Bayesian Model of Stochastic Volatility

Zhou Fang Saiyang Zhang

April 14, 2023

Stochastic Volatility Model

Question: Given the daily price of the $S\&P$ 500 index, how do we predict the volatility of the price based on the Stochastic Bayesian Model?

One of the earliest literature and perhaps the most influential in the Bayesian modeling for stochastic volatility is [2] Lots of works in more complicated scenarios have been derived since then, the followings are some of the influential ones, [4], [5], [3], [1].

Assume the underlying dynamics for one asset is

$$dS_t = \mu S_t dt + \sqrt{\upsilon_t} S_t dW_t \tag{1}$$

$$d\upsilon_t = (\theta - \alpha \log \upsilon_t)\upsilon_t dt + \xi \upsilon_t dB_t \tag{2}$$

in the above, $\alpha > 0$, $\theta > 0$, $\upsilon_t = \sigma_t^2$ is the variance at time t, S_t is the asset's price. W_t and B_t are two independent Brownian motions. The dynamic for S_t is assumed to be geometric Brownian motion with drift, which is quite common for asset prices' dynamics. The dynamic for the variance is assumed to be mean-reverting, with certain shocks θ proportional to the current variance. As one can see, if the current variance is too small, then the drift term $\alpha \log \upsilon_t$ will be positive, which makes the variance, in expectation, increase. When the variance is too large, the drift term will be negative, which will make the variance, in

expectation, decrease. This mean-reverting dynamic makes sense because, for one asset, the variance can neither be too large nor 0.

Now, let $h_t = \log \upsilon_t$, then by the Ito's formula,

$$d \log \upsilon_t = \frac{d\upsilon_t}{\upsilon_t} - \frac{\langle \upsilon, \upsilon \rangle_t}{2\upsilon_t^2} \tag{3}$$

$$= (\theta - \frac{\xi^2}{2} - \alpha \log \upsilon_t)dt + \xi dB_t \tag{4}$$

since $h_t = \log \upsilon_t$, the above becomes

$$dh_t = (\theta - \frac{\xi^2}{2} - \alpha h_t)dt + \xi dB_t \tag{5}$$

$$= a_1 dt - \alpha h_t dt + \xi dB_t \tag{6}$$

where, $a_1 = \theta - \frac{\xi^2}{2}$. Now, set $r_t = \frac{dS_t}{S_t}$ which is the return at time t. then, the dynamics after taking log scales

$$r_t = \mu dt + e^{\frac{h_t}{2}} dW_t \tag{7}$$

$$dh_t = a_1 dt - \alpha h_t dt + \xi dB_t \tag{8}$$

if we discretize the above dynamics by setting $dt = 1$ unit of time, and $a_2 = 1 - \alpha$ then we get the connecting functions

$$r_t = \mu + e^{\frac{h_t}{2}} \epsilon_t \ , \quad \epsilon_t \sim \mathcal{N}(0, 1) \tag{9}$$

$$h_t = a_1 + a_2 h_{t-1} + \xi \eta_t \ , \quad \eta_t \sim \mathcal{N}(0, 1) \tag{10}$$

to conduct Bayesian inference. To find the distribution for r_t, we rewrite Eq(9) as

$$r_t \sim \mathcal{N}(\mu, e^{\frac{h_t}{2}}) \tag{11}$$

First, we will need to set the priors for coefficients for μ, a_1, a_2, and ξ. We set the priors as

follows,

$$\mu \sim \mathcal{N}(\mu_0, \sigma_0) \tag{12}$$

$$a_1 \sim \mathcal{N}(\mu_1, \sigma_1) \tag{13}$$

$$a_2 \sim \mathcal{N}(\mu_2, \sigma_2) \tag{14}$$

$$\xi^2 \sim \Gamma^{-1}(\alpha_0, \beta_0) \tag{15}$$

We construct the script file as the following:

1: Read off the realized volatility data as *realized_vol* and return data as *return* from the file

2: Initialize the prior for μ, a_1 and a_2 for each run, as $\mu_0 = \mu_1 = \mu_2 = 0$ and $\sigma_0 = \sigma_1 = \sigma_2 = 0.25$, which is a realistic value for the stock market.

3: Initialize and tune the parameter for inverse gamma distribution for ξ, as $\alpha_0 = 100$ and $\beta_0 = 1$

4: Initialize the number of all data set as m and training set as $n = 20$ that used for posterior prediction

5: Construct matrix that store the posterior prediction value for volatility and returns from the training set

6: for i in 1:(m-n) {

7: *Initialize the training set for return as the i to the $(i+n)$-th element, record it as r. Do the same for logrithm of realized_vol as h*

8: *Run the fitting algorithm and collect the sample output lists*

9: *From the model, take the mean of the sampled μ, a_1, a_2 and ξ*

10: *From the mean of the sampled data, predict the volatility by taking the exponential power of h_t and return using eq(10) and eq(9)*

11: }

12: Sample and make the trace plot for μ, a_1, a_2 and ξ to check the convergence.

13: Sample and make the violin plot for μ, a_1, a_2 and ξ to check the confidence interval.

14: Compare realized volatility to the predicted volatility by ploting the distribution density plot

15: compare realized return to the predicted return by ploting the distribution density plot

And here we write the algorithm for each run:

1: fix the indexing in the time series, and make sure we have r_2 to r_t as the return at the end of each time interval, and h_1 to h_{t-1} as the volatility at the start of each time interval

2: **Build the Jags model**:

3: *for i in 1:n* {

4: r_i *given by normal distribution with mean* μ *and variance* e^{h_t}*, where* h_t *is given by Eq(10)*

5: *Generate* η_i *at each time step in the Eq(10) from normal distribution with mean of 0 and variance of 1*

6: }

7: μ, a_1 *and* a_2 *sampled from normal distribution with mean* μ_0, μ_1 *and* μ_2 *and standard deviation* σ_0, σ_1 *and* σ_2 *,respectively.*

8: *Sample* ξ^2 *from the inverse gamma distribution with shape parameter* α_0 *and* β_0

9: **End**

10: Run Jags model using the initialized value

11: Initialize number of MCMC trials

12: Update the burn-in period

Results Comparisons

When conducting the empirical studies, we set 1 unit time to be 30 mins. We study the Bayesian inference on the realized volatility of the $S\&P$ 500 index. The recent 1 month's data with 2 mins resolution is retrieved from yahoo finance, and the realized volatility is obtained from every 15 closed prices, which means the resolution for realized volatility is 30

mins.

After running the MCMC algorithm, we pick the last run for convergence check for four of our variables μ, a_1, a_2 and ξ, as shown in Fig.4 and 2. The results sampled from MCMC run show a nice convergence. And then, we make a violin plot of those variables to find the confidential interval as shown in Fig.3.

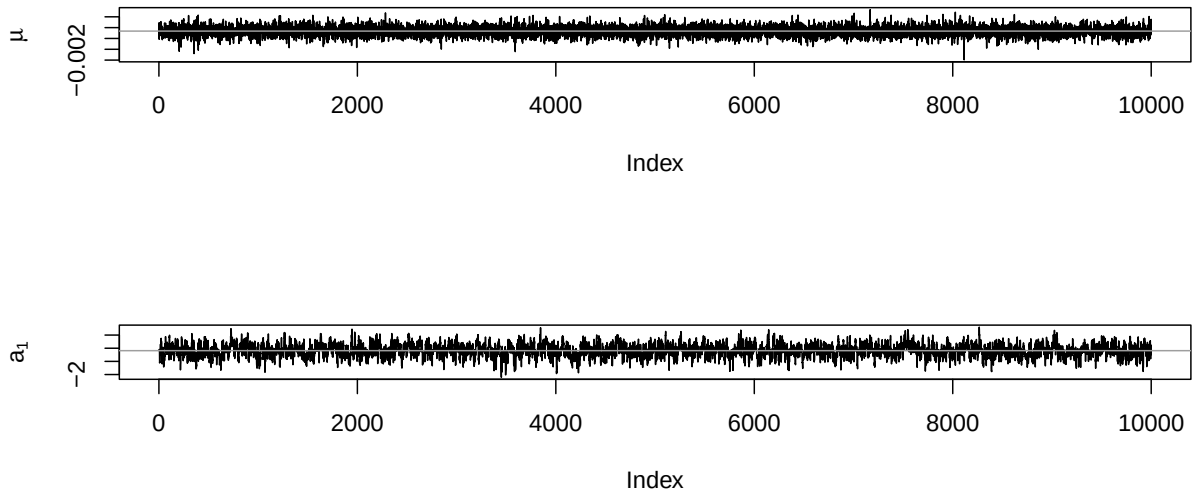

Figure 1: Trace plot for two variables μ and a_1 in the last run, which shows a nice convergence

After we check for convergence, we calculate the Return and Volatility using the sampled value from the posterior distribution. We compare the density distribution of predicted Return in green lines and Volatility with the real data as shown in blue lines. We find that our prediction captures the Return data but missed the distribution of Volatility and generating a much wider spread. However, we successfully predict the mean of volatility.

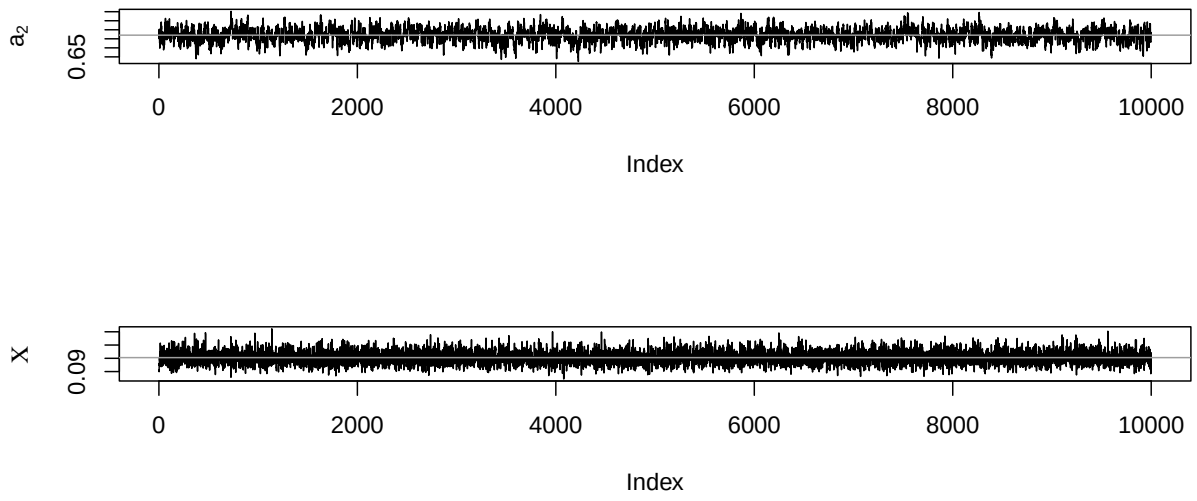

Figure 2: Trace plot for two variables a_2 and ξ in the last run, which shows a nice convergence

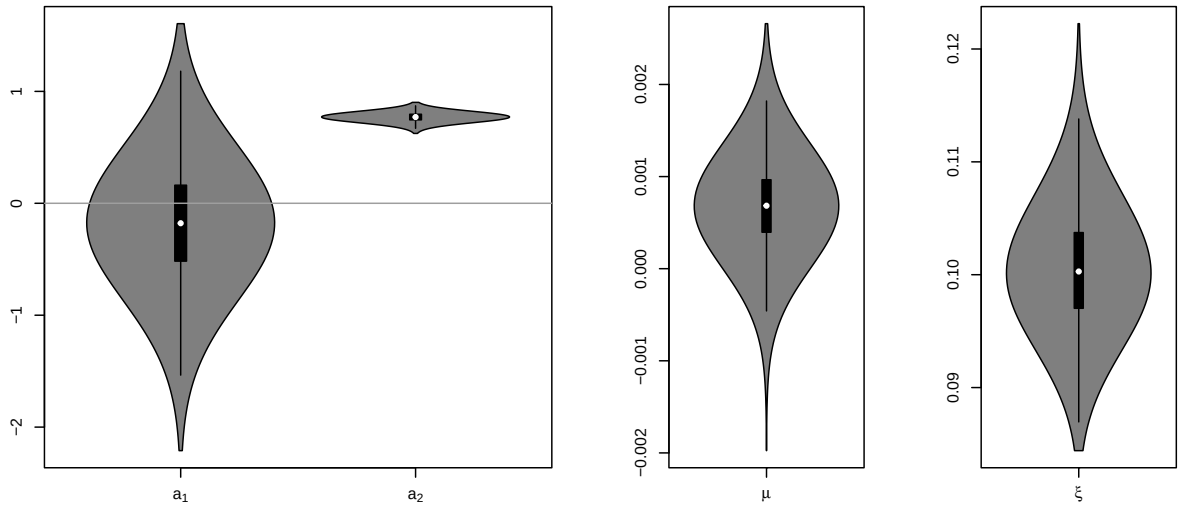

Figure 3: Violin plot showing the confidential interval for all variables μ, a_1, a_2 and ξ in the last run, which show a nice convergence

Author Contributions

- SZ wrote the solutions document and checked over ZF's MCMC algorithm and R script.

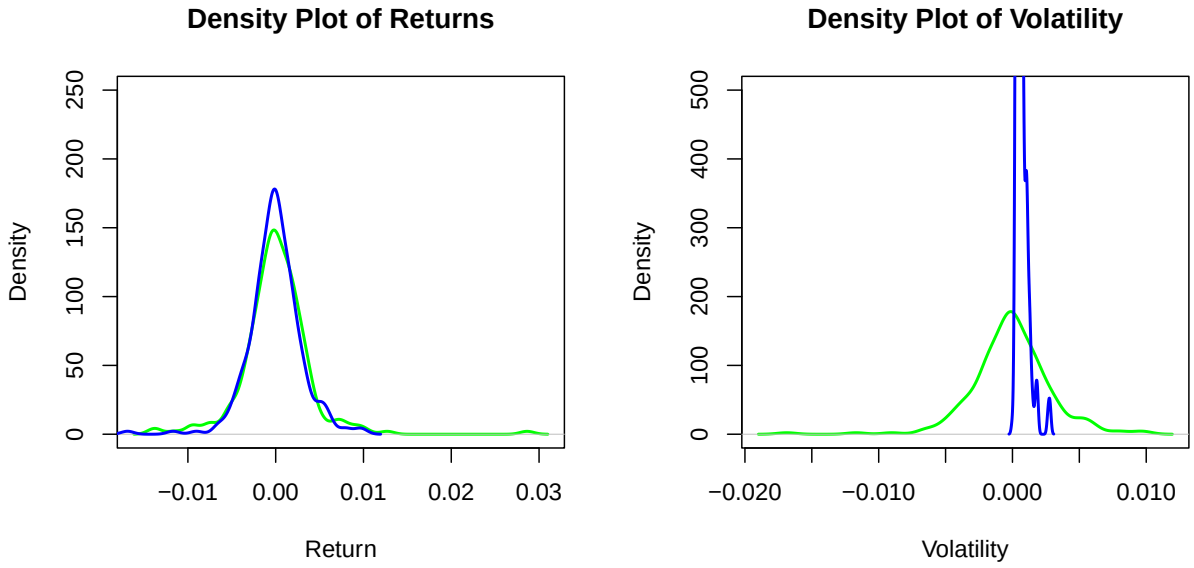

Figure 4: Density function for predictions of the volatility and return from posterior distribution (green), as compared to that for the volatility and return from real data (blue). In this case, we set the number of training data to be $n = 20$

- ZF wrote the MCMC algorithm and R script and checked over the solutions document.

APPENDIX A: R Script for The Plot

```
SV.model <- function(r, h, n, a, sigma, alpha, beta, n.mcmc){
r <- r[2:n]
h <- h[1:(n - 1)]
library(rjags)
m.jags <-
"model{
      for (i in 1:n){
      r[i] ~ dnorm(mu, exp(-(a1 + a2 * h[i] + xi * eta[i])))
      eta[i] ~ dnorm(0, 1)
      }
    a1 ~ dnorm(a[1], 1/sigma[1])
    mu ~ dnorm(a[2], 1/sigma[2])
    a2 ~ dnorm(a[3], 1/sigma[3])
    # write the inverse gamma distribution
    xi <- sqrt(1 / inv.xi.square)
    inv.xi.square ~ dgamma(alpha, beta)
}"

mod <- textConnection(m.jags)
m.out <- jags.model(mod, data=list('r'=r, 'h'=h, 'n'=(n - 1), 'a'=a, 'sigma'=
    sigma, 'alpha'=alpha, 'beta' = beta), n.chains=1)
n.burn <- round(.1 * n.mcmc)
update(m.out, n.burn)
m.samples <- jags.samples(m.out, c('mu', 'a1', 'a2', 'xi'), n.mcmc)
list(m.samples=m.samples)
}
```

APPENDIX B: R Script for The rjags Algorithm

```
realized_vol <- read.csv("~/Desktop/UTexas/Spring 2023/SDS 384 Baysian/hw 6/
    realized_vol.csv",header = FALSE)
returns <- read.csv("~/Desktop/UTexas/Spring 2023/SDS 384 Baysian/hw 6/sp500.
    csv",header = FALSE)
source("~/Desktop/UTexas/Spring 2023/SDS 384 Baysian/hw 6/Stochastic_
    Volatility.r")

a <- c(0, 0, 0)
sigma <- c(0.25, 0.25, 0.25)
alpha <- 100
beta <-1
n.mcmc <- 10000
n <- 20
m <- dim(returns)[1]
#create a vector to store the predictions
predictions <- rep(0, m - n)
returns.post <- rep(0, m - n)

```

```
for (i in 2:(m - n)){
  r <- returns[i:(i + n), ]
  h <- 2 * log(realized_vol[i:(i + n), ])
  out <- SV.model(r, h, n, a, sigma, alpha, beta, n.mcmc)
  coeff.samples <- out$m.samples
  mu <- mean(coeff.samples$mu)
  a1 <- mean(coeff.samples$a1)
  a2 <- mean(coeff.samples$a2)
  xi <- mean(coeff.samples$xi)
  # make inference based on the mean of those samples
  random <- rnorm(1, 0, 1)
  random1 <- rnorm(1, 0, 1)
  predict <- a1 + a2 * h[(n-1)] + xi * random
  return <- mu + exp(predict/2)*random1
  returns.post[i] = return
  predictions[i] = predict
  print(i)

}


#plot the posterior distribution to check for convergence
layout(matrix(1:2,2,1))
plot(coeff.samples$mu,type="l",lty=1,ylab=bquote(mu))
abline(h=mu,col=8)
plot(coeff.samples$a1,type="l",lty=1,ylab=bquote(a[1]))
abline(h=a1,col=8)
plot(coeff.samples$a2,type="l",lty=1,ylab=bquote(a[2]))
abline(h=a2,col=8)
plot(coeff.samples$xi,type="l",lty=1,ylab=bquote(Chi))
abline(h=xi,col=8)

library(vioplot)
layout(matrix(c(1,1,2,3),1,4))
data<-list(coeff.samples$a1,coeff.samples$a2)
vioplot(data,names=expression(a[1],a[2]))
abline(h=0,col=8)

vioplot(coeff.samples$mu,names=expression(mu))
vioplot(coeff.samples$xi,names=expression(xi))

predictions
predict_vol <- exp(predictions / 2)
predict_vol
realized_vol[20:m, ]

#plot the prior and posterior distribution for prediction and return

layout(matrix(1:2,1,2))
density <- density(returns.post)
plot(density, main = "Density Plot of Returns", xlab = "Return", col = "green"
```

```
   , lwd = 2,ylim = c(0, 250))
density1 <- density(returns$V1)
lines(density1, type = "l", lwd = 2, col = "blue")

density_vol <- density(predict_vol)
plot(density1, main = "Density Plot of Volatility", xlab ='Volatility', col =
   "green", lwd = 2,ylim = c(0, 500))
density_vol1 <- density(realized_vol$V1)
lines(density_vol1, type = "l", lwd = 2, col = "blue")
```

References

[1] Todd E Clark. "Real-time density forecasts from Bayesian vector autoregressions with stochastic volatility". In: *Journal of Business & Economic Statistics* 29.3 (2011), pp. 327–341.

[2] Eric Jacquier, Nicholas G Polson, and Peter E Rossi. "Bayesian analysis of stochastic volatility models". In: *Journal of Business & Economic Statistics* 20.1 (2002), pp. 69–87.

[3] Eric Jacquier, Nicholas G Polson, and Peter E Rossi. "Bayesian analysis of stochastic volatility models with fat-tails and correlated errors". In: *Journal of Econometrics* 122.1 (2004), pp. 185–212.

[4] Renate Meyer and Jun Yu. "BUGS for Bayesian analysis of stochastic volatility models". In: *The Econometrics Journal* 3.2 (2000), pp. 198–215.

[5] Harald Uhlig. "Bayesian vector autoregressions with stochastic volatility". In: *Econometrica: Journal of the Econometric Society* (1997), pp. 59–73.