bvarsv: An R implementation of the Primiceri (2005) model for macroeconomic time series

Fabian Krüger, Heidelberg Institute for Theoretical Studies, fabian.krueger@h-its.org

November 26, 2015

1 Introduction

There is evidence that the mean and volatility of macroeconomic variables change over time. For example, the US inflation rate[1] fluctuated between about two and ten percent in the 1970s and 1980s, whereas subsequent rates have been roughly between zero and four percent. While these instabilities are striking, it is not immediately clear how they should be modeled statistically. Studies such as Stock and Watson (2007), Clark and Ravazzolo (2015) and Bauwens et al. (2015) have shown that time series models which allow for time-varying parameters are useful in this context. Similarly, Primiceri (2005) and Del Negro and Primiceri (2015) propose a vector autoregressive (VAR) model in which regression coefficients and variance parameters are allowed to evolve according to random walks ("time-varying parameters and stochastic volatility").

The bvarsv package (initial CRAN release on August 28, 2014) provides functionality for Bayesian analysis of the Primiceri model. To the best of my knowledge, it is the only publicly available `R` code to do so. `bvarsv` complements `R` packages for Bayesian analysis of related time series models, including Ardia and Hoogerheide (2010), Brandt (2015), Kastner (forthcoming) and O'Hara (2015).[2] The MCMC algorithm underlying the Primiceri model is computationally challenging, since each iteration requires multiple calls of a recursive Kalman filter type algorithm. Therefore, bvarsv relies on C++ code, imported into R via Rcpp (Eddelbuettel and Francois 2011) and RcppArmadillo (Eddelbuettel and Sanderson 2014). The package allows to compute posterior predictive distributions, which can then be plotted and analyzed using forecast evaluation techniques. There is also functionality for impulse response analysis, which is often used to assess the dynamic effects of shocks to one of the variables in the system.

This paper aims to bridge the gap between Primiceri's original paper and the R help files for `bvarsv`. The reader is referred to Koop and Korobilis (2010) for an excellent overview of related literature, and to Dimitris Korobilis' website (https://sites.google.com/site/dimitriskorobilis/matlab) for associated `Matlab` code. Much of the code underlying `bvarsv` is based on the latter source.

The plan of this paper is as follows: Section 2 provides a concise description of the Primiceri model, with pointers to its implementation in `bvarsv`. Section 3 presents code examples illustrating the most important functionality. Section 4 concludes. A supplementary document available at https://sites.google.com/site/fk83research/code contains code to replicate results in Del Negro and Primiceri (2015).

2 The model

2.1 Data-generating process

Primiceri (2005) proposes the following model for an n-dimensional vector time series y_t:

[1] Measured as the year-over-year growth rate of the GDP price index. This time series is part of the data set usmacro.update contained in the bvarsv package.

[2] Ardia and Hoogerheide (2010) and Kastner (forthcoming) handle models for a single time series. In contrast, the Primiceri model is multivariate, describing several time series jointly. Brandt (2015) covers a multivariate Markov Switching model. Unlike the Primiceri model, Markov Switching implies abrupt (rather than smooth) change in the model parameters over time. O'Hara (2015) provides R/C++ code for several multivariate macroeconomic time series models, but does not cover the Primiceri model.

$$y_t = c_t + B_{1,t}\ y_{t-1} + \ldots + B_{p,t}\ y_{t-p} + A_t^{-1}\Sigma_t\ \varepsilon_t. \tag{1}$$

This is a vector autoregression (VAR) in y_t, with the special feature that the intercept c_t, as well as the coefficient matrices $\{B_{j,t}\}_{j=1}^p$, change over time. Furthermore, the composite error term $A_t^{-1}\Sigma_t\ \varepsilon_t$ (see details below) implies that the residual variance-covariance (VCV) matrix is allowed to vary over time as well. Both types of parameter instability are motivated by the characteristics of macroeconomic time series as sketched in the introduction.

More precisely, the full model is given by the following equations:

$$\begin{aligned} y_t &= \mathbf{X}_t' B_t + A_t^{-1}\Sigma_t\ \varepsilon_t && (2)\\ B_t &= B_{t-1} + \nu_t && (3)\\ \alpha_t &= \alpha_{t-1} + \zeta_t && (4)\\ \log\sigma_t &= \log\sigma_{t-1} + \eta_t, && (5) \end{aligned}$$

where y_t is a $n \times 1$ vector stacking the variables at a given date, $\mathbf{X}_t' = I_n \otimes \begin{bmatrix}1, & y_{t-1}, & \ldots, & y_{t-p}\end{bmatrix}$, B_t collects the parameters c_t and $\{B_{j,t}\}_{j=1}^p$ of Equation (1), A_t is a lower triangular matrix (with ones on the main diagonal) whose free elements are stacked in the vector α_t, and Σ_t is a diagonal matrix with positive elements $\sigma_t = diag(\Sigma_t)$. ε_t follows an n-variate standard normal distribution, and $\{\nu_t, \zeta_t, \eta_t\}$ are mean zero, homoscedastic and mutually independent Gaussian random vectors of appropriate dimensions.

2.2 Priors

The default settings for all priors follow *exactly* the same principles as in Primiceri (2005 Section 4.1), except that they account for the possibility that n is different from three (whereas $n = 3$ in the case of Primiceri's specification). Table 1 provides an overview. As to the priors on variance-covariance matrices, note that a random matrix V which follows an $\mathcal{IW}(A, b)$ distribution has mean $\frac{A}{b-d-1}$, where d is the (row and column) dimension of A (and V). Moreover, the variance of any element $V_{i,j}$ goes to zero as $b \to \infty$. That is, choosing a large value of b essentially fixes the matrix V at its mean. This can be used in practice to "shut off" some of the stochastic elements in the model.

Parameter	Description	Prior Family	Coefficient(s)
B_0	Initial betas	$\mathcal{N}(\hat{B}_{OLS},\ k_B \times \hat{V}(\hat{B}_{OLS}))$	$k_B = 4$
A_0	Initial covariance	$\mathcal{N}(\hat{A}_{OLS},\ k_A \times \hat{V}(\hat{A}_{OLS}))$	$k_A = 4$
$\log\sigma_0$	Initial log volatility	$\mathcal{N}(\log\hat{\sigma}_{OLS},\ k_\sigma \times I_n)$	$k_\sigma = 1$
Q	VCV of shocks to B_t	$\mathcal{IW}(k_Q^2 \times pQ \times \hat{V}(\hat{B}_{OLS}),\ pQ)$	$k_Q = 0.01, pQ = 40$
W	VCV of shocks to $\log\sigma_t$	$\mathcal{IW}(k_W^2 \times pW \times I_n,\ pW)$	$k_W = 0.01, pW = n + 1$
$S_j, j = 1, \ldots, n-1$	VCV of shocks to A_t	$\mathcal{IW}(k_S^2 \times pS_j \times \hat{V}(\hat{A}_{j,OLS}),\ pS_j)$	$k_S = 0.01, pS_j = j + 1$

Table 1: $\mathcal{N}$ and $\mathcal{IW}$ denote the normal and inverse Wishart distributions. $\hat{A}_{OLS}$, $\hat{V}(\hat{A}_{OLS})$, $\hat{B}_{OLS}$ and $\hat{V}(\hat{B}_{OLS})$ are obtained via training sample OLS (see Primiceri (2005) for details).

2.3 Sketch of the MCMC algorithm

`bvarsv` implements the algorithm by Primiceri (2005), with the correction noted by Del Negro and Primiceri (2015). The corrected algorithm is called "Algorithm 2" in the latter paper. This section sketches the algorithm; please refer to the original papers for derivations and details. Denote by B^T the entire path of

parameters $\{B_t\}_{t=1}^T$ (and similarly for Σ^T and A^T), let $\theta = [B^T, A^T, V]$ and let $V = [Q, S, W]$ collect the VCV matrices of the iid shock components $\{\nu_t, \zeta_t, \eta_t\}$. For clarity, we suppress dependence of the conditional posteriors on the observed data, and suppress variables which affect a conditional posterior in principle but not in practice.[3] Then the MCMC sampler can be summarized as follows.

1. Initialize A^T, Σ^T, s^T and V
2. Sample B^T from $p(B^T|\theta^{-B^T}, \Sigma^T)$, using the Carter and Kohn (1994) algorithm (henceforth, CK)
3. Sample Q from $p(Q|B^T)$, which is an inverse Wishart (IW) distribution
4. Sample A^T from $p(A^T|\theta^{-A^T}, \Sigma^T)$, again using CK
5. Sample S from $p(S|\theta^{-S}, \Sigma^T)$, which consists of several blocks that are IW
6. Sample the auxiliary discrete variables s^T from $p(s^T|\Sigma^T, \theta)$ for the Kim, Shephard, and Chib (1998) algorithm
7. Draw Σ^T from $p(\Sigma^T|\theta, s^T)$, using CK
8. Sample W from $p(W|\Sigma^T)$, which is IW
9. Go to Step 2.

2.4 Impulse Response Analysis

Questions of the type "What will happen if the central bank raises interest rates unexpectedly?" have both scientific and practical relevance. To some extent, these questions can be tackled with impulse response analysis (IRA). IRA provides an estimate of the response of $y_{t+h,i}$ (e.g., inflation in period $t+h$) given an unexpected shock to $y_{t,j}$ (e.g., the interest rate in period t). These estimates are functions of the model parameters in Equation (1); we refer to Hamilton (1994 Chapter 11.4) and Lütkepohl (2005 Chapter 2.3) for textbook treatments. Traditionally, IRA has been performed using constant parameter models. By contrast, the function `impulse.responses` provides functionality for IRA based on the Primiceri model, at a given (user specified) time period t. The current implementation covers three different scenarios:

- If scenario $= 1$, the analyis is based on the vector $u_t \equiv A_t^{-1}\Sigma_t\varepsilon_t$ (see Equation 1). Specifically, IRA estimates how a one-unit shock to a single element of u_t (i.e., $u_{t,j} = 1$, $u_{t,i} = 0$ for $i \neq j$) affects y_{t+h}, as compared to the case of u_t being a vector of zeros. Given that shocks are often correlated across variables (which is the case if $A_t^{-1}\Sigma_t$ is not a diagonal matrix), the comparison can be criticized as unrealistic. Note that the elements of c_t and $B_{j,t}$ from (1) are kept fixed at their time t values.
- If scenario $= 2$ (the default), IRA estimates the impact of a one-unit shock in some element of ε_t. The elements of $c_t, B_{j,t}, A_t$ and Σ_t in (1) are kept fixed at their time t values. Since the elements of ε_t are assumed to be independent, this scenario is more realistic than the one described in the last bullet point. However, this comes at a cost: The results are known to depend on the ordering of the variables in y_t. This ordering is an important modeling choice which needs to be justified on economic grounds (see Kilian 2013, Section 2, for a discussion and references).
- scenario $= 3$ is the same as scenario $= 2$, except that the elements of Σ_t are set to their average values over the sample period (rather than the time t value). This specification has been used by Primiceri (2005) and Del Negro and Primiceri (2015).[4]

[3] For example, in a Gibbs sampler with three blocks A, B, C, the conditional posterior $p(A|B, C)$ may be independent of C, and would be denoted $p(A|B)$ here.

[4] Code to replicate the findings in Del Negro and Primiceri (2015) is available at https://sites.google.com/site/fk83research/code.

Conceptually, it is important to note that impulse responses are estimated functions of the model's parameters. Hence, uncertainty about model parameters translates into uncertainty about impulse responses. In practice, we compute the relevant IRA quantities for each MCMC draw of the model parameters, and then provide summary statistics (such as the 5, 50 and 95 percent quantiles of the impulse responses across MCMC draws). We provide an example in Section 3.2.

2.5 Forecasting

The algorithm in Table 2 sketches the forecasting procedure implemented in the package. We provide a practical example in Section 3.3.

1. For each MCMC iteration $l = 1, \ldots,$ `nrep`:

   [A - Simulate Parameters]

   If `(pdrift == TRUE)`:

   Simulate paths of the relevant parameters for `nf` future periods, based on the current (iteration l) posterior draws.

   `else if (pdrift == FALSE)`

   Fix all parameters at their current values.

   [B - Simulate Disturbances]

   Simulate the disturbance vectors $\varepsilon_{T+j}, j = 1, \ldots,$ `nf` (iid standard normal)

   [C - Simulate Observations]

   Use the parameters and disturbances to simulate hypothetical observations $\{\tilde{y}_{T+j}^{(l)}\}_{j=1}^{\texttt{nf}}$ using the DGP in Equation (1).

   [D - Additionally compute Means/Variances conditional on parameters]

   For each period $j = 1, \ldots,$ `nf`, compute the mean vector $\mu_{T+j}^{(l)}$ and VCV matrix $V_{T+j}^{(l)}$ of Y_{T+j}, conditional on the simulated parameters until period $T + j$.

2. Thin the MCMC sequence: Retain draws for one out of every `thinfac` MCMC iterations, resulting in `nrep2 = nrep/thinfac` draws.

3. Return outputs:

   - `fc.ydraws` is an array of dimension $[M \times$ `nf` $\times$ `nrep2`$]$, where M is the number of variables in the VAR. For example, the `[1, 3, ]` elements contain all `nrep2` draws for the first variable, three periods ahead.
   - `fc.mdraws` contains the forecast means $\mu_{T+j}^{(l)}$, and is organized analogously to `fc.ydraws`
   - `fc.vdraws` contains the forecast variances $V_{T+j}^{(l)}$, and is an array of dimension $[0.5 \cdot M \cdot (M+1) \times$ `nf` $\times$ `nrep2`$]$. For example, the `[, 1, 1]` subarray contains the unique elements ($vech$) of $V_{T+1}^{(1)}$. The latter is the first draw of the joint VCV for all variables, one period ahead.

Table 2: Pseudo code for the forecasting algorithm implemented in the package. nrep, nf, pdrift and thinfac are input parameters to the function bvar.sv.tvp.

3 Code Examples

We next present examples of the package's main functions. The package is available on CRAN https://cran.r-project.org/web/packages/bvarsv/index.html, and can be installed via

```
install.packages(bvarsv)
```

3.1 Model Estimation

The following code estimates the model described in Section 2, using two lags of the dependent variable.

```
library(bvarsv)

# Set random seed
set.seed(1)
# Load data
data(usmacro)
# Fix number of Gibbs sampler draws used in this vignette
nburn.vignette <- 5000
nrep.vignette <- 50000
# Run estimation
fit <- bvar.sv.tvp(usmacro, p = 2, nburn = nburn.vignette, nrep = nrep.vignette)
```

```
## [1] "2015-11-26 11:31:12 -- now starting MCMC"
## [1] "2015-11-26 11:32:00 -- now at iteration 5000"
## [1] "2015-11-26 11:33:40 -- now at iteration 15000"
## [1] "2015-11-26 11:35:19 -- now at iteration 25000"
## [1] "2015-11-26 11:36:57 -- now at iteration 35000"
## [1] "2015-11-26 11:38:36 -- now at iteration 45000"
## [1] "2015-11-26 11:40:13 -- now at iteration 55000"
```

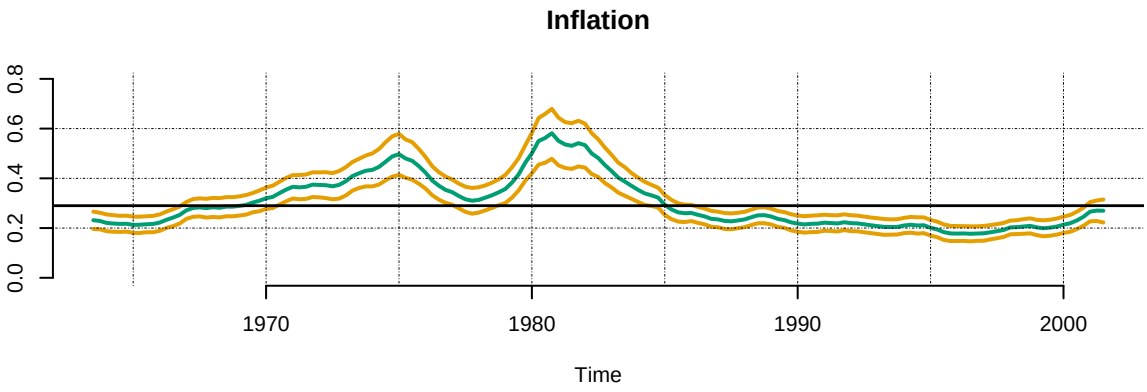
We compare the model to a simpler VAR, which is as in Equation (1), with the crucial difference that all parameters (for both the mean and variance) are constant over time. The latter model can be estimated using least squares, as implemented in the `vars` package due to Pfaff (2008).

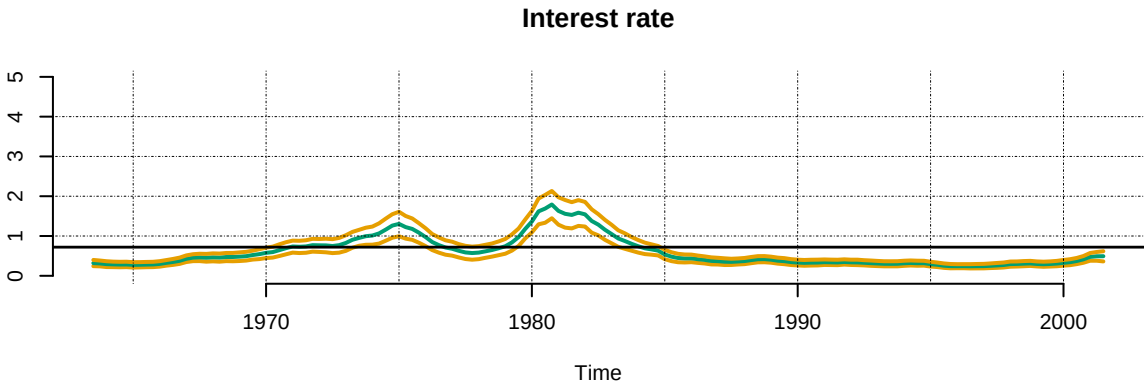
```
# Estimate simple VAR using vars package
library(vars)
fit.ols <- VAR(usmacro, p = 2)
```

To illustrate the model output, we next plot the standard deviation of the model residuals over time. Our figure replicates Figure 9 from the online appendix of Del Negro and Primiceri (2015). The figure shows clear time variation in the residual standard deviations fitted by the Primiceri model. By contrast, the standard deviation in the simpler benchmark model (black horizontal line) is constant over time.

```
# Replicate Figure 9 in Del Negro and Primiceri (2015)

# Some auxiliary definitions for plotting
matplot2 <- function(...) matplot(..., type = "l", lty = 1, lwd = 2,
                                  bty = "n", ylab = "")
stat.helper <- function(z) c(mean(z), quantile(z, c(0.16, 0.84)))[c(2, 1, 3)]
xax <- seq(1963.5, 2001.5, 0.25) # x axis
```

```r
gp <- seq(1965, 2000, 5) # marks for vertical lines
# colors, taken from http://www.cookbook-r.com
cols <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
          "#D55E00", "#CC79A7")
cols1 <- cols[c(2, 4, 2)]

# Residual variance from simpler benchmark model
sd.residuals.ols <- apply(residuals(fit.ols), 2, sd)

# Make plot
par(mfrow = c(3, 1))

# SD of inflation residual
# Get posterior draws
sd_inf <- parameter.draws(fit, type = "vcv", row = 1, col = 1)
x1 <- t(apply(sqrt(sd_inf), 2, stat.helper))
# Plot
matplot2(x = xax, y = x1, ylim = c(0, 0.8), col = cols1,
        main = "Inflation", xlab = "Time")
abline(h = c(0.2, 0.4, 0.6), v = gp, lty = 4, lwd = 0.3)
abline(h = sd.residuals.ols[1], col = cols[1], lwd = 1.4)

# SD of unemployment residual
# Get posterior draws
sd_une <- parameter.draws(fit, type = "vcv", row = 2, col = 2)
x2 <- t(apply(sqrt(sd_une), 2, stat.helper))
# Plot
matplot2(x = xax, y = x2, ylim = c(0, 1), col = cols1,
        main = "Unemployment", xlab = "Time")
abline(h = c(0.5, 1), v = gp, lty = 4, lwd = 0.3)
abline(h = sd.residuals.ols[2], col = cols[1], lwd = 1.4)

# SD of interest rate residual
# Get posterior draws
sd_tbi <- parameter.draws(fit, type = "vcv", row = 3, col = 3)
x3 <- t(apply(sqrt(sd_tbi), 2, stat.helper))
# Plot
matplot2(x = xax, y = x3, ylim = c(0, 5), col = cols1,
         main = "Interest rate", xlab = "Time")
abline(h = 1:4, v = gp, lty = 4, lwd = 0.3)
abline(h = sd.residuals.ols[3], col = cols[1], lwd = 1.4)
```





3.2 Impulse Response Analysis

We next estimate the response of the inflation rate to a shock in the interest rate, based on the model from Section 3.1. We refer to Primiceri (2005 Section 4.2) for an economic justification of the variable ordering chosen here.

```
par(mfrow = c(1, 1))
ira <- impulse.responses(fit, impulse.variable = 3, response.variable = 1)

# OLS impulse responses for comparison
```

```r
ira.ols <- irf(fit.ols, n.ahead = 20)[[1]][[3]][-1, 1]

# Add to plot
lines(x = 1:20, y = ira.ols, lwd = 3, lty = 5)
```

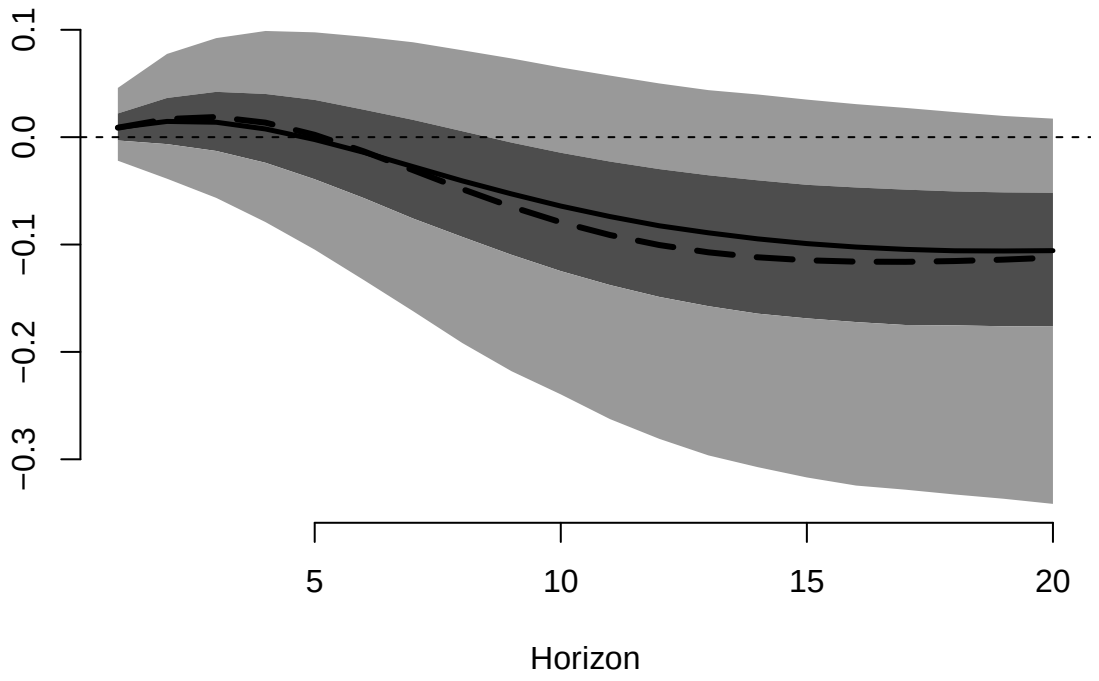


Figure 2 shows that the median estimate of the response (solid line) is positive at first, turning negative after a few quarters. The OLS estimate (dashed line) is very similar. That said, the 5 and 95 percent quantiles of the estimated response form a wide range, and include the zero line. This indicates that there is substantial uncertainty about the relevant effect, which is a typical finding in the literature.

3.3 Forecasting

Here we illustrate how the `bvarsv` package can be used to construct out-of-sample forecast distributions. We use an updated version of Primiceri's data set for this purpose. As a first step, we estimate the model using data from all but the last `n.fc` time periods.

```r
# Select variable to be forecast (1 = inf, 2 = une, 3 = tbi)
sel.var <- 2

# Select number of forecast periods
n.fc <- 2

# Load updated data set and define estimation sample
data(usmacro.update)
data.est <- head(usmacro.update, nrow(usmacro.update) - n.fc)
```

```r
# Fit BVAR to estimation sample
fit2 <- bvar.sv.tvp(data.est, p = 2, nburn = nburn.vignette, nrep = nrep.vignette)
```

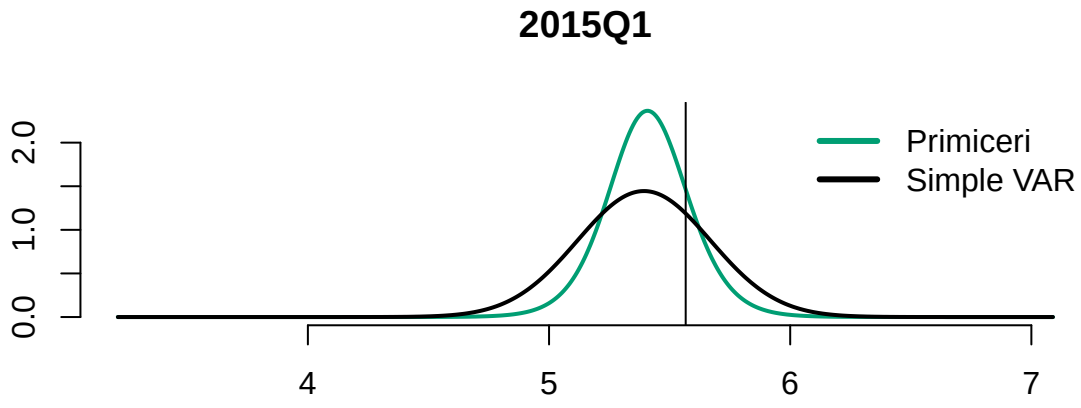
```
## [1] "2015-11-26 11:40:18 -- now starting MCMC"
## [1] "2015-11-26 11:41:22 -- now at iteration 5000"
## [1] "2015-11-26 11:43:28 -- now at iteration 15000"
## [1] "2015-11-26 11:45:34 -- now at iteration 25000"
## [1] "2015-11-26 11:47:40 -- now at iteration 35000"
## [1] "2015-11-26 11:49:45 -- now at iteration 45000"
## [1] "2015-11-26 11:51:53 -- now at iteration 55000"
```

```r
# Simpler model for comparison (constant parameters, OLS estimation)
fit2.ols <- VAR(data.est, p = 2)
fc.ols <- predict(fit2.ols, n.ahead = n.fc)[[1]][[sel.var]]
fc.ols.mean <- fc.ols[, 1]
fc.ols.sd <- (fc.ols[, 1] - fc.ols[, 2])/qnorm(0.975)
```

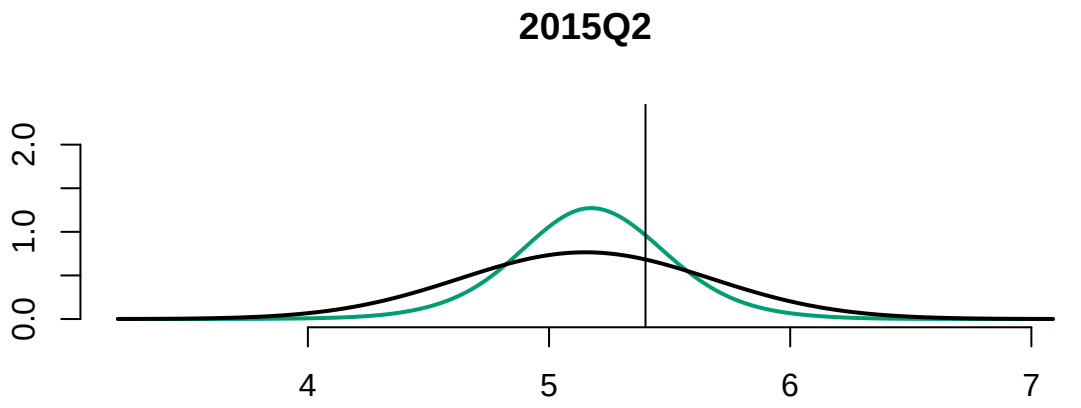
Next, we compute forecasts for the last `n.fc` time points, which are often called the "out-of-sample" period.

```r
# Get out-of-sample data
data.out <- tail(usmacro.update, n.fc)[, sel.var]

# Get time labels
lab.helper <- function(z) paste0(floor(z), "Q", 4*(z-floor(z)) + 1)
time.labels <- lab.helper(tail(time(usmacro.update), n.fc))

# Auxiliary definitions for plot
xax.lim <- qnorm(c(0.0001, 0.9999), mean = fc.ols.mean[2],
                 sd = fc.ols.sd[2])
xax <- seq(from = xax.lim[1], to = xax.lim[2],
           length.out = 500)
par(mfrow = c(n.fc, 1))
cols2 <- cols[c(4, 1)]

# Loop over hh, the time points in the out-of-sample period
for (hh in 1:n.fc){

  # Forecast density from fit
  f1 <- predictive.density(fit2, h = hh, v = sel.var)
  # OLS forecast density
  f2 <- function(z) dnorm(z, mean = fc.ols.mean[hh], sd = fc.ols.sd[hh])

  # Fix scaling
  if (hh == 1) y.max <- max(c(f1(xax), f2(xax)))

  # Compute log scores
  ls1 <- log(f1(data.out[hh]))
  ls2 <- log(f2(data.out[hh]))

  # Plot
  matplot2(x = xax, y = cbind(f1(xax), f2(xax)), col = cols2, main = time.labels[hh],
           ylim = c(0, y.max), sub = paste0("Log scores: ", round(ls1, 3),
```

```
                                        " (Primiceri), ", round(ls2, 3),
                                        " (Simple VAR)"), xlab = "")
  # Add legend to first panel
  if (hh == 1){
    legend("topright", c("Primiceri", "Simple VAR"), col = cols2, bty = "n", lwd = 3)
  }
  # Vertical line at realization
  abline(v = data.out[hh])
}
```



Log scores: 0.381 (Primiceri), 0.172 (Simple VAR)



Log scores: −0.04 (Primiceri), −0.382 (Simple VAR)

The figures show that the forecast distribution of the Primiceri model is more concentrated (i.e., has smaller variance) than the one from the simpler VAR with constant parameters. This is because the variance of the simpler VAR is a time average over the complete estimation sample, which includes the volatile period of the 1970s and 1980s. By contrast, the Primiceri model accounts for the fact that the system variables have become less volatile over time. In the present example, the smaller forecast variance of the Primiceri model seems to be desirable: The log score, which is a performance measure for distribution forecasts (see e.g. Gneiting and Katzfuss (2014)), is larger for the Primiceri model than for the simpler benchmark.

4 Conclusion

This paper has introduced the `R` package `bvarsv` which implements the model by Primiceri (2005) and Del Negro and Primiceri (2015). While there are several types of questions that can be studied with the Primiceri model, the main motivation for the development of `bvarsv` is to use the model for probabilistic forecasting (see Section 3.3). In contrast to traditional point forecasting, probabilistic forecasting entails the measurement and communication of uncertainty, which is an integral part of most practically relevant forecasting tasks. The Primiceri model seems well suited in economic contexts, since i) the model allows for time-varying parameters and stochastic volatility, which are important stylized facts of macroeconomic time series, and ii) Bayesian estimation naturally accounts for parameter uncertainty.

Acknowledgments

This work has been funded by the European Union Seventh Framework Programme under grant agreement no. 290976. I thank Dimitris Korobilis for sharing his excellent `Matlab` code, and Alexander Jordan, Giorgio E Primiceri, Francesco Ravazzolo and Patrick Schmidt for helpful comments.

References

Ardia, David, and Lennart F Hoogerheide. 2010. "Bayesian Estimation of the GARCH(1, 1) Model with Student-t Innovations." *The R Journal* 2: 41–47.

Bauwens, Luc, Gary Koop, Dimitris Korobilis, and Jeroen VK Rombouts. 2015. "The Contribution of Structural Break Models to Forecasting Macroeconomic Series." *Journal of Applied Econometrics* 30: 596–620.

Brandt, Patrick. 2015. *MSBVAR: Markov-Switching, Bayesian, Vector Autoregression Models*. http://CRAN.R-project.org/package=MSBVAR.

Carter, Chris K, and Robert Kohn. 1994. "On Gibbs Sampling for State Space Models." *Biometrika* 81: 541–53.

Clark, Todd E, and Francesco Ravazzolo. 2015. "Macroeconomic Forecasting Performance Under Alternative Specifications of Time-Varying Volatility." *Journal of Applied Econometrics* 30: 551–75.

Del Negro, Marco, and Giorgio E Primiceri. 2015. "Time-Varying Structural Vector Autoregressions and Monetary Policy: A Corrigendum." *Review of Economic Studies* 82: 1342–45.

Eddelbuettel, Dirk, and Romain Francois. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40: 1–18.

Eddelbuettel, Dirk, and Conrad Sanderson. 2014. "RcppArmadillo: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis* 71: 1054–63.

Gneiting, Tilmann, and Matthias Katzfuss. 2014. "Probabilistic Forecasting." *Annual Review of Statistics and Its Application* 1: 125–51.

Hamilton, James D. 1994. *Time Series Analysis*. Vol. 2. Princeton University Press.

Kastner, Gregor. forthcoming. "Dealing with Stochastic Volatility in Time Series Using the R Package stochvol." *Journal of Statistical Software*.

Kilian, Lutz. 2013. "Structural Vector Autoregressions." In *Handbook of Research Methods and Applications in Empirical Macroeconomics*, 515–54. Edward Elgar.

Koop, Gary, and Dimitris Korobilis. 2010. "Bayesian Multivariate Time Series Methods for Empirical Macroeconomics." *Foundations and Trends in Econometrics* 3: 267–358.

Lütkepohl, Helmut. 2005. *New Introduction to Multiple Time Series Analysis*. Springer.

O'Hara, Keith. 2015. *BMR: Bayesian Macroeconometrics in R*. http://bayes.squarespace.com/bmr/.

Pfaff, Bernhard. 2008. "VAR, SVAR and SVEC Models: Implementation Within R Package vars." *Journal of Statistical Software* 27: 1–32.

Primiceri, Giorgio E. 2005. "Time Varying Structural Vector Autoregressions and Monetary Policy." *The Review of Economic Studies* 72: 821–52.

Stock, James H, and Mark W Watson. 2007. "Why Has U.S. Inflation Become Harder to Forecast?" *Journal of Money, Credit and Banking* 39: 3–33.