bssm: Bayesian Inference of Non-linear/Non-Gaussian State Space Models in R

Jouni Helske and Matti Vihola
University of Jyväskylä, Department of Mathematics and Statistics, Finland

October 6, 2016

this vignette is very much work in progress, for details see function documentation and paper by Vihola, Helske, and Franks (2016)

Introduction

The R package `bssm` is designed for Bayesian inference of general state space models with Gaussian states dynamics and non-Gaussian and/or non-linear observational distribution. We define the state transition equation as

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t,$$

where $\eta_t \sim N(0, I_k)$ and $\alpha_1 \sim N(a_1, P_1)$ independently of each other. For observation level equation, the `bssm` package currently supports basic stochastic volatility model and general exponential family state space models. The observation equation for the stochastic volatility model is defined as

$$y_t = x_t'\beta + \sigma \exp(\alpha_t/2)\epsilon_t, \quad \epsilon_t \sim N(0,1),$$

where x_t contains the exogenous covariate values at time t, with β corresponding to the regression coefficients. For stochastic volatility model the state equation is defined as

$$\alpha_{t+1} = \rho \alpha_t + \sigma_\eta \eta_t,$$

with $\alpha_1 \sim N(0, \sigma^2_\eta/(1-\rho^2))$.

For exponential family models, general state transition equation is used, and the observation equation has a general form

$$p(y_t | Z_t \alpha_t, x_t'\beta, \phi, u_t).$$

Currently, following observational level distributions are supported:

- Gaussian distribution: $p(y_t|Z_t\alpha_t, x_t'\beta) = x_t'\beta + Z_t\alpha_t + H_t\epsilon_t$ with $\epsilon_t \sim N(0,1)$.
- Poisson distribution: $p(y_t|Z_t\alpha_t, x_t'\beta, u_t) = \text{Poisson}(u_t \exp(x_t'\beta + Z_t\alpha_t))$, where u_t is the known exposure at time t.
- Binomial distribution: $p(y_t|Z_t\alpha_t, x_t'\beta, u_t) = \text{binomial}(u_t, \exp(x_t'\beta + Z_t\alpha_t)/(1+\exp(x_t'\beta + Z_t\alpha_t)))$, where u_t is the size and $\exp(x_t\beta + Z_t\alpha_t)/(1+\exp(x_t'\beta + Z_t\alpha_t))$ is the probability of the success.
- Negative binomial distribution: $p(y_t|Z_t\alpha_t, x_t'\beta, \phi, u_t) = \text{negative binomial}(\exp(x_t'\beta + Z_t\alpha_t), \phi, u_t)$, where $u_t \exp(x_t'\beta + Z_t\alpha_t)$ is the expected value and ϕ is the dispersion parameter (u_t is again exposure term).

The `bssm` package aims to provide easy-to-use and efficient functions for fully Bayesian inference of common time series models such basic structural time series model (BSM) (Harvey 1989) with exogenous covariates, making it straighforward and efficient to make predictions and other inference in a Bayesian setting.

The Bayesian framework of `bssm` is based on Vihola, Helske, and Franks (2016). The core methodology reliest on Markov chain Monte Carlo (MCMC) approach with adaptive random walk Metropolis updating, using RAM algorithm by (Vihola 2012).

Several efficient MCMC variants can be used, such as pseudo-marginal MCMC based on particle filtering, optionally with delayed acceptance, as well as the two-step procedure based on importance sampling type correction introduced in (Vihola, Helske, and Franks 2016).

MCMC algorithm for Gaussian state space models

For Gaussian models given the parameters θ, the likelihood of the model can be computed using the well known Kalman filter recursions. The complete adaptive MCMC algorithm of `bssm` for Gaussian models is as follows (modified from (Vihola 2012)).

Given the target acceptance rate α^* (e.g. 0.234) and $\gamma \in (0, 1]$, at iteration i

1. Compute the proposal $\theta'_i = \theta_{i-1} + S_{i-1}u_i$, where u_i is simulated from the standard d-dimensional Gaussian distribution and S_{i-1} is a lower diagonal matrix with positive diagonal elements.
2. Accept the proposal with probability $\alpha_i := \min\{1, p(y|\theta'_i)/p(y|\theta_{i-1})\}$.
3. If the proposal θ'_i is accepted, set $\theta_i = \theta'_i$ and simulate a realization (or multiple realizations) of the states α from $p(\alpha|y, \theta'_i)$ using the simulation smoothing algorithm by (Durbin and Koopman 2002). Otherwise, set $\theta_i = \theta_{i-1}$ and $(\alpha_1, \ldots, \alpha_n)_i = (\alpha_1, \ldots, \alpha_n)_{i-1}$.
4. Compute (using Cholesky update or downdate algorithm) the Cholesky factor matrix S_i satisfying the equation
$$S_i S_i^T = S_{i-1} \left(I + \min\{1, d i^{-\gamma}\}(\alpha_i - \alpha^*) \frac{u_i u_i^T}{\|u_i\|^2} \right) S_{i-1}^T.$$

If the interest is in the posterior means and variances of the states, we can replace the simulation smoothing in step 3 with standard fixed interval smoothing which gives the smoothed estimates (expected values and variances) of the states given the data and the model parameters. From these, the posterior means and variances of the states can be computed straightforwardly.

Non-Gaussian models

The observational densities of our non-Gaussian/non-linear models are all twice differentiable, so we can straightforwardly use the Laplace approximation based on (Durbin and Koopman 2000). This gives us an approximating Gaussian model which has the same mode of $p(\alpha|y)$ as the original model. Often this approximating Gaussian model works well as such, and thus we can use it in MCMC scheme directly, which results in an approximate Bayesian inference. We can also use the approximating model together with importance sampling which produces exact Bayesian inference on $p(\alpha, \theta|y)$. We can factor the likelihood of the non-Gaussian model as (Durbin and Koopman 2012)

$$\begin{aligned} p(y|\theta) &= \int g(\alpha, y|\theta) \mathrm{d}\alpha \\ &= g(y|\theta) E_g \left[\frac{p(y|\alpha, \theta)}{g(y|\alpha, \theta)} \right], \end{aligned}$$

where $g(y|\theta)$ is the likelihood of the Gaussian approximating model and the expectation is taken with respect to the Gaussian density $g(\alpha|y, \theta)$. Equivalently we can write

$$\begin{aligned} \log p(y|\theta) &= \log g(y|\theta) + \log E_g \left[\frac{p(y|\alpha, \theta)}{g(y|\alpha, \theta)} \right] \\ &= \log g(y|\theta) + \log \frac{p(y|\hat{\alpha}, \theta)}{g(y|\hat{\alpha}, \theta)} + \log E_g \left[\frac{p(y|\alpha, \theta)/p(y|\hat{\alpha}, \theta)}{g(y|\alpha, \theta)/g(y|\hat{\alpha}, \theta)} \right] \\ &= \log g(y|\theta) + \log \hat{w} + \log E_g w^* \\ &\approx \log g(y|\theta) + \log \hat{w} + \log \frac{1}{N} \sum_{j=1}^N w_j^*, \end{aligned}$$

where $\hat{\alpha}$ is the conditional mode estimate obtained from the approximating Gaussian model. For approximating inference without the importance sampling we simply omit the term $\log \frac{1}{N}\sum_{j=1}^N w_i^*$. In principle, using the exact Bayesian inference we should simulate multiple realizations of the states α in each iteration of MCMC in order to compute $\log \frac{1}{N}\sum_{j=1}^N w_j^*$. Fortunately, we can use so called delayed acceptance approach (Christen and Fox 2005; Banterle et al. 2015) which speeds up the computation considerably:

1. Make initial acceptance of the given proposal θ_i' with probability $\min\left\{1, \frac{g(y|\theta_i')\hat{w}_i}{g(y|\theta_{i-1})\hat{w}_{i-1}}\right\}$.
2. If accepted, perform the importance sampling of the states α and make the delayed acceptance with probability $\min\{1, \sum_{j=1}^N w_{i,j}^{*'} / \sum_{j=1}^N w_{i-1,j}^*\}$.
3. If the delayed acceptance is successful, set $\theta_i = \theta_i'$ and sample one (or multiple) realization of the previously simulated states with weights $w_{i,j}, j = 1, \ldots, N$ (with replacement in case of multiple samples are stored). Otherwise, set $\theta_i = \theta_{i-1}$ and similarly for the states.

Particle filtering

In addition to simulation smoother based importance sampling inference, `bssm` supports particle filtering for computing the log-likelihood estimates and simulating states. Two types of filters are available, classical bootstrap filter, and ψ-auxiliary filter based on Guarniero, Johansen, and Lee (), where we use same Gaussian approximation as in simulation smoother based importance sampling.

Importance-sampling type correction

For efficient parallel computation, the importance-sampling type correction presented in Vihola, Helske, and Franks (2016) can be used, where the MCMC algorithm targets the approximating posterior (based on the Gaussian model), and the correction to actual target posterior is made in offline fashion.

Prediction intervals

For predicting future observations, one can extend the time series data with missing values which are handled straightforwardly by Kalman filter. This gives us point predictions in form of expected values, but for prediction intervals, no analytical formulas are available as the posterior distribution of the state is not necessarily normal. A standard way of obtaining the prediction intervals is to simulate future observations and use empirical quantiles for the limits of prediction interval. However, this is can be computationally expensive as a large number of simulations is needed for accurate values of tail probabilities. In addition of standard quantile method, the `bssm` package contains alternative method for Gaussian models based on the parametric method by (Helske et al. 2015, J. Helske (2016)). Here, instead of simulating the future values, we compute the expected values and variances of the future values using the Kalman filtering, and then solve simple equation based on the cumulative distribution function of Normal distribution.

Example

Here is a short example:

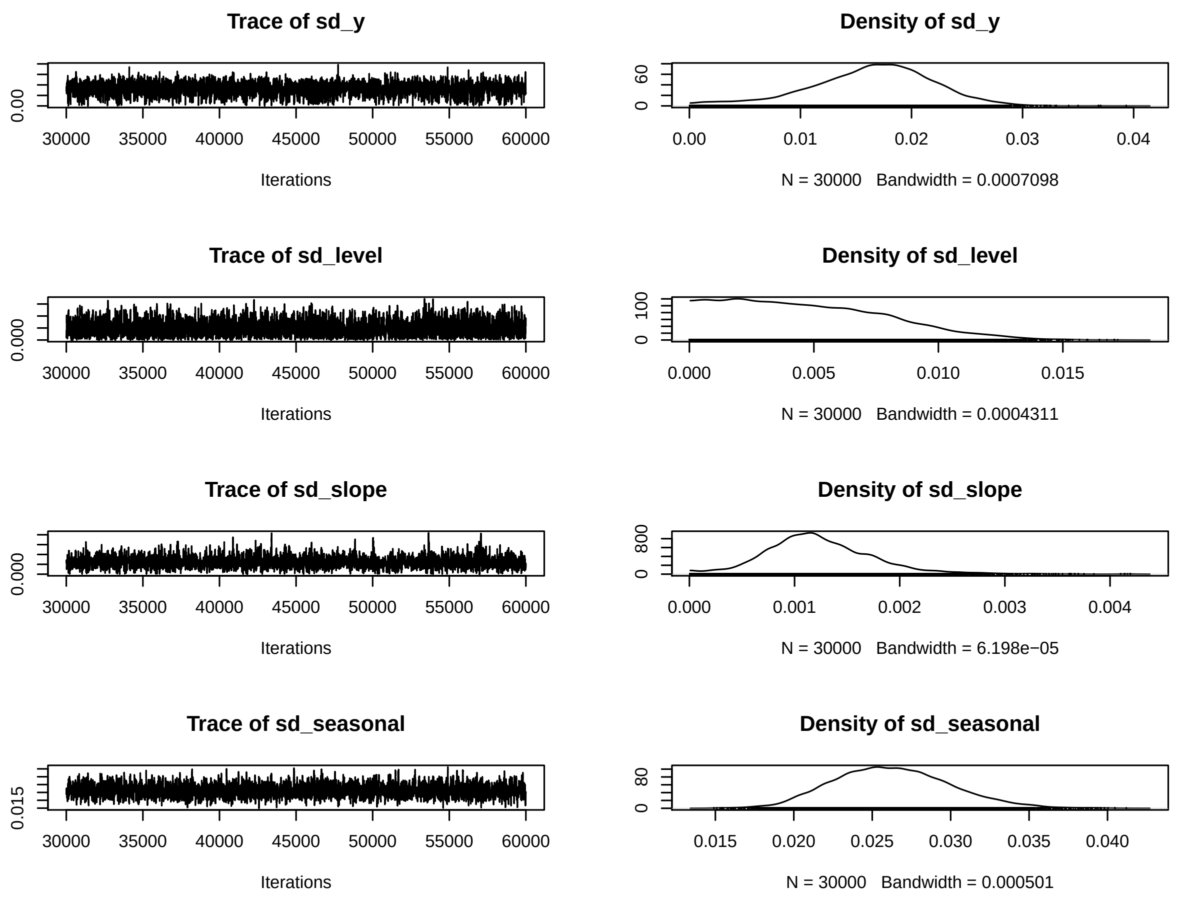
```r
library("bssm")
set.seed(123)

init_sd <- 0.1 * sd(log10(UKgas))
prior <- halfnormal(init_sd, 1)
model <- bsm(log10(UKgas), sd_y = prior, sd_level = prior,
  sd_slope = prior, sd_seasonal = prior)
```

```r
mcmc_out <- run_mcmc(model, n_iter = 6e4)
mcmc_out
```

```
##
## Call:
## run_mcmc.bsm(object = model, n_iter = 60000)
##
## Iterations = 30001:60000
## Thinning interval = 1
##
## Acceptance rate after the burnin period:  0.2417
##
## Summary for theta:
##
##                    Mean           SD     Naive SE Time-series SE
## sd_y        0.016388853 0.0055515986 3.205217e-05   1.290343e-04
## sd_level    0.004791204 0.0031963512 1.845414e-05   7.907511e-05
## sd_slope    0.001238565 0.0005199038 3.001666e-06   1.292040e-05
## sd_seasonal 0.026277860 0.0037147568 2.144716e-05   8.946217e-05
##
## Effective sample sizes for theta:
##
##        sd_y    sd_level    sd_slope sd_seasonal
##    1851.082    1633.915    1619.175    1724.177
##
## Summary for alpha_108:
##
##                    Mean          SD     Naive SE Time-series SE
## level       2.835249006 0.013798108 7.966341e-05   2.378171e-04
## slope       0.009772475 0.003698608 2.135392e-05   6.235012e-05
## seasonal_1  0.061572894 0.017956737 1.036733e-04   3.063212e-04
## seasonal_2 -0.295250269 0.015722111 9.077165e-05   2.654431e-04
## seasonal_3 -0.034840220 0.014496334 8.369462e-05   2.393691e-04
##
## Effective sample sizes for alpha_108:
##
##      level      slope seasonal_1 seasonal_2 seasonal_3
##   3366.301   3518.859   3436.377   3508.155   3667.584
```

```r
plot(mcmc_out$theta)
```



```r
# posterior mode estimates
mcmc_out$theta[which.max(mcmc_out$posterior), ]
```

```
##         sd_y     sd_level     sd_slope  sd_seasonal
## 0.0188472520 0.0004900121 0.0011749573 0.0253084266
```

```r
# posterior covariance matrix:
cov(mcmc_out$theta)
```

```
##                      sd_y      sd_level      sd_slope   sd_seasonal
## sd_y         3.082025e-05 -7.581897e-06  1.363660e-07 -1.277841e-05
## sd_level    -7.581897e-06  1.021666e-05 -5.828447e-07  2.231501e-06
## sd_slope     1.363660e-07 -5.828447e-07  2.702999e-07 -3.360741e-09
## sd_seasonal -1.277841e-05  2.231501e-06 -3.360741e-09  1.379942e-05
```

```r
# compare to shape of the proposal distribution:
cor(mcmc_out$theta)
```

```
##                    sd_y   sd_level    sd_slope sd_seasonal
## sd_y         1.00000000 -0.4272729  0.04724602 -0.61962402
## sd_level    -0.42727287  1.0000000 -0.35073199  0.18793696
## sd_slope     0.04724602 -0.3507320  1.00000000 -0.00174013
## sd_seasonal -0.61962402  0.1879370 -0.00174013  1.00000000
```

```r
cov2cor(mcmc_out$S %*% t(mcmc_out$S))
```

```
##                    sd_y    sd_level    sd_slope sd_seasonal
## sd_y         1.00000000 -0.33669116  0.08655638 -0.60798266
## sd_level    -0.33669116  1.00000000 -0.45392238  0.06626984
## sd_slope     0.08655638 -0.45392238  1.00000000 -0.01552120
## sd_seasonal -0.60798266  0.06626984 -0.01552120  1.00000000
```

Smoothed trend:

```
ts.plot(model$y, rowMeans(mcmc_out$alpha[, "level", ]), col = 1:2)
```

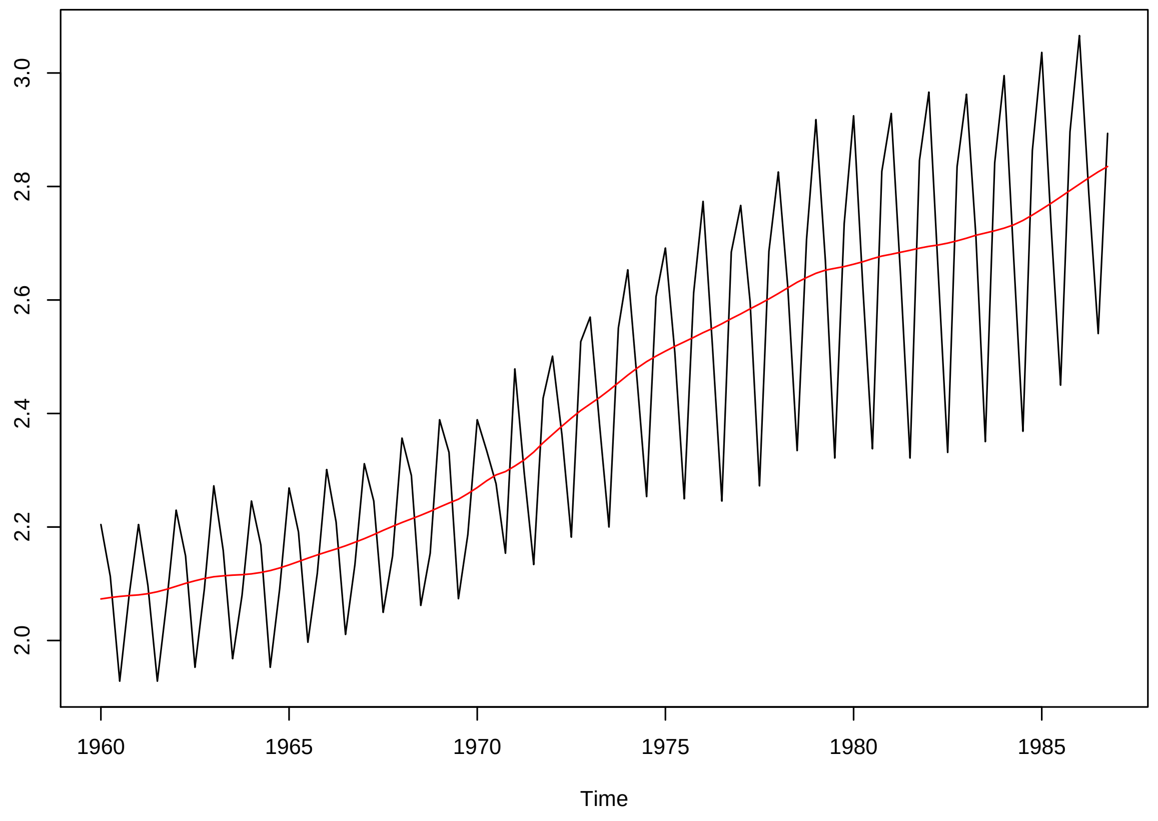


Figure 1: Smoothed trend component.

Prediction intervals:

```
pred <- predict(model, n_iter = 1e4, n_ahead = 40,
  probs = c(0.025, 0.1, 0.9, 0.975), S = mcmc_out$S)
ts.plot(log10(UKgas), pred$mean, pred$intervals[,-3],
  col = c(1, 2, c(3, 4, 4, 3)), lty = c(1, 1, rep(2, 4)))
```

With ggplot2:

```
require("ggplot2")
autoplot(pred, interval_color = "red", alpha_fill = 0.2)
```

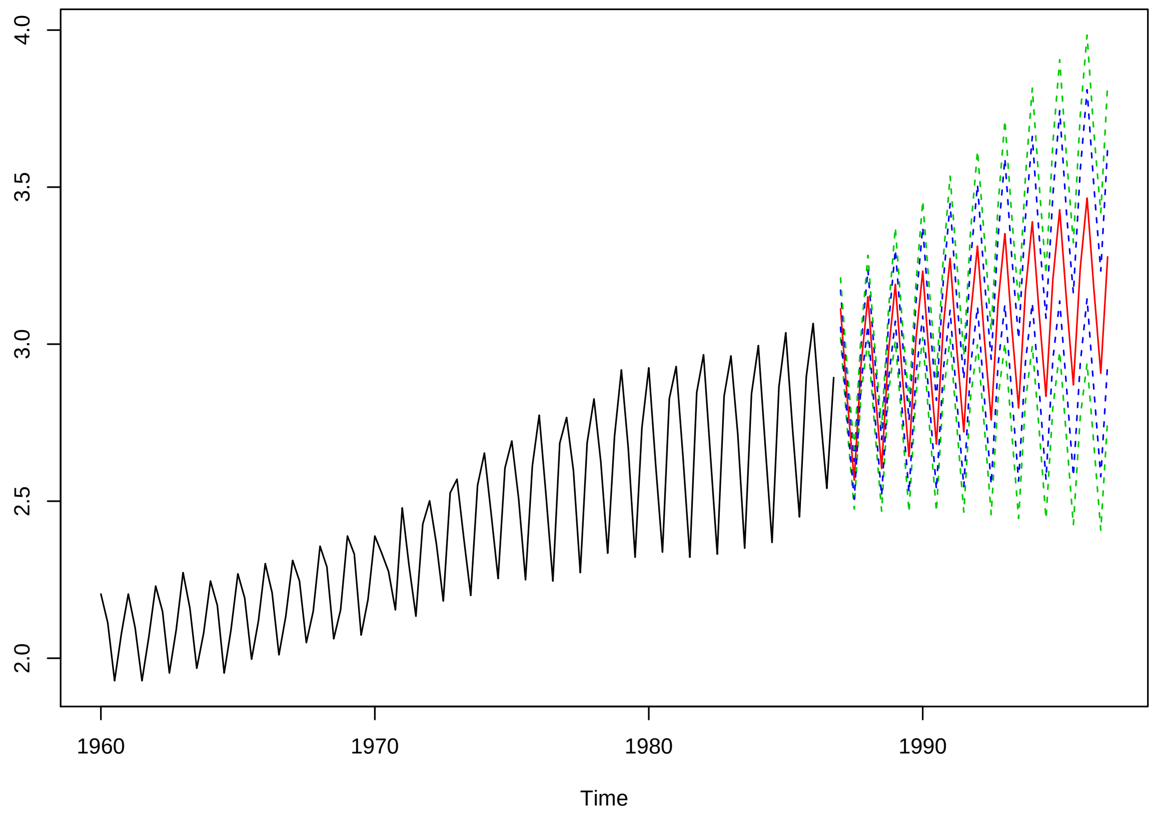


Figure 2: Mean predictions and prediction intervals.

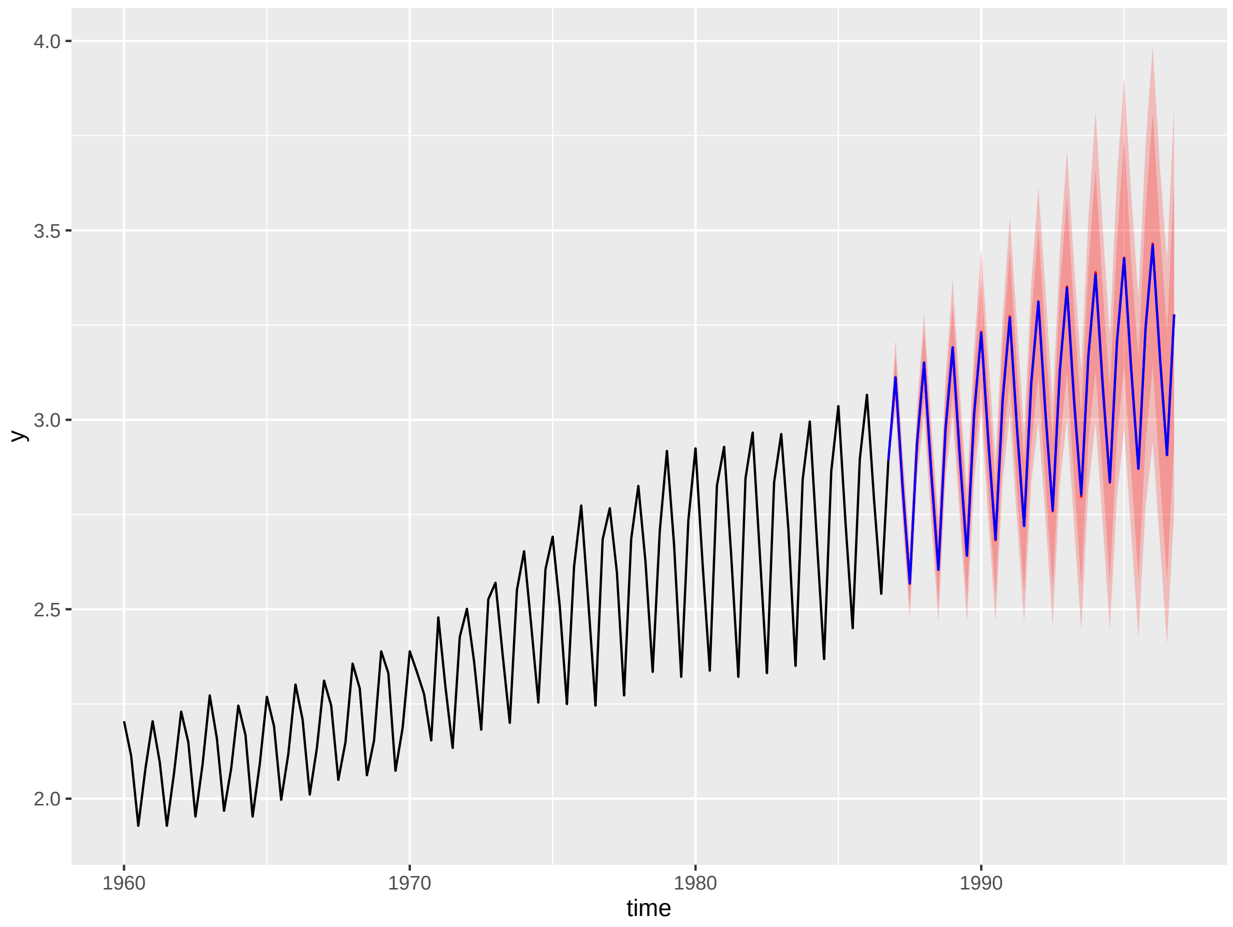


Figure 3: Prediction plots with ggplot2.

References

Banterle, M., C. Grazian, A. Lee, and C. P. Robert. 2015. "Accelerating Metropolis-Hastings algorithms by Delayed Acceptance." *ArXiv E-Prints*, March. http://arxiv.org/abs/1503.00996.

Christen, J. Andrés, and Colin Fox. 2005. "Markov Chain Monte Carlo Using an Approximation." *Journal of Computational and Graphical Statistics* 14 (4): 795–810. doi:10.1198/106186005X76983.

Durbin, J., and S. J. Koopman. 2000. "Time Series Analysis of Non-Gaussian Observations Based on State Space Models from Both Classical and Bayesian Perspectives." *Journal of Royal Statistical Society B* 62: 3–56.

———. 2002. "A Simple and Efficient Simulation Smoother for State Space Time Series Analysis." *Biometrika* 89: 603–15.

———. 2012. *Time Series Analysis by State Space Methods*. 2nd ed. New York: Oxford University Press.

Guarniero, Pieralberto, Adam M. Johansen, and Anthony Lee"The Iterated Auxiliary Particle Filter." *Journal of the American Statistical Association* 0 (ja): 0–0. doi:10.1080/01621459.2016.1222291.

Harvey, A. C. 1989. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Helske, Jouni. 2016. "Computationally Efficient Bayesian Prediction Intervals for Gaussian State State Models." *Working Paper*.

Helske, J., Nyblom, and J. 2015. "Improved Frequentist Prediction Intervals for Autoregressive Models by Simulation." In *Unobserved Components and Time Series Econometrics*, edited by Koopman, S. J., Shephard, and N., 291–309. Oxford University Press.

Vihola, M., J. Helske, and J. Franks. 2016. "Importance sampling type correction of Markov chain Monte Carlo and exact approximations." *ArXiv E-Prints*, September.

Vihola, Matti. 2012. "Robust Adaptive Metropolis Algorithm with Coerced Acceptance Rate." *Statistics and Computing* 22 (5): 997–1008. doi:10.1007/s11222-011-9269-5.