} = \begin{pmatrix} \mathbf{y}_0' & \mathbf{y}_{-1}' & \cdots & \mathbf{y}_{1-p}' & \mathbf{x}_1' \\ & & \vdots & & \\ \mathbf{y}_{T-1}' & \mathbf{y}_{T-2}' & \cdots & \mathbf{y}_{T-p}' & \mathbf{x}_T' \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{A}_1' \\ \vdots \\ \mathbf{A}_p' \\ \mathbf{C}' \end{pmatrix}, \mathbf{U} = \begin{pmatrix} \mathbf{u}_1' \\ \vdots \\ \mathbf{u}_T' \end{pmatrix}$$

$\mathbf{Y}$ is a $T \times k$ matrix, $\mathbf{X}$ is a $T \times (Kp + m)$ matrix, $\mathbf{B}$ is a $(Kp + m) \times K$ matrix of all coefficients, and $\mathbf{U}$ is a $T \times K$ matrix.

The OLS estimates of $\mathbf{B}$ and $\boldsymbol{\Sigma}$ are

$$\widehat{\mathbf{B}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

$$\widehat{\boldsymbol{\Sigma}}_{\text{OLS}} = \widehat{\mathbf{U}}'\widehat{\mathbf{U}}/(T - Kp - m - 1),\ \widehat{\mathbf{U}} = \mathbf{Y} - \mathbf{X}\widehat{\mathbf{B}} \tag{1}$$

Vectorizing the above matrix equation, we obtain

$$\mathbf{y} = \mathbf{X}^\star\boldsymbol{\beta} + \mathbf{u}$$

where $\mathbf{y} = \text{vec}(\mathbf{Y})$ is $KT \times 1$ vector, $\mathbf{X}^\star = I_K \otimes \mathbf{X}$ is a $KT \times K(Kp + m)$ matrix ($\otimes$ is the Kronecker product and I_K is a $K \times K$ identity matrix), $\boldsymbol{\beta} = \text{vec}(\mathbf{B})$ is a $K(Kp + m) \times 1$ vector of all coefficients, and $\mathbf{u} = \text{vec}(\mathbf{U})$ is a $KT \times 1$ error vector with a $KT \times KT$ covariance matrix $\boldsymbol{\Sigma}^\star = \boldsymbol{\Sigma} \otimes I_T$.

An essential component of every Bayesian VAR model is specifying a suitable prior for the vector of coefficients $\boldsymbol{\beta}$. In what follows, we will describe several commonly used priors, all based on a so-called Minnesota prior. But before we continue, let's define components that are used by all of these priors.

Consider a univariate AR(p) model for each outcome $k = 1, \ldots, K$,

$$y_{k,t} = a_1 y_{k,t-1} + \cdots + a_p y_{k,t-p} + a_0 + e_{k,t} \tag{2}$$

where $e_{k,t} \sim N(0, \sigma_k^2)$. All priors considered below use the OLS estimate, $\hat{\sigma}_k^2$, of σ_k^2. Some of the priors also use a diagonal covariance estimate formed by K error-variance estimates from separate AR(p) models: $\widehat{\mathbf{\Sigma}}_{\text{diag}} = \text{diag}(\hat{\sigma}_1^2, \ldots, \hat{\sigma}_K^2)$.

Original Minnesota prior with known (fixed) error covariance

The original Bayesian VAR model with a Minnesota prior (Litterman 1980, 1986) assumes that the covariance matrix of the error vectors $\mathbf{u}_t$ is known, $\mathbf{\Sigma} = \mathbf{\Sigma}_0$; that is,

$$\mathbf{u} \sim N(\mathbf{0}, \mathbf{\Sigma}_0 \otimes I_T)$$

The original formulation (suboption `arcov` in *minnopts*) used a diagonal matrix with the estimated error variances from K separate AR models (2) on the diagonal as a covariance estimate, $\mathbf{\Sigma}_0 = \widehat{\mathbf{\Sigma}}_{\text{diag}} = \text{diag}(\hat{\sigma}_1^2, \ldots, \hat{\sigma}_K^2)$. Litterman thus proposed to estimate the VAR model equation by equation, rather than as a system of equations, to reduce the computational burden, which at the time was a serious problem. Another formulation (suboption `varcov` in *minnopts*) used the OLS estimate of the covariance matrix from the VAR model, $\mathbf{\Sigma}_0 = \widehat{\mathbf{\Sigma}}_{\text{OLS}}$, defined in (1).

This prior is implemented by the `minnfixedcovprior` option of `bayes: var`, but it is not the default prior. The default prior is the conjugate Minnesota prior (option `minnconjprior`) described in the next section, which uses a less restrictive prior covariance. But we describe the original Minnesota prior first because the conjugate Minnesota prior is its extension.

The Minnesota prior for coefficient vector $\boldsymbol{\beta}$ is an MVN prior,

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \mathbf{\Omega}_0)$$

where a $KT \times 1$ vector $\boldsymbol{\beta}_0$ and a $KT \times KT$ matrix $\mathbf{\Omega}_0$ are defined in a way that accounts for the special time-series structure of the VAR(p) model, which we describe next.

The regression vector $\boldsymbol{\beta}$ is formed by the endogenous variable lag coefficients a_{ij}^l ($l = 1, \ldots, p$ and $i, j = 1, \ldots, K$) and exogenous variable coefficients c_{is} ($i = 1, \ldots, K$ and $s = 1, \ldots, m$). The Minnesota priors assumes that expected values for all coefficients are zero, except for the self-variable first-lag coefficients; that is,

$$E(a_{ij}^l) = \delta_{1l}\delta_{ij} \text{ and } E(c_{is}) = 0$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, so the prior mean vector $\boldsymbol{\beta}_0$ is a $K(Kp + m) \times 1$ vector of 0s and 1s, with 1s corresponding to the self-variable first-lag coefficients.

The original Minnesota prior assumes that there is no correlation between the coefficients of $\boldsymbol{\beta}$. The Minnesota covariance $\mathbf{\Omega}_0$ is thus a diagonal matrix, its diagonal formed by the prior variances $\sigma^2_{a_{ij}^l}$ for the endogenous variable lag coefficients and $\sigma^2_{c_{is}}$ for the exogenous variable coefficients. The prior variances are based on the OLS estimates of error variances, $\hat{\sigma}_k^2$'s, and are defined below.

For endogenous self-variable lag coefficients, the prior variances are

$$\sigma^2_{a_{ii}^l} = \left(\frac{\lambda_1}{l^{\lambda_3}}\right)^2$$

For endogenous cross-variable lag coefficients ($i \neq j$), the prior variances are

$$\sigma^2_{a_{ij}^l} = \left(\frac{\hat{\sigma}_i^2}{\hat{\sigma}_j^2}\right)\left(\frac{\lambda_1\lambda_2}{l^{\lambda_3}}\right)^2$$

For exogenous variable coefficients, the prior variances are

$$\sigma^2_{c_{is}} = \hat{\sigma}^2_i(\lambda_1\lambda_4)^2$$

In the above formulas, λ_1 controls the tightness of the prior variance for self-variable lag coefficients and can be specified in the `selftight()` suboption of the Minnesota prior options, *minnopts*; λ_2 controls the cross-variable lag coefficients spread and can be specified in the `crosstight()` suboption; λ_3 controls the lag attenuation and can be specified in the `lagdecay()` suboption (the higher the lag, the tighter the prior variances); and λ_4 controls the prior variance of the exogenous variable coefficients and can be specified in the `exogtight()` option. Default values for these control parameters are $\lambda_1 = 0.1$, $\lambda_2 = 0.5$, $\lambda_3 = 1$, and $\lambda_4 = 100$.

The prior mean $\boldsymbol{\beta}_0$ and diagonal covariance matrix $\boldsymbol{\Omega}_0$ described above define the original Minnesota prior, which is available by specifying the `minnfixedcovprior` option with `bayes: var`. You can customize this prior by specifying the `minnfixedcovprior(`*fixcovopts*`)` option.

In the model-summary output of `bayes: var`, we refer to the defaults $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\beta}_0$ as `_Sigma0` and `_b0`, respectively. $\boldsymbol{\Omega}_0$ is viewed as a function of `_Sigma0` and prior control parameters λ's.

Conjugate Minnesota prior for VAR model with unknown error covariance

Another framework of Bayesian VAR models assumes that error vectors $\mathbf{u}_t$ have an unknown covariance matrix $\boldsymbol{\Sigma}$. In this case, $\mathbf{u} \sim N(\mathbf{0}, \boldsymbol{\Sigma} \otimes I_T)$.

Karlsson (2013) proposed a prior for $\boldsymbol{\beta}$ with the prior covariance having a similar form to the covariance matrix $\boldsymbol{\Sigma}$. Specifically, the author defined the prior covariance as a product of $\boldsymbol{\Sigma}$ and another covariance matrix, $\boldsymbol{\Phi}_0$, which we refer to as a Minnesota factor covariance,

$$\boldsymbol{\beta} \sim N(\boldsymbol{\beta}_0, \boldsymbol{\Sigma} \otimes \boldsymbol{\Phi}_0)$$

The prior mean vector $\boldsymbol{\beta}_0$ is the same as in the original Minnesota prior, and $\boldsymbol{\Phi}_0$ is a fixed $(Kp + m) \times (Kp + m)$ covariance matrix as defined below.

$\boldsymbol{\Phi}_0 = \text{diag}\left(\left\{\sigma^2_{a^l_j}\right\}^{K,p}_{j=1,l=1}, \left\{\sigma^2_{c_s}\right\}^{m}_{s=1}\right)$ is set to be a diagonal matrix similar to the Minnesota covariance $\boldsymbol{\Omega}_0$ but of a lower dimension $(Kp+m) \times (Kp+m)$ compared with $KT \times KT$. The elements of $\boldsymbol{\Phi}_0$ are defined below for $l = 1, \ldots, p$, $j = 1 \ldots, K$, and $s = 1, \ldots, m$.

For endogenous variable lag coefficients,

$$\sigma^2_{a^l_j} = \left(\frac{1}{\hat{\sigma}^2_j}\right)\left(\frac{\lambda_1}{l^{\lambda 3}}\right)^2$$

And for exogenous variable coefficients,

$$\sigma^2_{c_s} = (\lambda_1\lambda_4)^2$$

In the above formulas, λ_1, λ_3, and λ_4 have the same interpretation as in the original Minnesota prior. In this formulation, λ_2 is not used because there is no distinction between the self- and cross-variables.

The covariance parameter $\boldsymbol{\Sigma}$ has an inverse-Wishart prior with a scale matrix $\mathbf{S}_0$ and degrees of freedom α_0:

$$\boldsymbol{\Sigma} \sim \text{InvWishart}(\alpha_0, \mathbf{S}_0)$$

You can specify α_0 in the `df()` suboption and $\mathbf{S}_0$ in the `scale()` suboption of the `minnconjprior()` option. The default values are $\alpha_0 = K + 2$, the minimum possible value such that the mean exists, and $\mathbf{S}_0 = (\alpha_0 - K - 1)\boldsymbol{\Sigma}_0$, where $\boldsymbol{\Sigma}_0$ is as it is defined for the original Minnesota prior. With these default values, the prior mean of $\boldsymbol{\Sigma}$ is $\boldsymbol{\Sigma}_0$.

The conjugate Minnesota prior is the default prior for `bayes: var`, and it corresponds to the `minnconjprior` option, which is implied by default. You can customize this prior by specifying the `minnconjprior(`*conjopts*`)` option.

In the model-summary output of `bayes: var`, we refer to the defaults $\boldsymbol{\Phi}_0$ as `_Phi0`, $\boldsymbol{\beta}_0$ as `_b0`, and $\mathbf{S}_0$ as `_Scale0`. With degrees of freedom $K + 2$ (default), $\mathbf{S}_0$ is displayed as `_Sigma0`.

MVN-inverse Wishart prior

You can also specify an MVN-inverse Wishart prior for the VAR coefficients and error covariance. This prior also assumes an unknown error covariance $\boldsymbol{\Sigma}$, $\mathbf{u} \sim N(0, \boldsymbol{\Sigma} \otimes I_T)$.

The regression vector $\boldsymbol{\beta}$ has an MVN prior

$$\boldsymbol{\beta} \sim N(\widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\Omega}})$$

with a fixed mean vector $\widetilde{\boldsymbol{\beta}}$ and a covariance matrix $\widetilde{\boldsymbol{\Omega}}$, which can be specified independently.

The covariance parameter $\boldsymbol{\Sigma}$ has an inverse-Wishart prior with a scale matrix $\mathbf{S}_0$ and degrees of freedom α_0,

$$\boldsymbol{\Sigma} \sim \text{InvWishart}(\alpha_0, \mathbf{S}_0)$$

You can specify this prior by using the `minniwishprior(`*iwishopts*`)` option. You can specify $\widetilde{\boldsymbol{\beta}}$ using the `mean()` suboption and $\widetilde{\boldsymbol{\Omega}}$ using the `cov()` suboption. Default values for $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\Omega}}$ are the prior mean and covariance, $\boldsymbol{\beta}_0$ and $\boldsymbol{\Omega}_0$, of the original Minnesota prior.

You can specify α_0 using the `df()` suboption and $\mathbf{S}_0$ using the `scale()` suboption. The default values are $\alpha_0 = K + 2$, the minimum possible value such that the mean exists, and $\mathbf{S}_0 = (\alpha_0 - K - 1)\boldsymbol{\Sigma}_0$.

In the model-summary output of `bayes: var`, we refer to the defaults $\boldsymbol{\beta}_0$ as `_b0`, $\boldsymbol{\Omega}_0$ as `_Omega0`, and $\mathbf{S}_0$ as `_Scale0`. With degrees of freedom $K + 2$ (default), $\mathbf{S}_0$ is displayed as `_Sigma0`.

MVN-diffuse (normal-Jeffreys) prior

Instead of an inverse-Wishart prior for the covariance matrix, as in the previous section, one may consider a diffused (multivariate Jeffreys) prior. As before, $\mathbf{u} \sim N(0, \boldsymbol{\Sigma} \otimes I_T)$.

The prior for $\boldsymbol{\beta}$ is still the MVN prior, as defined in *MVN-inverse Wishart prior*,

$$\boldsymbol{\beta} \sim N(\widetilde{\boldsymbol{\beta}}, \widetilde{\boldsymbol{\Omega}})$$

with a mean vector $\widetilde{\boldsymbol{\beta}}$ and a covariance matrix $\widetilde{\boldsymbol{\Omega}}$. But the covariance matrix $\boldsymbol{\Sigma}$ has a multivariate Jeffreys prior,

$$\pi(\boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{\frac{K+1}{2}}$$

You can specify the `minnjeffprior(`*jeffopts*`)` option for this prior. You can specify $\widetilde{\boldsymbol{\beta}}$ using the `mean()` suboption and $\widetilde{\boldsymbol{\Omega}}$ using the `cov()` suboption. As with MVN-inverse Wishart prior, the default values for $\widetilde{\boldsymbol{\beta}}$ and $\widetilde{\boldsymbol{\Omega}}$ are the prior mean and covariance, $\boldsymbol{\beta}_0$ and $\boldsymbol{\Omega}_0$, of the original Minnesota prior.

In the model-summary output of `bayes: var`, we refer to the defaults β_0 and Ω_0 as `_b0` and `_Omega0`, respectively.

Also see *Methods and formulas* in [BAYES] **bayesmh**.

References

Bańbura, M., D. Giannone, and L. Reichlin. 2008. Large Bayesian VARs. Working Paper Series 966, European Central Bank. https://www.ecb.europa.eu/pub/pdf/scpwps/ecbwp966.pdf.

Dieppe, A., R. Legrand, and B. van Roye. 2016. The BEAR toolbox. Working Paper Series 1934, European Central Bank. https://www.ecb.europa.eu/pub/pdf/scpwps/ecbwp1934.en.pdf.

Doan, T., R. B. Litterman, and C. A. Sims. 1984. Forecasting and conditional projection using realistic prior distributions. *Econometric Reviews* 1: 1–100. https://doi.org/10.1080/07474938408800053.

Kadiyala, K. R., and S. Karlsson. 1997. Numerical methods for estimation and inference in Bayesian VAR-models. *Journal of Applied Econometrics* 12: 99–132. https://doi.org/10.1002/(SICI)1099-1255(199703)12:2<99::AID-JAE429>3.0.CO;2-A.

Karlsson, S. 2013. "Forecasting with Bayesian vector autoregression". In *Handbook of Economic Forecasting*, edited by G. Elliott and A. Timmermann, vol. 2B: 791–897. Amsterdam: North-Holland. https://doi.org/10.1016/B978-0-444-62731-5.00015-4.

Litterman, R. B. 1980. A Bayesian procedure for forecasting with vector autoregressions. Mit working paper, Massachusetts Institute of Technology.

———. 1984. Specifying vector autoregressions for macroeconomic forecasting. Staff Report 92, Federal Reserve Bank of Minneapolis, Research Department. https://lccn.loc.gov/2007702559.

———. 1986. Forecasting with Bayesian vector autoregressions—five years of experience. *Journal of Business and Economic Statistics* 4: 25–38. https://doi.org/10.2307/1391384.

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

Also see

[BAYES] **bayes: var postestimation** — Postestimation tools for bayes: var

[BAYES] **bayes** — Bayesian regression models using the bayes prefix

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

[BAYES] **Bayesian postestimation** — Postestimation tools after Bayesian estimation

[BAYES] **Bayesian estimation** — Bayesian estimation commands

[BAYES] **Bayesian commands** — Introduction to commands for Bayesian analysis

[BAYES] **Intro** — Introduction to Bayesian analysis

[BAYES] **Glossary**