CALIBRATION ON MERTON JUMP DIFFUSION USING BAYESIAN MCMC METHOD

RAN ZHAO

ABSTRACT.

1. INTRODUCTION

Advances in computing powers and numerical methods have largely improve the capability of solving econometric and statistical models using computational intense methods, includes Markov Chain Monte Carlo (MCMC) method. Especially in dynamic asset pricing models, the MCMC method is widely utilized to extracting information about latent state variables (such as implied volatility), structural parameters and market prices of risk (volatility or jump risks) from observed prices or market quotes. The Bayesian inference is to obtain the distribution of parameter set, Θ, and (optional) state variables, X, conditioning on the observed prices, Y. That is, the posterior distribution, $p(\Theta, X|Y)$ is vital to the parameters estimation and their statistical inference.

Consider a stochastic process $\{X_t\}$, where each X_t assumes value in space Ω. Then the process $\{X_t\}$ is a Markov process if given the value of X_t, the values of X_{t+h}, $h > 0$, do not depend on the values X_s, $s < t$. That is, $\{X_t\}$ is a Markov process if its conditioanl distribution function satisfies

$$\mathbb{P}(X_{t+h}|X_s, s \leq t) = \mathbb{P}(X_{t+h}|X_t), \quad h > 0.$$

In continuous-time asset pricing models, MCMC that explore their posterior distributions samples from high-dimensional and sophisticated distributions by generating Markon process over (Θ, X), $\{\Theta^{(g)}, X^{(g)}\}_{g=1}^{G}$. And the equilibrium distribution of (Θ, X) is $p(\Theta, X|Y)$. Then Monte Carlo methods use these samples for statistical inference on parameters and states.

However, $p(\Theta, X|Y)$ in continuous-time asset pricing models is usually not easy to obtain. Johannes and Polson [4] listed the reasons for this difficulty, which summarize as

(1) market prices are observed discretely (e.g. on daily basis) while the asset pricing models specify the prices and states to evolve continuously;
(2) the state variables are latent based on researcher's perspective but not observable on the market;
(3) $p(\Theta, X|Y)$ is usually in high dimension, causing common sampling method to fail;
(4) the transition distributions for prices and states of the asset pricing model are non-normal and non-standard, complication the standard estimation methods such as MLE and GMM;
(5) the parameters of the asset pricing models are usually nonlinear and non-analytic form as the implicit solution to a stochastic differential equations.

A typical application of MCMC technique in asset pricing model is Jacquier, Polson and Rossi [3], where a cyclic Metropolis algorithm is used to construct a Markov-chain simulation on stochastic volatility model.

2. MODEL SPECIFICATION

2.1. **Geometric Brownian Motion (Black-Scholes).** The baseline model selected for fitting the underlying stock returns is Black-Scholes model [1], where the stock price dynamic, S_t, follows

Geometric Brownian Motion

$$dS_t = \left(\mu + \frac{1}{2}\sigma^2\right) S_t dt + \sigma S_t dW_t$$

where μ is the drift term and σ is the volatility. W_t is the Wiener process. This model assumes the stock returns follow a random walk. In reality, the S&P500 index level and returns on daily basis are plotted in Figure 1.

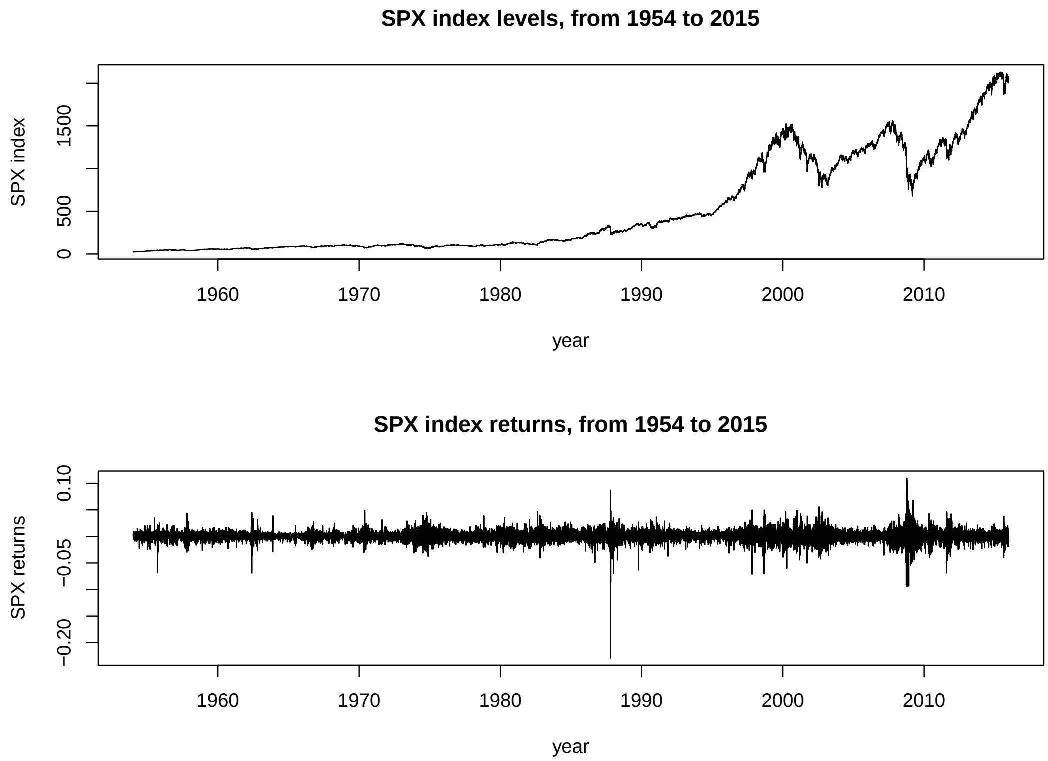

Figure 1 The SPX index levels and return on daily basis. Time period is from 1954 to 2015.

In discrete time equally space, the model has close-form solution for the return

$$Y_t = \log(S_t / S_{t-1}) = \mu + \sigma \epsilon_t$$

where $\epsilon_t \sim N(0,1)$. We have $\Theta = (\mu, \sigma^2)$. There is no latent variable, which implies the posterior to be $p(\Theta|Y) = p(\mu, \sigma|Y)$.

Using Hammersley-Clifford theorem [2], $p(\mu|\sigma^2, Y)$ and $p(\sigma^2|\mu, Y)$ are complete conditionals to the posterior. Assuming independent priors on μ and σ^2, Bayes rule implies that

$$\begin{aligned} p(\mu|\sigma^2, Y) &\propto p(Y|\mu, \sigma^2)(\mu) \\ p(\sigma^2|\mu, Y) &\propto p(Y|\mu, \sigma^2)(\sigma^2) \\ p(Y|\mu, \sigma^2) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t - \mu}{\sigma}\right)^2\right) \end{aligned}$$

where T is the sample size. $p(\mu)$ and $p(\sigma^2)$ are priors. Here we choose the standard conjugate priors on μ and σ^2. First select the inverse gamma distribution as the prior for σ^2. The inverse gamma distribution relies on two parameters α and β. The density is

$$f(\sigma^2|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1}\exp(-\beta/\sigma^2)$$

Therefore, the marginal density $p(\sigma^2)$ combines the prior $p(\sigma^2)$ and density $p(Y|\mu,\sigma^2)$, which yields

$$\begin{aligned}
p(\sigma^2|\mu,Y) &\propto p(Y|\mu,\sigma^2)\times p(\sigma^2) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t-\mu}{\sigma}\right)^2\right)\times\frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1}\exp(-\beta/\sigma^2) \\
&\propto (\sigma^2)^{-T/2-\alpha-1}\exp\left(-\left[\frac{1}{2}\sum_{t=1}^{T}(Y_t-\mu)^2+\beta\right]/\sigma^2\right) \\
&\propto IG\left(\alpha+\frac{T}{2},\beta+\frac{1}{2}\sum_{t=1}^{T}(Y_t-\mu)^2\right)
\end{aligned}$$

That is, given μ and Y_t, we are able to generate the σ^2 according to the marginal density $p(\sigma^2|\mu,Y)$. Similarly, select normal distribution as prior for μ. The density is

$$f(\mu|\theta,\delta) = \frac{1}{\sqrt{2\pi\delta^2}}\exp\left(-\frac{1}{2}\left(\frac{\mu-\theta}{\delta}\right)^2\right)$$

and the marginal density is

$$\begin{aligned}
p(\mu|\sigma^2,Y) &\propto p(Y|\mu,\sigma^2)\times p(\mu) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t-\mu}{\sigma}\right)^2\right)\times\frac{1}{\sqrt{2\pi\delta^2}}\exp\left(-\frac{1}{2}\left(\frac{\mu-\theta}{\delta}\right)^2\right)
\end{aligned}$$

To deal with $Y_t-\mu$, denote $\hat{\mu} = \left(\sum_{t=1}^{T} Y_t\right)/T$, and

$$\begin{aligned}
\sum_{t=1}^{T}(Y_t-\mu)^2 &= \sum_{t=1}^{T}(Y_t-\hat{\mu}+\hat{\mu}-\mu)^2 \\
&= \sum_{t=1}^{T}(Y_t-\hat{\mu})^2+2(\hat{\mu}-\mu)\sum_{t=1}^{T}(Y_t-\hat{\mu})+\sum_{t=1}^{T}(\hat{\mu}-\mu)^2 \\
&= \sum_{t=1}^{T}(Y_t-\hat{\mu})^2+T(\hat{\mu}-\mu)^2
\end{aligned}$$

Continuing on the marginal density, we have

$$\begin{aligned}
p(\mu|\sigma^2,Y) &\propto \exp\left(-\frac{T}{2\sigma^2}(\mu-\mu)^2-\frac{1}{2\delta^2}(\mu-\theta)^2\right) \\
&\propto \exp\left(-\frac{T}{2\sigma^2}(-2\hat{\mu}\mu+\mu^2)-\frac{1}{2\delta^2}(\mu^2-2\mu\theta)\right) \\
&\propto \exp\left(-\frac{1}{2\delta^{*2}}\left(\mu-\left(\frac{T\hat{\mu}}{\sigma^2}+\frac{\theta}{\delta^2}\right)\delta^{*2}\right)^2\right) \\
&\propto N\left(\left(\sum_{t=1}^{T}Y_t/\sigma^2+\theta/\delta^2\right)\delta^{*2},\delta^{*2}\right)
\end{aligned}$$

where $\delta^{*2} = (T/\sigma^2+1/\delta^2)^{-1}$.

Given the prior distributions, the complete MCMC method to conduct parameter estimation and statistical inference is

(1) initialize the parameters $\mu^{(0)}$ and $(\sigma^2)^{(0)}$;
(2) specify the parameters of the prior $\alpha, \beta, \theta, \delta$;
(3) draw $\mu^{(g+1)} \sim p(\mu | (\sigma^2)^{(g)}, Y)$;
(4) draw $(\sigma^2)^{(g+1)} \sim p(\sigma^2 | \mu^{(g+1)}, Y)$;
(5) estimate parameters in $\{\mu^{(g)}, (\sigma^2)^{(g)}\}_{g=1}^{G}$

and G is the simulation size. In this paper, we select $G = 2000$.

2.2. **Merton Jump Diffusion Model.** Merton's (1976) jump diffusion model assumes the (log) return process Y_t as a mixture of Poisson distributed jumps and the Geometric Brownian Motion

$$Y_t \equiv \log(S_t / S_{t-1}) = \mu + \sigma + Z_t \xi_t \tag{1}$$

As shown in Equation 1, in the additional to the Black-Scholes part, a jump process is incorporated, where Z_t is the jump indictor equal to 1 with probability λ (the jump intensity) and equal to 0 with probability $(1 - \lambda)$. ξ_t is a normally distributed random variable representing the jump size.

Different from the Black-Scholes model, there are two latent variables (state variables), Z_t and ξ_t. That is, $X = \{Z_t, \xi_t\}_{t=1}^{T}$. As shown below,

$$\begin{aligned} Z_t &\sim \begin{cases} 1 & \text{with probability } \lambda \\ 0 & \text{with probability } (1-\lambda) \end{cases} \\ \xi &\sim N(\mu_s, \sigma_s^2) \end{aligned}$$

The parameter set is $\Theta = \{\mu, \sigma^2, \lambda, \mu_s, \sigma_s^2\}$, where μ_s and σ_s^2 are the parameters that define the distribution of ξ. The observed returns are $Y = \{r_t\}_{t=1}^{T}$. The object is to estimate Θ (and states X) conditional to the returns Y. Given $p(\Theta, X|Y) = p(\Theta, Z, \xi|Y)$, the marginal densities are obtained

$$p(\Theta, X|Y) \propto p(Y|\Theta, X) p(X|\Theta) p(\Theta)$$

Similar with Black-Scholes model, Hammersley-Clifford suggestions the following approach

(1) draw $\Theta_i^{(g+1)} \sim p(\Theta_i | \Theta_{i \backslash c}^{(g)}, Z^{(g)}, \xi^{(g+1)}, Y)$;
(2) draw $Z^{(g+1)} \sim p(Z | \Theta^{(g+1)}, \xi^{(g)}, Y)$;
(3) draw $\xi^{(g+1)} \sim p(\xi | \Theta^{(g+1)}, Z^{(g+1)}, Y)$.

where $\Theta_{i \backslash c}$ represents the parameter set without parameter Θ_i. G is the simulation size. In this paper, we select $G = 2000$.

Defining the priors and deriving the posteriors is vital to MCMC algorithm. The density of the observed return is

$$\begin{aligned} p(Y|\Theta, Z, \xi) &= \Pi_{t=1}^{T} p(Y_t | \Theta, Z_t, \xi_t) \\ p(Y_t|\Theta, Z_t, \xi_t) &\sim N(\mu + \xi_t Z_t, \sigma^2) \\ p(Y|\Theta, Z, \xi) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t - \mu - Z_t \xi_t}{\sigma}\right)^2\right) \end{aligned}$$

Using the conjugate priors, we assume the prior distribution of the five parameters in Θ as

$$\begin{aligned} \mu &\sim N(\theta, \delta^2) \\ \sigma^2 &\sim IG(\alpha, \beta) \\ \mu_s &\sim N(\theta_s, \delta_s^2) \\ \sigma_s^2 &\sim IG(\alpha_s, \beta_s) \\ \lambda &\sim B(\gamma, \eta) \end{aligned}$$

where IG represents the inverse gamma distribution, and B is the Beta distribution with density

$$B(\gamma,\eta) = \frac{\Gamma(\gamma+\eta)}{\Gamma(\gamma)\Gamma(\eta)}\lambda^{\gamma-1}(1-\lambda)^{\eta-1}$$

The posterior distribution of σ^2 can be derived by combining the likelihood of Y_t and the prior of σ^2.

$$\begin{aligned}
p(\sigma^2|\Theta_{\sigma^2\backslash c}, Z_t, \xi_t, Y) &\propto p(Y|\Theta, Z_t, \xi_t, \sigma^2)\times p(\sigma^2) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t-\mu-Z_t\xi_t}{\sigma}\right)^2\right)\times\frac{\beta^\alpha}{\Gamma(\alpha)}(\sigma^2)^{-\alpha-1}\exp(-\beta/\sigma^2) \\
&\propto (\sigma^2)^{-T/2-\alpha-1}\exp\left(-\left[\frac{1}{2}\sum_{t=1}^{T}(Y_t-\mu-Z_t\xi_t)^2+\beta\right]/\sigma^2\right) \\
&\propto IG\left(\alpha+\frac{T}{2}, \beta+\frac{1}{2}\sum_{t=1}^{T}(Y_t-\mu-Z_t\xi_t)^2\right)
\end{aligned}$$

which yields very similar result as the Black-Scholes model. The only difference is that the return Y_t is adjusted not only by μ but also the $Z_t\xi_t$.

The posterior of μ obtains by defining $\hat{\mu} = \left[\sum_{t=1}^{T}(Y_t - Z_t\xi_t)\right]/T$.

$$\sum_{t=1}^{T}(Y_t-\mu-\xi_t Z_t)^2 = \sum_{t=1}^{T}(Y_t-\hat{\mu}-\xi_t Z_t)+T(\hat{\mu}-\mu)^2$$

Then we yield

$$\begin{aligned}
p(\mu|\Theta_{\mu\backslash c}, Z, \xi, Y) &\propto p(\mu|\sigma^2, Z, \xi, Y) \\
&= p(Y|\mu, \sigma^2, \xi, Z)p(\mu) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{Y_t-\mu-Z_t\xi_t}{\sigma}\right)^2\right)\times\frac{1}{\sqrt{2\pi\delta^2}}\exp\left(-\frac{1}{2}\left(\frac{\mu-\theta}{\delta}\right)^2\right) \\
&\propto \exp\left(-\frac{1}{2\delta^{*2}}\left(\mu-\left(\frac{T\hat{\mu}}{\sigma^2}+\frac{\theta}{\delta^2}\right)\delta^{*2}\right)^2\right) \\
&\propto N\left(\left[\sum_{t=1}^{T}(Y_t-\xi_t Z_t)/\sigma^2+\theta/\delta^2\right]\delta^{*2}, \delta^{*2}\right)
\end{aligned}$$

where $\delta^{*2} = (T/\sigma^2+1/\delta^2)^{-1}$.

Then we drive the posteriors for μ_s and σ_s^2, and they define the jump size in the return process. Analogy to the derivation of the μ and σ^2, we yield

$$\begin{aligned}
p(\sigma_s^2|\Theta_{\sigma_s^2\backslash c}, Z_t, \xi_t, Y) &\propto p(\xi|\mu_s, \sigma_s^2)\times p(\sigma_s^2) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma_s^2}}\right)^T \exp\left(-\frac{1}{2}\sum_{t=1}^{T}\left(\frac{\xi_t-\mu}{\sigma_s}\right)^2\right)\times\frac{\beta_s^{\alpha_s}}{\Gamma(\alpha_s)}(\sigma_s^2)^{-\alpha_s-1}\exp(-\beta_s/\sigma_s^2) \\
&\propto IG\left(\alpha_s+\frac{T}{2}, \beta_s+\frac{1}{2}\sum_{t=1}^{T}(\xi_t-\mu_s)^2\right)
\end{aligned}$$

And the posterior of μ_s, using similar technique, obtains

$$\begin{aligned}
p(\mu_s|\Theta_{\mu_s\backslash c}, Z, \xi, Y) &\propto p(\xi|\mu_s, \sigma_s^2)p(\mu_s) \\
&\propto N\left(\left[\frac{\sum_{t=1}^{T}\xi_t}{\sigma_s^2}+\frac{\theta_s}{\delta_s^2}\right]\delta_s^{*2}, \delta_s^{*2}\right)
\end{aligned}$$

where $\delta_s^{*2} = (T/\sigma_s^2 + 1/\delta_s^2)^{-1}$.

The jump intensity λ is conditional on the jump indicator Z. Then the posterior of λ is

$$\begin{aligned}
p(\lambda|\Theta_{\lambda\backslash c}, Z, \xi, Y) &= p(\lambda|Z) = p(Z|\lambda)p(\lambda) \\
&\propto \prod_{t=1}^{T} p(Z_t|\lambda)p(\lambda) \\
&= \prod_{t=1}^{T} \lambda^{Z_t}(1-\lambda)^{1-Z_t} p(\lambda) \\
&\propto \lambda^{\sum_{t=1}^{T} Z_t}(1-\lambda)^{T-\sum_{t=1}^{T} Z_t}\lambda^{\gamma-1}(1-\lambda)^{\eta-1} \\
&\propto B\left(\sum_{t=1}^{T} Z_t + \gamma, T - \sum_{t=1}^{T} Z_t + \eta\right)
\end{aligned}$$

The posteriors of the state variables, ξ_t and Z_t, are

$$\begin{aligned}
p(\xi_t|\Theta, Z_t, Y) &\propto p(Y_t|\Theta, Z_t, \xi_t)p(\xi_t|\Theta) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{Y_t - \mu - \xi_t Z_t}{\sigma}\right)^2 - \frac{1}{2}\left(\frac{\xi_t - \mu_s}{\sigma_s}\right)^2\right) \\
&\propto N(((Y_t - \mu)Z_t/\sigma^2 + \mu_s/\sigma^2)\sigma_t^{*2}, \sigma_t^{*2})
\end{aligned}$$

where $\sigma_t^{*2} = (Z_t/sigma^2 + 1/sigma_s^2)^{-1}$. And

$$\begin{aligned}
p(Z_t = 1|\Theta, \xi_t, Y_t) &\propto p(Y_t|\Theta, Z_t = 1, \xi_t)p(Z_t = 1|\Theta) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{Y_t - \mu - \xi_t}{\sigma}\right)^2\right)\lambda \\
p(Z_t = 0|\Theta, \xi_t, Y_t) &\propto p(Y_t|\Theta, Z_t = 0, \xi_t)p(Z_t = 0|\Theta) \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{Y_t - \mu}{\sigma}\right)^2\right)(1-\lambda)
\end{aligned}$$

and the integrating constant is determined by insuring that the two probability ($p(Z_t = 0)$ and $p(Z_t = 1)$) add up to one.

The derivations of the posteriors complete the MCMC algorithm for the Merton jump diffusion model.

3. Data and Empirical Results

3.1. **Data.** The data used for the Merton jump diffusion is the daily return of S&P500 index level over January 1954 to December 2015. The index level and the daily (log) return series are plotted in Figure 1. Given the prices of the index, the log return is calculated by

$$Y_t = \log(S_t/S_{t-1})$$

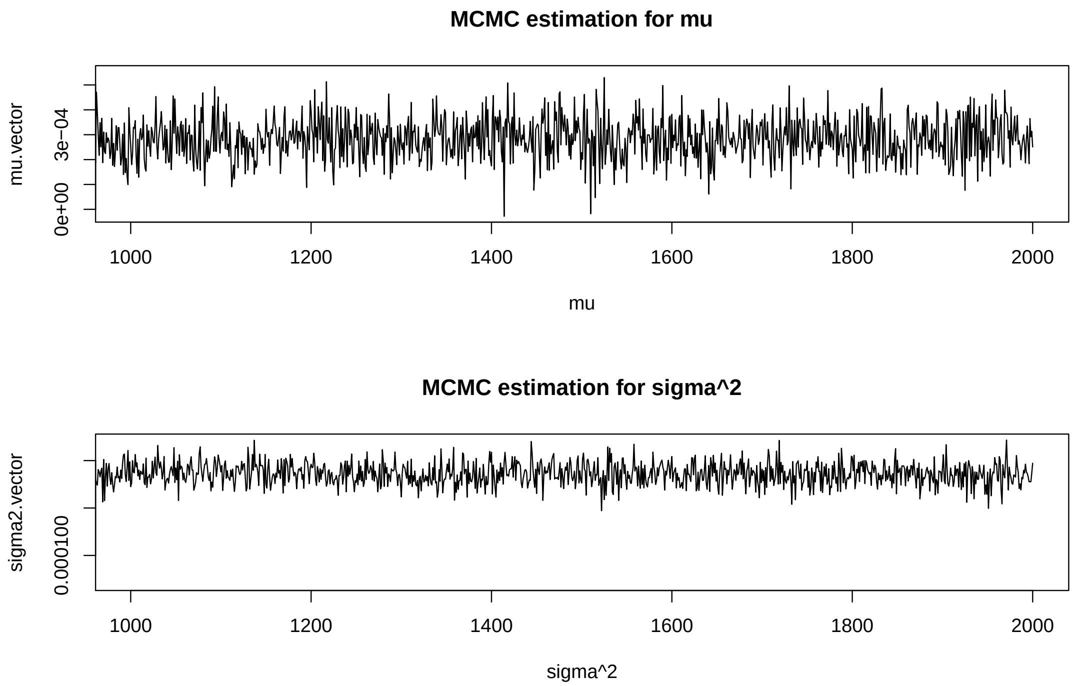

Figure 2 The μ and σ^2 with Monte Carlo draws on the Markov Chain. The MCMC algorithm is based on Gibbs sampler. The simulation size is 2000. Parameter estimation and statistical inference are conducted between steps 1001 and 2000. The first 1000 iteration are in the burn-in period.

3.2. **Estimation from Black-Scholes Model.**

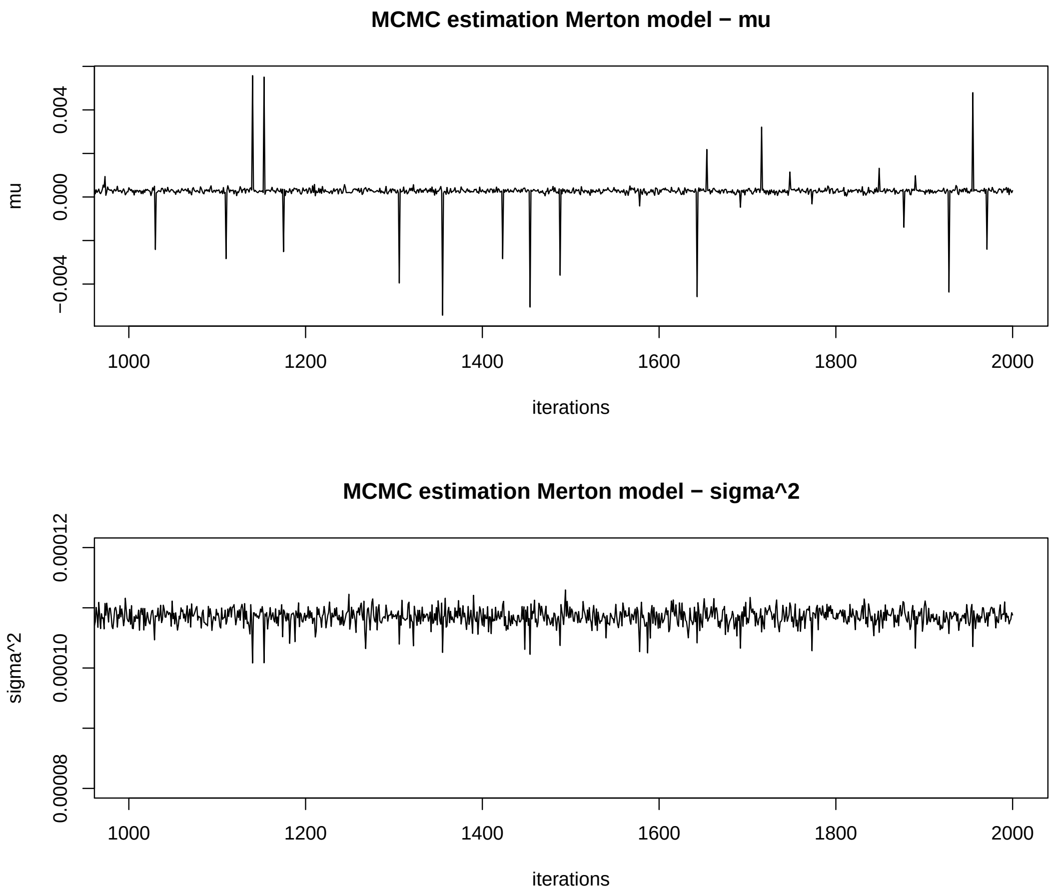

Figure 3 The μ and σ^2 with Monte Carlo draws on the Markov Chain for the Merton jump diffusion model. The MCMC algorithm is based on Gibbs sampler. The simulation size is 2000. Parameter estimation and statistical inference are conducted between steps 1001 and 2000. The first 1000 iteration are in the burn-in period.

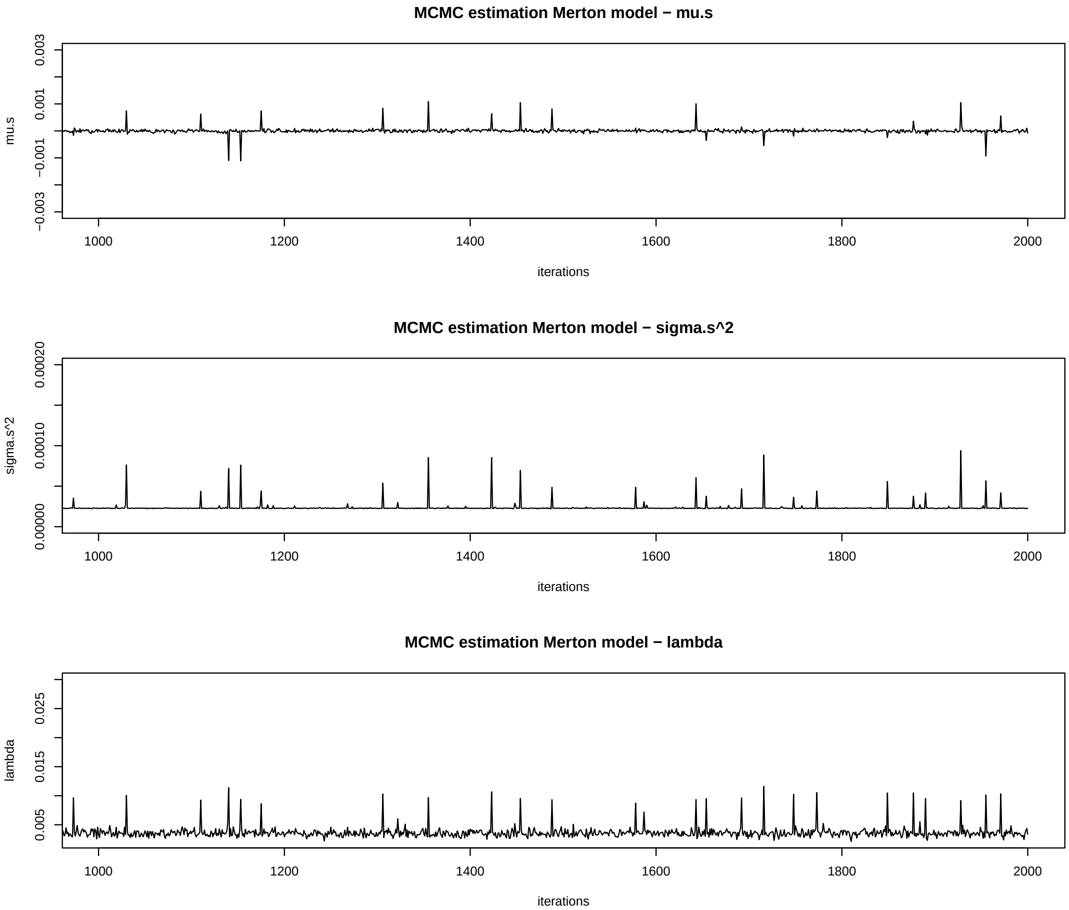

Figure 4 The μ_s, σ_s^2 and λ with Monte Carlo draws on the Markov Chain for the Merton jump diffusion model. The MCMC algorithm is based on Gibbs sampler. The simulation size is 2000. Parameter estimation and statistical inference are conducted between steps 1001 and 2000. The first 1000 iteration are in the burn-in period.

3.3. **Estimation from Black-Scholes Model.**

4. CONCLUSION

REFERENCES

[1] Black F., Scholes M. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[2] John Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. *Unpublished Manuscipt*, 1970.

[3] Jacquier, Eric, Nicholas G. Polson, and Peter Rossi. Bayesian analysis of stochastic volatility models. *Journal of Business and Economic Statistics*, 12:69–87, 1994.

[4] Johannes, Michael, and Nicholas G. Polson. Mcmc methods in financial econometrics, in yacine aït-sahalia, and lars hansen. *Handbook of Financial Econometrics (Elsevier: Oxford)*, 2002.

Appendix: Code for Black-Scholes and Merton jump diffusion models using Bayesian MCMC algorithm

```r
setwd('C:\\Users\\ranzhao\\Documents\\Empirical-Asset-Pricing\\Assignment 2')
setwd('D:\\PhD FE\\Empirical-Asset-Pricing\\Assignment 2')
setwd('D:\\Empirical-Asset-Pricing\\Assignment 2')
require(pscl)

# Data loading
spx_index_values = read.csv('spx_index_values.csv', header = TRUE)
par(mfrow=c(2,1))
plot(as.Date(as.character(spx_index_values$Date), "%m/%d/%Y"), spx_index_values$SPX.Index, type='l',
     main='SPX index levels, from 1954 to 2015',
     xlab='year', ylab='SPX index')

# calculate the return series
spx_index_values$Return = rep(0, dim(spx_index_values)[1])
spx_index_values$Return[2:length(spx_index_values$Return)] =
  log(spx_index_values$SPX.Index[2:length(spx_index_values$SPX.Index)] /
  spx_index_values$SPX.Index[1:(length(spx_index_values$SPX.Index)-1)])
data.length = length(spx_index_values$Return)

plot(as.Date(as.character(spx_index_values$Date), "%m/%d/%Y"), spx_index_values$Return, type='l',
     main='SPX index returns, from 1954 to 2015',
     xlab='year', ylab='SPX returns')

##################################################
# Black-Scholes model with Bayesian MCMC
simulation.length = 2000
mu.vector = rep(0, simulation.length)
sigma2.vector = rep(0, simulation.length)

# initialize the parameters (priors)
mu.vector[1] = mean(spx_index_values$Return)
sigma2.vector[1] = var(spx_index_values$Return)
alpha = 1000
beta = 0.2
theta = 0
delta2 = 0.001


for (i in 2:simulation.length){
  # draw mu first, does the order matter?
  delta.star.2 = 1/(data.length/sigma2.vector[i-1] + 1/delta2)
  mu.vector[i] = rnorm(1,(sum(spx_index_values$Return)/sigma2.vector[i-1]+theta/delta2)*delta.star.2, sqrt(delta.star.2))
  sigma2.vector[i] = rigamma(1,alpha+0.5*data.length, beta+0.5*sum((spx_index_values$Return - mu.vector[i])^2))
}

plot(mu.vector, type='l', xlab='mu',xlim=c(1001,2000),main='MCMC estimation for mu')
plot(sigma2.vector, type='l', xlab='sigma^2',xlim=c(1001,2000),main='MCMC estimation for sigma^2')
mu.vector.est = mu.vector[1001:2000]
sigma2.vector.est = sigma2.vector[1001:2000]
```

```
quantile.mu.vector = c(quantile(mu.vector.est, 0.025), quantile(mu.vector.est, 0.5),
    quantile(mu.vector.est, 0.975))
quantile.sigma2.vector = c(quantile(sigma2.vector.est, 0.025), quantile(sigma2.vector.
    est, 0.5), quantile(sigma2.vector.est, 0.975))

# check with existing library in r
model <- set.to.class("Diffusion", parameter = list(phi = mean(spx_index_values$Return
    ), gamma2 = var(spx_index_values$Return)))
est_diff <- estimate(model, 1:length(spx_index_values$Return), spx_index_values$Return
    , 2000)
mu.vector.est.comp = est_diff@phi[1001:2000]
sigma2.vector.est.comp = est_diff@gamma2[1001:2000]

quantile.mu.vector = c(quantile(mu.vector.est.comp, 0.025), quantile(mu.vector.est.
    comp, 0.5), quantile(mu.vector.est.comp, 0.975))
quantile.sigma2.vector = c(quantile(sigma2.vector.est.comp, 0.025), quantile(sigma2.
    vector.est.comp, 0.5), quantile(sigma2.vector.est.comp, 0.975))

##################################################


##################################################
# Jump diffusion model with Bayesian MCMC
simulation.length = 2000
merton.mu.vector = rep(0, simulation.length)
merton.sigma2.vector = rep(0, simulation.length)
merton.mu.s.vector = rep(0, simulation.length)
merton.sigma2.s.vector = rep(0, simulation.length)
merton.lambda.vector = rep(0, simulation.length)
merton.Z = rep(0, simulation.length)
merton.xi = rep(0, simulation.length)
jump.times = rep(0, simulation.length)

# initialize the parameters (priors)
merton.mu.vector[1] = mean(spx_index_values$Return)
merton.sigma2.vector[1] = var(spx_index_values$Return)
merton.mu.s.vector[1] = 0
merton.sigma2.s.vector[1] = 0.03
merton.lambda.vector[1] = 0.03
merton.Z.data = as.numeric(runif(data.length, 0, 1) < merton.lambda.vector[1])
merton.xi.data = rnorm(data.length, merton.mu.s.vector[1], sqrt(merton.sigma2.s.vector
    [1]))
merton.Z[1] = 0
merton.xi = 0
alpha = 1000
beta = 0.2
theta = 0
delta2 = 0.001
alpha.s = 1000
beta.s = 0.2
theta.s = 0
delta2.s = 0.001
gamma = 50
eta = 2

for (i in 2:simulation.length){
  # mu and sigma for the merton model
```

```r
  delta.star.2 = 1/(data.length/merton.sigma2.vector[i-1] + 1/delta2)
  merton.mu.vector[i] = rnorm(1,(sum(spx_index_values$Return-merton.Z.data*merton.xi.data)/merton.sigma2.vector[i-1]+theta/delta2)*delta.star.2, sqrt(delta.star.2))
  merton.sigma2.vector[i] = rigamma(1,alpha+0.5*data.length, beta+0.5*sum((spx_index_values$Return - merton.mu.vector[i] - merton.Z.data*merton.xi.data)^2))
  # mu and sigma for the jump size
  delta.star.s.2 = 1/(data.length/merton.sigma2.s.vector[i-1] + 1/delta2.s)
  merton.mu.s.vector[i] = rnorm(1,(sum(merton.xi.data*merton.Z.data)/merton.sigma2.vector[i-1]+theta.s/delta2.s)*delta.star.s.2, sqrt(delta.star.s.2))
  merton.sigma2.s.vector[i] = rigamma(1,alpha.s+0.5*data.length, beta.s+0.5*sum((merton.Z.data*merton.xi.data - merton.mu.s.vector[i])^2))
  # jump intensity
  merton.lambda.vector[i] = rbeta(1, max(5, sum(merton.Z.data)+gamma, data.length - sum(merton.Z.data) + eta))
  # state variable xi
  sigma.star.xi = 1/(merton.Z[i-1]/merton.sigma2.vector[i]+1/merton.sigma2.s.vector[i])
  for (j in 1:data.length){
    merton.xi.data[j] = rnorm(1, (spx_index_values$Return[j]-merton.mu.vector[i])*merton.Z.data[j]/merton.sigma2.vector[i]+merton.mu.s.vector[i]/merton.sigma2.s.vector[i], sqrt(sigma.star.xi))
    jump.ind = runif(1, 0, 1)
    if (jump.ind < exp(-0.5*(spx_index_values$Return[j]-merton.mu.vector[i]-merton.xi.data[j])^2/merton.sigma2.vector[i])){
      merton.Z.data[j] = 1
    }
    else{
      merton.Z.data[j] = 0
    }
  }
  jump.times[i] = sum(merton.Z.data)
}



par(mfrow=c(2,1))
plot(merton.mu.vector, type='l', xlab='iterations',ylab='mu',xlim=c(1001,2000),main='MCMC estimation Merton model - mu')
plot(merton.sigma2.vector, type='l', xlab='iterations',ylab='sigma^2',xlim=c(1001,2000), ylim=c(0.00008,0.00012),main='MCMC estimation Merton model - sigma^2')
par(mfrow=c(3,1))
plot(merton.mu.s.vector, type='l', xlab='iterations',ylab='mu.s',xlim=c(1001,2000), ylim=c(-0.003,0.003),main='MCMC estimation Merton model - mu.s')
plot(merton.sigma2.s.vector, type='l', xlab='iterations',ylab='sigma.s^2',xlim=c(1001,2000), ylim=c(0,0.0002),main='MCMC estimation Merton model - sigma.s^2')
plot(merton.lambda.vector, type='l', xlab='iterations',ylab='lambda',xlim=c(1001,2000),main='MCMC estimation Merton model - lambda')

merton.mu.vector.est = merton.mu.vector[1001:2000]
merton.sigma2.vector.est = merton.sigma2.vector[1001:2000]
merton.mu.s.vector.est = merton.mu.s.vector[1001:2000]
merton.sigma2.s.vector.est = merton.sigma2.s.vector[1001:2000]
merton.lambda.vector.est = merton.lambda.vector[1001:2000]

quantile.mu.vector = c(quantile(merton.mu.vector.est, 0.025), quantile(merton.mu.vector.est, 0.5), quantile(merton.mu.vector.est, 0.975))
quantile.sigma2.vector = c(quantile(merton.sigma2.vector.est, 0.025), quantile(merton.sigma2.vector.est, 0.5), quantile(merton.sigma2.vector.est, 0.975))
```

```r
quantile.mu.vector = c(quantile(merton.mu.s.vector.est, 0.025), quantile(merton.mu.s.vector.est, 0.5), quantile(merton.mu.s.vector.est, 0.975))
quantile.sigma2.vector = c(quantile(merton.sigma2.s.vector.est, 0.025), quantile(merton.sigma2.s.vector.est, 0.5), quantile(merton.sigma2.s.vector.est, 0.975))
quantile.lambda.vector = c(quantile(merton.lambda.vector.est, 0.025), quantile(merton.lambda.vector.est, 0.5), quantile(merton.lambda.vector.est, 0.975))


# check with existing library in R
# non-informative
model <- set.to.class("jumpDiffusion", Lambda = function(t, xi) (t/xi[2])^xi[1],
                      parameter = list(theta = 0.1, phi = 0.05, gamma2 = 0.1, xi = c(3, 1/4)))
est <- estimate(model, 1:length(spx_index_values$Return), spx_index_values$Return, 2000)
plot(est)

# informative
model2 <- set.to.class("jumpDiffusion", Lambda = function(t, xi) (t/xi[2])^xi[1],
                      parameter = list(theta = 0.1, phi = 0.05, gamma2 = 0.1, xi = c(3, 1/4)),
                      priorDensity = list(phi = function(phi) dnorm(phi, 0.05, 0.01),
                                          theta = function(theta) dgamma(1/theta, 10, 0.1*9),
                                          gamma2 = function(gamma2) dgamma(1/gamma2, 10, 0.1*9),
                                          xi = function(xi) dnorm(xi, c(3, 1/4), c(1,1))))
est2 <- estimate(model2, 1:length(spx_index_values$Return), spx_index_values$Return, 2000)
plot(est)
##################################################
```

assignment2.R