Comparison of Stochastic Volatility Models Using Integrated Information Criteria

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By
Yunyang Wang

Permission to Use

Abstract

Stochastic volatility (SV) models are a family of models that commonly used in the modeling of stock prices. In all SV models, volatility is treated as a stochastic time series. However, SV models are still quite different from each other from the perspective of both underlying principles and parameter layouts. Therefore, selecting the most appropriate SV model for a given set of stock price data is important in making future predictions of stock market. To achieve this goal, leave-one-out cross-validation (LOOCV) methods could be used. However, LOOCV methods are computationally expensive, thus its use is very limited in practice. In our studies of SV models, we proposed two new model-selection approaches, integrated widely applicable information criterion (iWAIC) and integrated importance sampling information criterion (iIS-IC), as alternatives to approximate LOOCV results. In iWAIC and iIS-IC methods, we first calculate the expected likelihood of each observation as an integral with respect to the corresponding latent variable (the current log-volatility parameter). Since the observations are highly correlated with their corresponding latent variable, the integrated likelihood of each t^{th} observation ($\boldsymbol{y}_t^{\text{obs}}$) is expected to approximate the expect likelihood of $\boldsymbol{y}_t^{\text{obs}}$ calculated from the model with $\boldsymbol{y}_t^{\text{obs}}$ as its holdout data. Second, the integrated expected likelihood is used, as a replacement of the expected likelihood, in the calculation of information criteria. Since the integration with respect to the latent variable largely reduces the model's bias towards the corresponding observation, the integrated information criteria are expected to approximate LOOCV results. To evaluate the performance of iWAIC and iIS-IC, we first conducted an empirical study using simulated data sets. The results from this study show that iIS-IC method has an improved performance over the traditional IS-IC, but iWAIC does not outperform the non-integrated WAIC method. A further empirical study using real-world stock market return data was subsequently carried out. According to the model-selection results, the best model for the given data is either the SV model with two independent autoregressive processes, or the SV model with nonzero expected returns.

Acknowledgements

I thank my adviser, Dr. Longhai Li, for his kindly guidance in my thesis project and the thesis writing. I thank my committee member, Dr. Juxin Liu, for her suggestions to improve my thesis. I thank my colleague, Ms. Zhouji Zheng, for her preliminary work on the project. I also thank the University of Saskatchewan and the U of S math & stats department for providing me the financial support during my studies, and my parents for their mental support on me over the years.

Contents

Permission to Use **i**

Abstract **ii**

Acknowledgements **iii**

Contents **iv**

List of Tables **vi**

List of Figures **vii**

1 Introduction **1**

1.1 Stochastic Volatility Models 1

1.2 Review of Model Comparison Methods 2

1.3 Contributions of This Thesis 6

2 The Stochastic Volatility Models and the Model-Fitting Process **9**

2.1 The Stochastic Volatility Models 9

2.2 Bayesian Inference and Markov Chain Monte Carlo Sampling for Fitting SV Models 15

3 Statistical Methods for Comparing the Stochastic Models **21**

3.1 The Cross-Validation Information Criterion 21

3.2 The Deviance Information Criterion 23

3.3 The Widely Applicable Information Criterion 24

3.4 The Importance Sampling Information Criterion 26

3.5 The Integrated WAIC and IS Criteria 27

4 Empirical Results **31**

4.1 Simulation Studies 31

4.2 An Empirical Study on S&P 100 Data 41

5 Conclusions and Discussions **47**

References **51**

Appendix A THE DEVIATION OF THE CONDITIONAL DISTRIBUTIONS OF LOG-VOLATILITIES IN THE AR-SV MODELS **55**

A.1 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in AR(1) Process 55

A.2 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in AR(2) Process 57

A.3 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in Two Independent AR(1) Processes 59

Appendix B R CODE FOR THE STOCHASTIC VOLATILITY MODELS 60
B.1 Generating Simulated Data Sets From Model 6 60
B.2 Model Specifications and Model Fittings 61

Appendix C R CODE FOR CALCULATING THE INFORMATION CRITERIA 80
C.1 Utility Functions 80
C.1.1 Sums and Means 80
C.1.2 WAIC and IS-IC 81
C.1.3 Functions Related to the Calculation of the Integrals in Model 5 . . . 82
C.2 Computing the DIC, WAIC, and IS for the SV Models 84

List of Tables

4.1 **Average Parameter $\hat{R}$ for Simulated Data** 33
4.2 **Average Parameter Estimates for Simulated Data** 37
4.2 **Average Parameter Estimates for Simulated Data** (continued) 38
4.3 **The Model Selection Results for Simulated Data** 40
4.4 **The Average CPU Time Used to Compute the Criteria for the 100 Simulated Data Sets (in Seconds)** 41
4.5 **Parameter $\hat{R}$ for S&P 100 Index Data** 43
4.6 **Parameter Estimates for S&P 100 Index Data** 44
4.6 **Parameter Estimates for S&P 100 Index Data** (continued) 45
4.7 **The DIC, nWAIC, iWAIC, nWAIC, nIS, and iIS for S&P 100 Index Data** 46
4.8 **The CPU Time Used to Compute the Criteria for S&P 100 Index Data (in Seconds)** 46

List of Figures

4.1 **Example of trace plots of parameters in model 6** 35
4.2 **Example of trace plots of ϕ and ϕ_2 in model 4 when two Markov chains converge to different modes** 36
4.3 **The plot of S&P 100 daily returns and index data from September 2010 to August 2015** 42

Chapter 1
Introduction

1.1 Stochastic Volatility Models

Stochastic volatility (SV) models are widely used in modeling stock prices, as described in journal papers written by Taylor (1982) and Hull and White (1987). In the basic stochastic volatility model, the mean-corrected daily continuously compounded returns y_t can be modeled as normal distributions with stochastic volatilities. Unlike the exponentially weighted moving average (EWMA) model and the generalized autoregressive conditional heteroskedasticity (GARCH) model, log-volatilities are treated as a Markov process in the SV model.

As a result of the Markov process, the log-volatility itself becomes a stochastic process. Therefore, SV models do not need to assume a constant volatility or a fixed volatility process like some other models (i.e., the well-known Black-Scholes model proposed by Black and Scholes (1973)). Since volatilities do change over time, the assumption of a constant volatility is a major shortcoming for many non-SV models, especially when the time horizon is long. Thus SV models are often a good alternative in the modeling of stock prices and some other derivatives with changing volatilities.

In addition to the basic model, many extended SV models are used for the purpose of stock price modeling as well, as described in the papers published by Harvey et al. (1994); Shephard (1996); Gallant and Tauchen (1996); Chernov et al. (2003).

In this thesis, eight different models were tested and compared for stock price modeling. Each of the tested models is either the basic SV model or its variation.

To sample from the posterior distributions of SV model parameters using Markov chain Monte Carlo method, we need to know a function that is proportional to the posterior distributions. To achieve this goal, Bayesian inference were used in the study. According to Bayes' rule, given the prior distribution of model parameters $\pi(\boldsymbol{\theta})$, and a set of observed data D, the posterior distribution of model parameters is proportional to the product of the posterior likelihood function of model parameters $f(D|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ and the prior distribution of model parameters:

$$f_{\text{post}}(\boldsymbol{\theta}|D) = \frac{f(D|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int f(D|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d(\boldsymbol{\theta})}, \tag{1.1}$$

where $\boldsymbol{\theta}$ are model parameters, $f_{\text{post}}(\boldsymbol{\theta}|D)$ is the posterior distribution of model parameters, and f_{post} stands for posterior density functions.

1.2 Review of Model Comparison Methods

Currently, many model comparison techniques are used to select an appropriate model for a given real data set. Since the ability to make out-of-sample predictions is a vital criterion for comparing models, a proper model-comparison method should be able to choose the model that best predicts out-of-sample data.

Naturally, cross-validation methods are used to evaluate out-of-sample predictive performance, and one of the most commonly used cross-validation method is leave-one-out cross-validation (LOOCV). In LOOCV, each validating set contains only one observation from the original data set, and the training set comprises all the other observations. When we estimate out-of-sample predictive performance with LOOCV, we hold one observation out, and test the model with the holdout observation upon the completion of the training. This training-and-testing procedure goes on until each of the observations has served as a holdout. In the end, predictions of the holdouts are compared against true observations, and a loss function is calculated as a measure of the goodness of the prediction. However, since LOOCV requires multiple rounds of model-training procedure, the method is often very computationally expensive.

Common alternatives to cross-validation approach is to measure the adjusted within-sample predictive accuracy. Since within-sample predictions of a model is always better as model-complexity increases, the goodness of within-sample predictions needS to be penalized with a model complexity measure to estimate the out-of-sample predictive performance. Therefore, in the alternative approaches, the model selection criteria are corrected by imposing a penalty for the model's complexity. As a result, comparison methods are able to select models with balanced goodness-of-fit and model complexity as an approximate of out-of-sample predictive information criterion. Examples of the adjusted within-sample predictive accuracy approach include Akaike information criterion (AIC), deviance information criterion (DIC), widely applicable information criterion (WAIC), and importance sampling information criterion (IS-IC, or just IS). Formulas of these information criteria for models without latent variables (or viewing latent variables as part of parameters $\boldsymbol{\theta}$) are given as follows. We will use y_t^{obs} to denote the actual observation of the variable y_t.

- **The Akaike Information Criterion and Deviance Information Criterion** The Akaike information criterion (AIC) has long been used to evaluate the quality of a model for a given set of data. The goodness of fit is AIC is measured by the logarithm of the maximum likelihood function for the model, and the model complexity is measured simply by the number of parameters:

  $$\text{AIC} = -2[\log(L) - k], \tag{1.2}$$

  where L is the maximal value of the likelihood function, and k is the number of parameters in the model.

  The deviance information criterion (DIC) is a widely used Bayesian model selection criterion that is closely related to the AIC. As described by Spiegelhalter et al. (2002), the general formula for DIC is given by:

  $$\text{DIC} = -2[\log f(\boldsymbol{y}^{\text{obs}}|\overline{\boldsymbol{\theta}}) - p_D], \tag{1.3}$$

  where the term $\log f(\boldsymbol{y}^{\text{obs}}|\overline{\boldsymbol{\theta}})$ measures within-sample goodness of fit, and the term p_D is the effective number of parameters (a measure of model complexity).

  The first term in the DIC formula, $\log f(\boldsymbol{y}^{\text{obs}}|\overline{\boldsymbol{\theta}})$, measures the relative goodness of fit for a given model. The $\overline{\boldsymbol{\theta}}$ is posterior expected value of model parameters, which can be calculated by the following formula:

$$\overline{\boldsymbol{\theta}} = E_{\text{post}}(\boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}). \tag{1.4}$$

On the other hand, the second term in the formula, p_D, measures effective number of parameters. The p_D can be calculated by the following formula:

$$p_D = 2[\log f(\boldsymbol{y}^{\text{obs}}|\overline{\boldsymbol{\theta}}) - E_{\text{post}}[\log f(\boldsymbol{y}^{\text{obs}}|\boldsymbol{\theta})]. \tag{1.5}$$

A larger p_D value suggests a larger number of effective model parameters, which corresponds to a higher level of model complexity. As a result, the p_D needs to be subtracted from the goodness-of-fit measurement in order to correct for model complexity.

The use of DIC is limited by a number of theoretical issues (Spiegelhalter et al., 2002, 2014). In the case of SV models, the number of latent variable increases with the increase of the sample size in latent variable models, causing non-regular likelihood-based statistical inference problems. As a result, the asymptotic justification of DIC is not validated, since the asymptotic theory of DIC is derived from regular likelihood (Gelman et al., 2014). Another major shortcoming of DIC is that the DIC is not invariant to re-parametrization. For the same model with different parametrizations, we may get different DIC values.

- **The Widely Applicable Information Criterion**

The widely applicable information criterion (WAIC) is an approximation to the cross-validation approach proposed by Watanabe (2010). In the WAIC method, goodness-of-fit is measured by calculating the summation of the log-scaled expected predictive probability density of sample points $y_1^{\text{obs}}, ..., y_n^{\text{obs}}$:

$$\sum_{t=1}^{n} \log E_{\text{post}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})], \tag{1.6}$$

where $\boldsymbol{\theta}$ is the estimated model parameters. In this formula, a higher log-probability summation value suggests a better fit of the data.

On the other hand, model-complexity is measured by effective number of parameters (p_{WAIC}):

$$p_{WAIC1} = 2\sum_{t=1}^{n}\{\log E_{\text{post}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})] - E_{\text{post}}[\log f(y_t^{\text{obs}}|\boldsymbol{\theta})]\}, \text{ or} \tag{1.7}$$

$$p_{WAIC2} = \sum_{t=1}^{n}\text{Var}_{\text{post}}[\log f(y_t^{\text{obs}}|\boldsymbol{\theta})], \tag{1.8}$$

where p_{WAIC1} is calculated by summing up the difference between the logarithm of the expected posterior probability of each observation and the expected value of the log-scaled posterior probability of each observation, and p_{WAIC2} is the sum of the variance of the posterior probability of each observation. Either one of the two can be used as an estimate of the effective number of parameters. A higher p_{WAIC} implicates that a more complicated model is used, and the model needs to be penalized more as a result.

Similar to the DIC method, the WAIC criterion takes both goodness-of-fit and model complexity into consideration:

$$\text{WAIC} = -2\sum_{t=1}^{n}\log E_{\text{post}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})] + 2p_{WAIC}. \tag{1.9}$$

In the formula above, the summation of the log-probabilities measures how well the model predicts the sample data, and effective number of parameters (p_{WAIC}) adjusts the criterion for model complexity.

The WAIC method is different from AIC and DIC methods mainly on how posterior parameter estimates are used during the calculation. In the AIC and DIC methods, likelihood functions are calculated according to point estimates of the posterior parameters, while in the WAIC method, the calculation is based on the posterior distribution of the parameters. Therefore, the WAIC approach generally works better on reflecting posterior uncertainties, and can be used as an alternative to the AIC and DIC methods when parameters are non-identifiable (Celeux et al., 2006). However, the asymptotic justification of WAIC only holds when all the observations are independent of each other, which is not the case in SV models.

- **The Importance Sampling Information Criterion**

  As described by Gelfand et al. (1992) and Gelfand (1996), the importance sampling information criterion (IS-IC) approach approximates cross-validation results by introducing a weight measurement:

$$W_t = \frac{1}{f(y_t^{\text{obs}}|\boldsymbol{\theta})}. \tag{1.10}$$

  The weight formula suggests that the higher the posterior predictive density of y_i^{obs} is, the smaller the weight is. Using this weight, the IS-IC can be calculated as:

$$\begin{aligned}\text{IS-IC} &= -2\sum_{t=1}^{n}\log\frac{E_{\text{post}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})W_t]}{E_{\text{post}}(W_t)} &(1.11)\\ &= -2\sum_{t=1}^{n}\log\frac{1}{E_{\text{post}}[1/f(y_t^{\text{obs}}|\boldsymbol{\theta})]}, &(1.12)\end{aligned}$$

  where the IS-IC is -2 times the summation of the weighted estimates of expected pointwise predictive densities.

  The IS-IC is asymptotically equivalent to LOOCV, but the criterion's approximation to LOOCV may be negatively affected if a certain observation has a large impact on the value of $\boldsymbol{\theta}$ (for example, an outlier observation) (Peruggia, 1997; Vehtari, 2001; Vehtari and Lampinen, 2002; Epifani et al., 2008; Li et al., 2015).

1.3 Contributions of This Thesis

In this thesis, we propose to compare stochastic volatility models which have latent variables (such as h_t) using regular DIC, WAIC, IS-IC methods, as well as two new approaches named integrated widely applicable information criterion (iWAIC) method and integrated importance sampling information criterion (iIS-IC, or just iIS) method. In non-integrated WAIC (nWAIC) and IS-IC (nIS-IC, or nIS) approaches, the likelihood $f(y_t^{\text{obs}}|h_t,\boldsymbol{\theta})$ is computed conditional on the full set of fitted model parameters and latent variables h_t. However, unlike LOOCV, nWAIC and nIS are calculated according to models fitted with entire data set. Therefore, the $f(y_t^{\text{obs}}|h_t, \boldsymbol{\theta})$ might be biased towards y_t^{obs} and thus cannot be considered as out-of-sample likelihood. In our particular case of SV models, estimated log-volatility parameters, h_t, are strongly correlated with corresponding y_t^{obs}. As a result, the bias of the likelihood $f(y_t^{\text{obs}}|h_t, \boldsymbol{\theta})$ could be greatly reduced if direct uses of h_t is avoided.

If we denote h_{-t} as all the estimated log-volatilities except h_t, and $\boldsymbol{\theta}$ as all the other parameters, then this integrated likelihood $f(y_t^{\text{obs}}|h_{-t}, \boldsymbol{\theta})$ could be calculated according to the following integral:

$$f(y_t^{\text{obs}}|h_{-t}, \boldsymbol{\theta}) = \int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) f(h_t|\boldsymbol{\theta}, h_{-t}) dh_t, \tag{1.13}$$

where $t = 1, ..., n$, and n is total number of observations.

Once $f(y_t^{\text{obs}}|h_{-t}, \boldsymbol{\theta})$ is calculated, the integrated likelihood could be plugged into the nWAIC and nIS formulas (in replacement of $f(y_t^{\text{obs}}|h_t, \boldsymbol{\theta})$) to calculate the iWAIC and iIS values. In our studies, we proposed integrated iWAIC and iIS methods as potential alternatives to the nWAIC and nIS. Therefore, the performance of the two newly proposed model selection methods needs to be evaluated as well. In order to make comparisons, another popular approach, DIC, was also included in the testing process.

In our studies, DIC, nWAIC, nIS, iWAIC, and iIS criteria are studied for their performance in model selection. The model selection methods were used to select models that best fit the given data sets. In the first study, data sets were firstly generated from an SV model, and the generated data sets were subsequently fitted into eight candidate SV models. The data-generating model is one of the candidate models, and we choose to use eight different models so that it is unlikely that a certain model-selection criterion would favor the correct model just by chance. In the end, the models were compared by applying DIC, nWAIC, nIS, iWAIC, and iIS methods. Through this simulation study, we have found that all the tested information criteria are able to select the correct model (in this case, the data-generating model) most of the time, and the integration does improve the performance of IS method. In the second study, a real set of stock market return data (S&P 100) was respectively fitted into the eight different SV models, and DIC, nWAIC, nIS, iWAIC, and iIS criteria were used to determine the best model for the given data set. The results of the real-world stock data indicate that the best model for the given data is either the SV model with two independent autoregressive processes of h_t, or the SV model with a non-zero expected return.

Chapter 2

The Stochastic Volatility Models and the Model-Fitting Process

2.1 The Stochastic Volatility Models

The price of a corporation stock is determined by the entity's capability to generate future cash flows, and is also affected by the stock's supply and demand. If we make an investment on a certain stock, then the profit of the investment on the stock over a period of time is called the stock's rate of return. In practice, the return of a stock is closely related to the stock's volatility. If y_t is continuously compounded rate of return, then the relationship between the two can be modeled by the following formula:

$$y_t|\sigma_t \sim N(0, \sigma_t^2), t = 1, ..., n, \tag{2.1}$$

where σ_t^2 is the corresponding price volatility.

The stock price volatility is a measure of expected magnitude of the change of prices (up or down) of an underlying asset, which is a very important feature of a stock. The volatility for a given stock is essential in predicting the price of a stock itself, as well as many other stock-related derivatives. For example, according to the famous Black-Scholes model, a European call option on a given stock (with the same strike price and expiration) requires more premium (more valuable) when the implied volatilities of the underlying stock is higher (Black and Scholes, 1973). In addition, from the risk management point of view, volatilities of stocks are needed to determine the value at risk (VaR) of a portfolio (Giot and Laurent, 2004).

Traditional approaches such as historical simulation may not recognize changes in volatility, and generalized autoregressive conditional heteroskedasticity (GARCH) models are often used to forecast future volatilities as a result (Engle, 1982; Bollerslev, 1986). For example, in the GARCH (1, 1) model, volatilities σ_t^2 are calculated according to the following formula:

$$\sigma_t^2 = \omega + \alpha y_{t-1}^2 + \beta \sigma_{t-1}^2, t = 1, ..., n, \tag{2.2}$$

where $\omega = \rho V_L$ is weighted long-run variance (V_L is long-run variance, and ρ is its weight), αy_{t-1}^2 is weighted previous period's return (y_{t-1} is previous period's return), and α is its weight), and $\beta \sigma_{t-1}^2$ is weighted previous volatility estimate (β is the weight). This model becomes popular in estimating volatilities due to its simplicity and some theoretical justifications. However, in practice, the estimated parameters often renders the model unusable. For example, the summation of the estimated weights assigned to y_{t-1}^2 and σ_{t-1}^2, $\alpha + \beta$, often exceeds one. In this situation, the volatility is not a stationary process.

Stochastic volatility (SV) models are alternatives to GARCH models in the modeling of stock price volatilities (Taylor, 1982; Hull and White, 1987). In SV models, volatility is considered as a random process. By allowing randomness in the process, SV models have more theoretical benefits. In this study, we tested several autoregressive stochastic volatility (AR-SV) models, which is a popular sub-category of SV models. In basic AR-SV models, the logarithm of volatilities , $h_t = log(\sigma_t)$, are modeled as a stochastic autoregressive process:

$$h_t = \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, \tag{2.3}$$

which can also be written as:

$$h_t = (1 - \phi)\mu + \phi h_{t-1} + v_t, t = 1, ..., n, \tag{2.4}$$

where μ is long-run volatility, h_{t-1} is previous volatility, and v_t is normally distributed with mean 0 and variance τ^2. If the weight assigned to previous log-volatility is between zero and one, then h_t is a stationary process. This formula also shows that the long-run mean of log-volatilities is evaluated as μ, indicating h_t is mean reverting. This mean-reverting property of h_t is consistent with many empirical studies. In addition, the equation suggests that current log-volatility is dependent on the previous log-volatility estimate, which satisfies the volatility-clustering feature of stock price (high volatility is typically followed by another high volatility, and vise versa). Furthermore, the introduction of v_t suggests that log-volatility is not a fixed function of its previous estimate, but also has its own randomness.

Given log-volatilities, daily stock returns y_t can be modeled as:

$$y_t = \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, \tag{2.5}$$

where u_t follows a normal distribution with mean zero and variance $\boldsymbol{\tau}^2$, and is independent of v_t.

In the literature, various AR-SV models have been proposed. In this study we consider the following eight plausible AR-SV models, as described and summarized by Berg et al. (2004).

- **Model 1**

  This model is the basic AR-SV model as we mentioned previously. The state equation governing the log-volatility process is given by:

  $$h_t = \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, \tag{2.6}$$

  and the observation equation equation of daily return is:

  $$y_t = \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, \tag{2.7}$$

  where u_t is a standard normal distribution, $v_t \sim N(0, \tau^2)$, and v_t and u_t are independent of each other.

- **Model 2**

  Model 2 is a variation of the basic SV model. In this model, the state equation of the log-volatilities is the same with the basic AR-SV model, but the mean of daily returns

y_t is α (nonzero) instead of zero:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, && (2.8) \\ y_t &= \alpha + \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, && (2.9) \end{aligned}$$

where u_t is a standard normal distribution, $v_t \sim N(0, \tau^2)$, and v_t and u_t are independent of each other. In practice, it makes sense to assume that the expected daily return is not zero. According to the "trade-off between risk and reward" principle, the more risk taken, the greater the potential reward (Lundblad, 2007). Therefore, mean returns on stocks should be at least not less than the risk-free rate.

- **Model 3**

In this model, log-volatilities h_t follow an AR(2) process:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + \psi(h_{t-2} - \mu) + v_t, t = 1, ..., n, && (2.10) \\ y_t &= \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, && (2.11) \end{aligned}$$

where u_t is a standard normal distribution, $v_t \sim N(0, \tau^2)$, and v_t and u_t are independent of each other.

This equation is best used to model a log-volatility process that has a lower autocorrelation with lag-1 log-volatility. According to the Yule-Walker equation (Cheng, 2005), for any h_t in this AR(2) process, lag-1 autocorrelation (the correlation between h_t and h_{t-1}) is the coefficient of h_{t-1}, which is ϕ. On the other hand, the lag-n autocorrelation (the correlation between h_t and h_{t-n}) is given by $\phi^n + \psi^{n-1}$. Therefore, the model indicates that current log-volatility is less correlated with its lag-1 log-volatility, but is more correlated with all the other lagged log-volatilities.

- **Model 4**

The model consists of two independent AR(1) processes, as described by Harvey et al.

(1994), Shephard (1996), and Chernov et al. (2003):

$$
\begin{aligned}
h_t^{(1)} &= \phi h_{t-1}^{(1)} + v_t^{(1)}, t = 1, ..., n, && (2.12)\\
h_t^{(2)} &= \phi_2 h_{t-1}^{(2)} + v_t^{(2)}, t = 1, ..., n, && (2.13)\\
y_t|h_t &= \exp\left(\frac{\mu}{2} + \frac{h_t^{(1)}}{2} + \frac{h_t^{(2)}}{2}\right) u_t, t = 1, ..., n, && (2.14)
\end{aligned}
$$

where u_t is a standard normal distribution, and $v_t^{(1)} \sim N(0, \tau^2)$, $v_t^{(2)} \sim N(0, \tau^2)$. Besides, $v_t^{(1)}$, $v_t^{(2)}$, and u_t are all independent of each other.

In this model, the log-volatility h_t is given by $\mu + h_t^{(1)} + h_t^{(2)}$, with $h_t^{(1)}$ and $h_t^{(2)}$ being two independent AR(1) processes.

- **Model 5**

  Model 5 allows a correlation between u_t and v_{t+1}, which causes an asymmetric effect of y_t. This correlation between u_t and v_{t+1} has long been noticed by Black (1976), and Engle and Ng (1993). In a previous study completed by Engle and Ng (1993), it was found that return shocks has an impact on volatility. As a result, it is reasonable to assume a correlation between the two. In model 5, the correlation is described by the following covariance matrix:

$$
\begin{bmatrix} u_t \\ v_{t+1} \end{bmatrix} \sim N(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho\tau \\ \rho\tau & \tau^2 \end{bmatrix}) \qquad (2.15)
$$

  Therefore, the SV model equation and the state equation of h_t can be written as:

$$
\begin{aligned}
h_t &= \mu + \phi(h_{t-1} - \mu) \\
&\quad + \rho\tau \exp(-0.5 h_{t-1}) y_{t-1} \\
&\quad + \tau\sqrt{1-\rho^2} w_t, t = 1, ..., n, && (2.16)\\
y_t|h_t &= \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, && (2.17)
\end{aligned}
$$

  where u_t and w_t are both independent normal distributions with mean 0 and unit variance.

- **Model 6**

  In this model, a jump component (an additional random upward or downward movement of an observation) is included in observation equation. Besides, y_t is also affected by its lagged observation y_{t-1}:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, && (2.18) \\ y_t &= \beta y_{t-1} + s_t q_t + \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n, && (2.19) \end{aligned}$$

  where v_t and u_t are independently distributed with $v_t \sim N(0, \tau^2)$ and $u_t \sim N(0, 1)$. In addition, another parameter s_t with the distribution of $\log(1 + s_t) \sim N(-\frac{\delta^2}{2}, \delta^2)$ measures jump sizes, and $q_t \sim \text{Bern}(\kappa)$ is the probability of the occurrence of jumps. The β parameter is a measurement of the sensitivity of the current observation (y_t) to the previous observation (y_{t-1}).

  In general, this model suggests that the current return y_t is determined by the current price volatility, the occurrence of a random jump, and the previous observation y_{t-1}.

- **Model 7**

  Similar to model 6, model 7 also includes a jump component, but not the previous observations:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, && (2.20) \\ y_t &= s_t q_t + \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n. && (2.21) \end{aligned}$$

  The distributions of all the parameters in model 7 are identical to the ones in model 6.

- **Model 8**

  To obtain this model, Gaussian observation errors in the observation equation are replaced by Student t distributions with $\boldsymbol{\nu}$ degrees of freedom:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, && (2.22) \\ y_t &= \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n. && (2.23) \end{aligned}$$

where $u_t \sim t_\nu$ is a student t distribution with ν degrees of freedom, $v_t \sim N(0, \tau^2)$, and v_t and u_t are independent of each other. This model assumes the distribution of daily return y_t has a heavier tail comparing to the basic model.

Since the errors are symmetric and nonnormal, scale-mixtures of normals could be used for model fitting according to Andrews and Mallows (1974):

$$\begin{aligned} y_t &\sim N\left(0, \frac{\exp(h_t)}{w_t}\right), t = 1, ..., n, \\ w_t &\sim \frac{1}{\nu}\chi^2_\nu = \text{Gamma}\left(\frac{\nu}{2}, \frac{\nu}{2}\right), t = 1, ..., n. \end{aligned} \tag{2.24}$$

2.2 Bayesian Inference and Markov Chain Monte Carlo Sampling for Fitting SV Models

It is rather difficult to apply classical statistical inferences, such as maximum likelihood estimation, to SV models due to the nonanalytic form of the likelihood function. To overcome this problem, several alternative approaches have been proposed. For example, in quasi-maximum likelihood method proposed by Harvey et al. (1994), an approximation of the actual likelihood function is obtained by considering the distribution of $\log(y_t)$ as a normal distribution. This approximated function (quasi-maximum likelihood function) is then maximized instead of the actual likelihood function.

In another approach called efficient method of moments (EMM), the derivative of quasi-likelihood function is used as the moment condition of generalized method of moments (GMM). The EMM-estimated parameters are then calculated by minimizing the norm of the moment condition. By using this moment condition instead of selecting a few low order moments on an *ad hoc* basis, EMM method is found to be more efficient (Andersen et al., 1999).

In our studies, we used Bayesian inference for SV models. According to Bayes' rule, given the prior distributions of model parameters $\pi(\boldsymbol{\theta}, \boldsymbol{h})$ and the observed data $\boldsymbol{y}^{\text{obs}}$, the posterior distributions of model parameters can be expressed as:

$$f_{\text{post}}(\boldsymbol{\theta}, \boldsymbol{h}|\boldsymbol{y}^{\text{obs}}) = \frac{f(\boldsymbol{y}^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h})\pi(\boldsymbol{\theta}, \boldsymbol{h})}{\int f(\boldsymbol{y}^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h})\pi(\boldsymbol{\theta}, \boldsymbol{h})d(\boldsymbol{\theta}, \boldsymbol{h})}, \tag{2.25}$$

where $f(\boldsymbol{y}^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h})$ is the likelihood function of model parameters $(\boldsymbol{\theta}, \boldsymbol{h})$ given the observed data $\boldsymbol{y}^{\text{obs}}$, and the integral in the denominator calculates the normalizing constant. Since it is usually very hard to calculate the integral, we apply MCMC sampling technique to sample from the posterior distributions of model parameters based on the numerator function $f(y^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h})\pi(\boldsymbol{\theta}, \boldsymbol{h})$, which is proportional to the posterior density function of model parameters. This Bayesian approach, together with MCMC technique, is more efficient than non-likelihood methods in estimating parameters of SV models (Jacquier et al., 2012). A more recently developed MCMC technique, Hamiltonian Monte Carlo method, is especially suitable for SV models due to its better handling of correlated model parameters h_t (Carpenter et al., 2015).

In order to fit the models to a given data set, we used Markov chain Monte Carlo (MCMC) method to sample from the posterior distributions of parameters in each model. In an MCMC process, model parameters were sampled according to a Markov chain. The Markov chain is a random process that undergoes state transitions in a given state space. Given a finite state space, a Markov chain is bound to reach a stationary state (invariant distribution) when the chain is long enough (Gilks, 2005).

In MCMC, a Markov chain is designed to sample model parameter(s) so that the equilibrium distribution(s) of the Markov chain is the posterior distribution of the parameter(s). Several algorithms have been developed to accomplish this goal. One of the most commonly used MCMC methods is Metropolis-Hastings algorithm proposed by Metropolis et al. (1953). This algorithm starts with a non-normalized posterior probability density function $f(\boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}})$ ($\boldsymbol{\theta}$ is the parameter of interest), an arbitrary initial value $\boldsymbol{\theta}_0$, and an arbitrary transition probability density function $Q(a|b)$. In the first step of the Metropolis-Hastings algorithm, a new candidate $\boldsymbol{\theta}_i$ is generated according to the probability density function of $Q(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1})$, where i is the iteration number. In the following step, an acceptance rate $\alpha = \frac{f(\boldsymbol{\theta}_i|\boldsymbol{y}^{\text{obs}})Q(\boldsymbol{\theta}_{i-1}|\boldsymbol{\theta}_i)}{f(\boldsymbol{\theta}_{i-1}|\boldsymbol{y}^{\text{obs}})Q(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1})}$ is calculated to determine if the newly-proposed $\boldsymbol{\theta}_i$ should be accepted in the new iteration.

Finally, we need to determine whether to accept the newly-proposed $\boldsymbol{\theta}_i$. If α is greater than one, the $\boldsymbol{\theta}_i$ would be automatically accepted; if α is between zero and one, the $\boldsymbol{\theta}_i$ will be accepted with a probability of α (if rejected, set $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1}$). These steps will be repeated until we have collected necessary samples.

Although the Metropolis-Hastings algorithm could theoretically provide samples from a posterior distribution in almost any situation, the method might not work well in some practical problems, for example the sampling for SV models described in section 2.1. Since the posterior density of $\boldsymbol{\theta}$ does not change over the Metropolis-Hastings iterations, it is quite likely that the sampling process gets stuck in a low-density region of $\boldsymbol{\theta}$. For example, if the $\boldsymbol{\theta}_i$ in the current iteration is far away from the high-density region, and most of the newly-proposed $\boldsymbol{\theta}_{i+1}$ happen to have a very low acceptance rate α over the current $\boldsymbol{\theta}_i$, then the Markov chain is highly likely to be stuck at the $\boldsymbol{\theta}_i$ for quite a long time. If the Markov chain gets stuck in the low-density region, we may observe an extremely slow convergence of the chain. This convergence failure greatly limits the use of the Metropolis-Hastings method, especially when the number of parameter is large.

To overcome the convergence problem of the Metropolis-Hastings algorithm, the Hamiltonian Monte Carlo (HMC) method was developed for posterior sampling (Alder and Wainwright, 1959; Andersen, 1980; Neal et al., 2011). The HMC method makes use of the concept of a canonical distribution by replacing the energy function with the Hamiltonian energy function $H(\boldsymbol{q}, \boldsymbol{p})$:

$$\begin{aligned} P(\boldsymbol{q}, \boldsymbol{p}) &= \frac{1}{Z}\exp[-H(\boldsymbol{q}, \boldsymbol{p})] \\ &= \frac{1}{Z}\exp[-U(\boldsymbol{q})]\exp[-K(\boldsymbol{p})] \\ &= C\exp[-U(\boldsymbol{q})]\exp[-K(\boldsymbol{p})], \end{aligned} \tag{2.26}$$

where Z and C are normalizing constants, $\boldsymbol{q}$ is the position variable corresponding to the potential energy, and $\boldsymbol{p}$ is the momentum variable that determines the kinetic energy. The application of canonical distribution is to ensure a statistical equilibrium (steady state) in the long run, since the mechanical system will reach a thermal equilibrium with a heat bath eventually. To sample from the posterior distribution of parameter $\boldsymbol{q}$, we define the following position function:

$$U(\boldsymbol{q}) = -\log g(\boldsymbol{q}|\boldsymbol{y}^{\text{obs}}), \tag{2.27}$$

where $g(\boldsymbol{q}|\boldsymbol{y}^{\text{obs}})$ is non-normalized posterior probability density function of $f(\boldsymbol{q}|\boldsymbol{y}^{\text{obs}})$, which could be easily obtained according to the Bayes rule. In our studies of the SV models, $\boldsymbol{q}$ is $(\boldsymbol{\theta}, h_1, \ldots, h_n)$

On the other hand, the momentum variable $\boldsymbol{p}$ is used as an auxiliary variable to facilitate the sampling of our target distribution of $\boldsymbol{q}$. If we want to sample $\{\boldsymbol{q}_1, ..., \boldsymbol{q}_d\}$ from the target distribution of $f(\boldsymbol{q}|\boldsymbol{y}^{\text{obs}})$, we would need to assign d auxiliary variables $\{\boldsymbol{p}_1, ..., \boldsymbol{p}_d\}$ as well. The $\boldsymbol{p}$ is usually defined as a zero-mean, unit variance multivariate normal distribution independent of each other and the position variable $\boldsymbol{q}$. Therefore, the kinetic function $K(\boldsymbol{p})$ can be expressed as:

$$K(\boldsymbol{p}) = \sum_{i=1}^{d} \frac{1}{2} p_i^2 \tag{2.28}$$

We are able to use the HMC method to draw sample from the posterior thanks to three fine properties of the Hamiltonian dynamics: conserved total energy, volume preservation, and time reversible. First, since the Hamiltonian dynamics is defined by the following differential equations:

$$\frac{dq_i}{d\boldsymbol{\tau}} = \frac{\partial H}{\partial p_i} = p_i; \tag{2.29}$$

$$\frac{dp_i}{d\boldsymbol{\tau}} = -\frac{\partial H}{\partial q_i} = -\frac{\partial U}{\partial q_i}, \tag{2.30}$$

where $\boldsymbol{\tau}$ is the physical time. We can easily tell that the Hamiltonian dynamics is conserved over time:

$$\frac{dH}{d\boldsymbol{\tau}} = \sum_{i=1}^{d} \left[\frac{dq_i}{d\boldsymbol{\tau}} \frac{\partial H}{\partial q_i} + \frac{dp_i}{d\boldsymbol{\tau}} \frac{\partial H}{\partial p_i} \right] = \sum_{i=1}^{d} \left[\frac{\partial H}{\partial p_i} \frac{\partial H}{\partial q_i} - \frac{\partial H}{\partial q_i} \frac{\partial H}{\partial p_i} \right] = 0. \tag{2.31}$$

The conserved total energy indicates the Hamiltonian function H is invariant. Therefore, when we use the Hamiltonian dynamics to propose a new state in the Metropolis updates, the acceptance rate is always one. This high acceptance is desirable in the Metropolis method. However, in practice, we are only able to approximate the Hamiltonian dynamics using multiple steps of discrete transformation functions (e.g., the leapfrog method), which makes the actual acceptance rate a little bit smaller than one.

Besides, the Hamiltonian dynamics also preserves the volume after each transformation. According to Liouville's theorem, the phase space is conserved under the dynamics. This volume preservation property is important, since constantly changing phase spaces can distort the sampling process.

Thirdly, the time reversible property of the Hamiltonian dynamics comes from the way the transformation (denoted by T_s) works. Since the mapping from $(\boldsymbol{q}(\boldsymbol{\tau}), \boldsymbol{p}(\boldsymbol{\tau}))$ to $(\boldsymbol{q}(\boldsymbol{\tau}+s), \boldsymbol{p}(\boldsymbol{\tau}+s))$ is one-to-one according to the T_s, the backward mapping from $(\boldsymbol{q}(\boldsymbol{\tau}+s), -\boldsymbol{p}(\boldsymbol{\tau}+s))$ to $(\boldsymbol{q}(\boldsymbol{\tau}), -\boldsymbol{p}(\boldsymbol{\tau}))$ can be achieved through T_{-s}. This time reversible property suggests the transformation leaves the target distribution invariant. As a result, the transformation would provide a random sample from the target distribution.

One common approach of the Hamiltonian dynamics is through the application of multiple steps of leapfrog transformation. In each single leapfrog jump, we start at a given state of $(\boldsymbol{p}(\boldsymbol{\tau}), \boldsymbol{q}(\boldsymbol{\tau}))$ at time $\boldsymbol{\tau}$, and after an arbitrary period of time $\boldsymbol{\epsilon}$, the new state $(\boldsymbol{p}(\boldsymbol{\tau}+\boldsymbol{\epsilon}), \boldsymbol{q}(\boldsymbol{\tau}+\boldsymbol{\epsilon}))$ is calculated by the following formulas:

$$\begin{aligned}
\hat{p}_i\left(\boldsymbol{\tau}+\frac{\boldsymbol{\epsilon}}{2}\right) &= \hat{p}_i(\boldsymbol{\tau}) - \frac{\boldsymbol{\epsilon}}{2}\frac{\partial U}{\partial q_i}(\hat{\boldsymbol{q}}(\boldsymbol{\tau})); && (2.32)\\
\hat{q}_i(\boldsymbol{\tau}+\boldsymbol{\epsilon}) &= \hat{q}_i(\boldsymbol{\tau}) + \boldsymbol{\epsilon}\hat{p}_i\left(\boldsymbol{\tau}+\frac{\boldsymbol{\epsilon}}{2}\right); && (2.33)\\
\hat{p}_i(\boldsymbol{\tau}+\boldsymbol{\epsilon}) &= \hat{p}_i\left(\boldsymbol{\tau}+\frac{\boldsymbol{\epsilon}}{2}\right) - \frac{\boldsymbol{\epsilon}}{2}\frac{\partial U}{\partial q_i}(\hat{\boldsymbol{q}}(\boldsymbol{\tau}+\boldsymbol{\epsilon})). && (2.34)
\end{aligned}$$

Since a series of leapfrog transformations is an approximation to the Hamiltonian dynamics, the energy function H is not totally invariant. To compensate for this, we would accept the newly proposed state $(\boldsymbol{p}', \boldsymbol{q}')$ with a Metropolis acceptance rate instead of one.

Once we define the position function, the kinetic energy function and the Hamiltonian dynamics, we can start the HMC sampling process with a current state q_0 by repeating a three-step iteration. In the first step of the iteration, we randomly select $-p_0$ from their independent standard normal distributions, and then we negate the variables to get p_0. In the second step, through the application of L leapfrog steps with an arbitrary time period of $\boldsymbol{\epsilon}$ each, a new (p', q') is proposed from the current state (p_0, q_0). In the end, the newly-proposed

(p', q') is accepted with a probability of $\min(1, \exp[-H(q', p') + H(\boldsymbol{q}, -p)])$. If rejected, the next state would be the same as the current state. However, no matter whether the new state is accepted or not, the momentum variable $\boldsymbol{p}$ would always be re-generated from its independent multivariate normal distribution at the beginning of each iteration.

In the HMC sampling process, the choice of the leapfrog step size parameter $\boldsymbol{\epsilon}$ and the trajectory length parameter L is of paramount importance to the performance of the HMC sampling. In order to best approximate a continuous Hamiltonian dynamics process and thus avoid a high rejection rate, the step size parameter $\boldsymbol{\epsilon}$ cannot be too large. However, an $\boldsymbol{\epsilon}$ that's too small might slow down the trajectory too much. The trajectory length parameter, L, on the other hand, also needs to be carefully chosen to minimize auto-correlations between different samples. In other words, the ideal choice of L should be able to maximize the the difference between q_0 and q'.

In our studies, the model-fitting process was completed through the application of rstan package, which is an R package based on stan sampler (Guo et al., 2016). The stan is a recently developed MCMC sampler using the no-U-turn (NUT) HMC algorithm, which is a modified HMC method that with a default scheme for choosing L (Carpenter et al., 2015; Gelman et al., 2015). In the NUT sampling, the trajectory will automatically stop once the next single leapfrog jump would make q' closer to q_0. Through the application of the NUT HMC algorithm, the stan sampler generally works much better than traditional samplers when dealing with models with a large number of parameters. As a result, we used the rstan package in our studies to make posterior parameter sampling for our SV models.

Chapter 3

Statistical Methods for Comparing the Stochastic Models

Model selection is important in the study of stock market data and forecasting future trends. By using the correct model, the property of the data is better understood and interpreted, thus better predictions and estimations can be made. Using wrong models in practice, on the other hand, may lead to unexpected losses that could be prevented.

Traditional methods, including the mean squared error (MSE) and the coefficient of determination (R^2), only measure the fit of the data to the models. Since adding additional parameters into a model typically increases the goodness-of-fit, these methods tend to favor complex models that may over-fit the data. To overcome the over-fitting problem, cross-validation methods are introduced. The cross-validation methods involve partitioning the data set into two subsets, fitting the model with one subset, and testing the model with the other subset. Although the cross-validation methods seem to be able to fully address the over-fitting problem, these methods are time-consuming and costly. Alternatively, many methods impose penalties for model complexity.

3.1 The Cross-Validation Information Criterion

Among all cross-validation methods, leave-one-out-cross-validation (LOOCV) is usually used to minimize the number of tests to run. In addition, LOOCV makes better use of the given data set than any other cross-validation method. In LOOCV, the information criterion (cross-validation information criterion, or CVIC) is computed by the following formula:

$$\text{CVIC} = -2\sum_{t=1}^{n} \log f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}}). \tag{3.1}$$

In this equation, the information criterion is calculated as -2 times the sum of $\log f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}})$. The bigger the CVIC value, the worse the fit.

Applied to SV models, the LOOCV posterior predictive density of the t^{th} observation, $f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}})$, can be calculated by the following equation:

$$f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}}) = \iint f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) f(\boldsymbol{\theta}, \boldsymbol{h}_{1:n}|\boldsymbol{y}_{-t}^{\text{obs}}) d\boldsymbol{\theta} d\boldsymbol{h}_{1:n}. \tag{3.2}$$

In the equation above, h_j is log-volatility parameter corresponding to the j^{th} observation (y_j^{obs}), and $\boldsymbol{\theta}$ are all the other non-log-volatility model parameters. In addition, $\boldsymbol{y}_{-t}^{\text{obs}}$ and $\boldsymbol{h}_{-t}$ represent all observations and latent variables except y_t^{obs} and h_t, respectively. The equation (3.2) calculates $f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}})$ as the expected value of $f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)$, where the distributions of $\boldsymbol{\theta}$ and h_t could be obtained according to the data set $\boldsymbol{y}_{-t}^{\text{obs}}$. If we assume that the structural information of the log-volatility series ($\boldsymbol{h}_{1:n}$) is preserved, then MCMC samples of $\boldsymbol{\theta}, \boldsymbol{h}_{1:n}|\boldsymbol{y}_{-t}^{\text{obs}}$ can be drawn according to the following joint posterior distribution:

$$f(\boldsymbol{\theta}, \boldsymbol{h}_{1:n}|\boldsymbol{y}_{-t}^{\text{obs}}) \propto \prod_{j \neq t} f(y_j^{\text{obs}}|h_j, \boldsymbol{\theta}) f(\boldsymbol{h}_{1:n}|\boldsymbol{\theta}) f(\boldsymbol{\theta}), \tag{3.3}$$

where $f(y_j^{\text{obs}}|h_j, \boldsymbol{\theta})$ is the probability density of y_j^{obs} conditional on h_j and $\boldsymbol{\theta}$, $f(\boldsymbol{h}_{1:n}|\boldsymbol{\theta})$ is the probability density of $\boldsymbol{h}_{1:n}$ conditional on $\boldsymbol{\theta}$, and $f(\boldsymbol{\theta})$ is the prior distribution of $\boldsymbol{\theta}$. Once we obtain the sample of the parameters,$f(y_t^{\text{obs}}|\boldsymbol{y}_{-t}^{\text{obs}})$ could be readily calculated according to equation (3.2) by approximating the integral with averaging over MCMC samples. Finally, the CVIC values of the SV models could be subsequently computed by formula (3.1) for each $t = 1, \ldots, T$.

However, when the model fitting process is time consuming, even LOOCV method might become too expensive. In LOOCV, we need to simulate at least T Markov chains. In our study of SV models, the time for a computer to finish running a single Markov chain ranges from hours to several days, which is extremely time-consuming. Therefore, non-exhaust cross-validation methods are sometimes used to approximate exhaustive methods to further reduce the computation. Instead of testing all possible combinations of training sets and validating sets, non-exhaustive methods only use a finite number of randomly selected combinations.

In general, cross-validation approach is effective in dealing with optimistic bias problem, but the great cost of computation limits its use. In our studies, the number of observations in each data set is big, and the model fitting process is extremely time consuming (typically from hours to days). Therefore, cross-validation approach is not a good option. Instead, several other methods (DIC, WAIC, IS) were used to approximate cross-validation information criterion.

3.2 The Deviance Information Criterion

The deviance information criterion (DIC) is often used for Bayesian model selections (Spiegelhalter et al., 2002). In particular, DIC method is very useful in the Bayesian model selection problems when the posteriors are generated from Markov chain Monte Carlo (MCMC) simulation. In the case of MCMC simulation, both expected deviance of fit ($\overline{D}$) and effective number of parameters (p_D) can be readily calculated from samples of simulated posterior parameters and the original data set (Dempster, 1997). For example, for the basic SV model:

$$\begin{aligned} h_t &= \mu + \phi(h_{t-1} - \mu) + v_t, t = 1, ..., n, && (3.4) \\ y_t &= \exp\left(\frac{h_t}{2}\right) u_t, t = 1, ..., n. && (3.5) \end{aligned}$$

The MCMC simulation will provide us with a list of parameter and latent variable h_t values from each iteration. So at the end of the simulation, we have $h_1^{(i)}, h_2^{(i)}, ..., h_n^{(i)}, i = 1, ..., I$, where i means the i^{th} sample and the I is the total number of samples. So in this particular case, the goodness-of-fit measurement, $\overline{D}$, is calculated by the following formula:

$$\begin{aligned} \overline{D} &= -2E_{\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{h}}[\log f(\boldsymbol{y}|\boldsymbol{\theta}, h_t)], \text{ which can be estimated by} && (3.6) \\ \widehat{\overline{D}} &= -2\sum_{t=1}^{n} \frac{1}{I} \sum_{i=1}^{I} \log f(y_t^{\text{obs}}|h_t^{(i)}), && (3.7) \end{aligned}$$

where $\log f(y_t^{\text{obs}}|h_t^{(i)})$ is the log-scaled probability density of the t^{th} observation with mean zero and standard deviation $\exp(h_t^{(i)})$. On the other hand, the measurement of model complexity, p_D, is given by $\overline{D} - D(\overline{\boldsymbol{\theta}})$. The calculation of $\overline{D}$ has been shown in the above formula, and the following equation can be used to compute the $D(\overline{\boldsymbol{\theta}})$:

$$\widehat{D(\overline{\boldsymbol{\theta}})} = -2\sum_{t=1}^{n} \log f(y_t^{\text{obs}}|\overline{h_t}), t = 1, ..., n, \tag{3.8}$$

where $\overline{h_t}$ is the posterior mean of h_t, which can be estimated by the average of h_t in MCMC samples.

In general, the major advantage of DIC method is that the method is very easy to implement and is proven to work reasonably well for the problems with identifiable parameterization. The DIC method typically does not require many model specific adjustments, and the calculation is usually not time consuming. Therefore, DIC method is a cost-efficient solution to Bayesian model selection problems. However, the method assumes a posterior multivariate normal distribution, which may not be guaranteed in some cases.

3.3 The Widely Applicable Information Criterion

The widely applicable information criterion (WAIC, also called non-integrated WAIC, or nWAIC) is a recently developed model-selection method aiming to correct for optimistic bias. Like other popular model selection methods, the formula of WAIC approach comprises two terms, goodness-of-fit measurement and penalty for model-complexity, which is given below(Watanabe, 2010):

$$\text{WAIC} = -2\sum_{t=1}^{n} \log E_{\text{post}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})] + 2p_{WAIC}. \tag{3.9}$$

The first term in the above formula is -2 times the summation of the log-scaled expectation of predictive probability density. The higher the value, the worse the fit. The second term p_{WAIC} (effective number of parameters) measures model complexity, which could be calculated by either of the following two formulas:

$$\begin{aligned} p_{WAIC1} &= 2\sum_{t=1}^{n}\{\log E_{post,\ \boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[f(y_t^{\text{obs}}|\boldsymbol{\theta})] - E_{post,\ \boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[\log f(y_t^{\text{obs}}|\boldsymbol{\theta})]\}, \text{ or} \quad (3.10) \\ p_{WAIC2} &= \sum_{t=1}^{n}\text{Var}_{\text{post}}[\log f(y_t^{\text{obs}}|\boldsymbol{\theta})]. \quad (3.11) \end{aligned}$$

A higher p_{WAIC} value corresponds to a higher level of model complexity. In our study, the effective number of parameters is calculated according to the latter formula (p_{WAIC2}).

For all SV models in this study, goodness-of-fit measurement is calculated by -2 times the summation of the log-scaled expected predictive probabilities of the observations:

$$-2\sum_{t=1}^{n}\log E_{\boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)], \quad (3.12)$$

which is estimated by:

$$-2\sum_{t=1}^{n}\log\left[\frac{1}{I}\sum_{i=1}^{I} f(y_t^{\text{obs}}|\boldsymbol{\theta}^{(i)}, h_t^{(i)})\right]. \quad (3.13)$$

In the formula above, the logarithm of expected probabilities are calculated as the natural logarithm of the average predictive probabilities of the observations over different parameter samples.

The effective number of parameters, p_{WAIC}, is calculated by summing up the variance of $\log f(y_t^{\text{obs}}|\boldsymbol{\theta})$. For each y_t^{obs}, the variance is estimated by the sample variance of $\left\{\log f(y_t^{\text{obs}}|\boldsymbol{\theta}^{(i)}, h_t^{(i)})\right\}$, where $i = 1, ..., I$ are numbers of samples.

In general, WAIC method is similar to DIC method. Like DIC method, WAIC method contains a goodness-of-fit element and a penalty-for-model-complexity element. Also, WAIC is easy to compute and requires few adjustments when implemented. However, the theoretical basis of this method to models with latent variables is unknown, and further studies are necessary to examine its correctness.

3.4 The Importance Sampling Information Criterion

The importance sampling information criterion (IS-IC, also called non-integrated IS-IC, or nIS-IC) is another model selection technique in an effort to approximate cross-validation results (Gelfand et al., 1992; Gelfand, 1996). Although the importance sampling method has been proposed for a long time, its application to SV models hasn't been studied yet. Suppose X and Y follow two distributions with probability density function of $f(X)$ and $g(Y)$, and we can sample directly from the distribution of X and we know that $h(Y) \propto g(Y)$, then we can estimate the expected value of any function with respect to the distribution of Y. If the function of interest is $j(Y)$, and a sample of X is given by $\{X_1, X_2, ..., X_I\}$, then according to the importance sampling rule, $E[j(Y)]$ can be estimated as:

$$\widehat{E[j(Y)]} = \frac{\frac{1}{I}\sum_{i=1}^{n} j(X_i) * W_i}{\frac{1}{I}\sum_{i=1}^{n} W_i}, \tag{3.14}$$

$$W = \frac{h(X_i)}{f(X_i)}, \tag{3.15}$$

where W is called the importance weight. If we apply the above formulas to our specific problem of SV models, and our goal is to approximate the LOOCV result, then the above formulas would be equivalent to:

$$\begin{aligned} \text{IS-IC} &= -2\sum_{t=1}^{n} \log \frac{E_{post,\ \boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)W_t(\boldsymbol{\theta}, h_t)]}{E_{post,\ \boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[W_t(\boldsymbol{\theta}, h_t)]} \\ &= -2\sum_{t=1}^{n} \log \frac{1}{E_{post,\ \boldsymbol{\theta}|\boldsymbol{y}^{\text{obs}}}[1/f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)]}, \end{aligned} \tag{3.16}$$

where $W_t(\boldsymbol{\theta}, h_t) = \frac{1}{f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)}$ is the weight measurement. The higher the pointwise probability is, the smaller the weight. In our studies, the IS-IC can be estimated by:

$$\text{IS-IC} = -2\sum_{t=1}^{n} \log \frac{1}{\frac{1}{I}\sum_{i=1}^{I} 1/f(y_t^{\text{obs}}|\boldsymbol{\theta}^{(i)}, h_t^{(i)})} \tag{3.17}$$

In the SV models, the i^{th} sample of model parameters corresponding to the t^{th} observation is given by $\boldsymbol{\theta}^{(i)}$ and $h_t^{(i)}$. Given the $\boldsymbol{\theta}^{(i)}$ and $h_t^{(i)}$, the $f(y_t^{\text{obs}}|\boldsymbol{\theta}^{(i)}, h_t^{(i)})$ is the probability density of

y_t^{obs} according to a normal distribution with mean zero and standard deviation of $\exp\left(\frac{h_t^{(i)}}{2}\right)$.

3.5 The Integrated WAIC and IS Criteria

Although both the non-integrated WAIC and IS-IC can be used to validate all SV models covered in our studies, it should be noted that simulated h_t are heavily affected by their corresponding observations. In an effort to fit the observations, the MCMC samples of h_t are largely confined to the regions that fit y_t^{obs} well (Li et al., 2015). As a result, the marginal distribution of h_t may be biased to the regions that fit y_t^{obs} well. To reduce this bias, the distribution of h_t should cover larger regions that are not affected by y_t^{obs}.

In actual LOOCV, the expectation of the likelihood of the test observation y_t^{obs} is calculated using the fitted parameters unaffected by the test observation itself. Under this condition, model evaluation is generally considered to be unbiased estimate of the out-of-sample predictive evaluation. Therefore, the likelihood in the formulas of WAIC and IS-IC could be replaced by $f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$, where $\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ are fitted model parameters.

In our studies, each SV model has a series of latent variables h_t as a measurement of log-volatilities for the corresponding observations of y_t. Since the latent variables h_t are the log-volatilities of the y_t, each h_t is highly correlated to its corresponding observation y_t. Due to this correlation, the likelihood estimates of the observations are considered to be biased. As a result, by integrating with respect to the latent variable h_t when calculating the likelihood of the corresponding observation y_t, this optimistic bias can be largely reduced.

Here, we denote $f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ as the likelihood of y_t^{obs} based on all of the fitted parameters except h_t. To calculate this new likelihood, the original function $f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t)$ needs to be weighted by the distribution of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$. This weighted likelihood function can be calculated by the following integral:

$$f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) = \int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) dh_t, \tag{3.18}$$

where $\boldsymbol{h}_{-t}$ means $h_1, h_2, ..., h_{t-1}, h_{t+1}, ..., h_n$ (n is the total number of the observations), and $\boldsymbol{\theta}$ denotes all the other model parameters (for example, μ, ϕ, τ). Once we obtain this integrated likelihood, we can use it to calculate a new version of WAIC:

$$
\begin{aligned}
p_{iWAIC} &= \sum_{t=1}^{n} \mathrm{Var}_{\mathrm{post}}[\log f(y_t^{\mathrm{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})], && (3.19)\\
\mathrm{iWAIC} &= -2\sum_{t=1}^{n} \log E[f(y_t^{\mathrm{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})] + 2p_{WAIC}, && (3.20)
\end{aligned}
$$

and IS-IC:

$$
\text{iIS-IC} = -2\sum_{t=1}^{n} \log \frac{1}{E_{\mathrm{post}}[1/f(y_t^{\mathrm{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})]}. \tag{3.21}
$$

This new version of WAIC and IS-IC is called integrated WAIC (iWAIC) and integrated IS-IC (iIS-IC), respectively. Due to the avoidance of direct use of fitted h_t during the calculation of the integrated criteria, the optimistic bias is expected to be smaller. As a result, iWAIC and iIS methods could potentially improve the performance of the model-selection criteria over the conventional WAIC and IS-IC approaches.

In our study, we assume the distribution of current log-volatility, h_t, can only be obtained according to the values of the other parameters. The parameters determining the distribution of h_t include its neighboring log-volatility measurements and $\boldsymbol{\theta}$. This h_t distribution can be calculated as follows (see the appendix for the detailed derivations).

- **Conditional distribution of h_t given $\boldsymbol{h}_{-t}$ in AR(1) process**

  When log-volatilities follow an AR(1) process (model 1, 2, 6, 7, and 8), then for $t = 1, ..., n$, the posterior of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ follows the following normal distribution:

$$
\begin{aligned}
h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} \sim\ & N\left(\frac{\phi}{\phi^2+1}h_{t+1} + \frac{\phi}{\phi^2+1}h_{t-1} + \frac{\mu}{\phi(\phi^2+1)} - \frac{\phi\mu}{\phi^2+1} - \frac{\mu}{\phi}\right. && (3.22)\\
& \left. +\mu, \frac{\tau^2\phi^2}{\phi^2+1}\right).
\end{aligned}
$$

- **Conditional distribution of h_t given $\boldsymbol{h}_{-t}$ in AR(2) process**

  When the log-volatilities follow an AR(2) process (model 3), the distribution of $h_t|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$

is normal as well:

$$\begin{aligned} h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} \sim & N\left(\frac{\phi+\psi-\phi\psi}{1+\phi^2}h_{t-1} + \frac{\phi}{1+\phi^2}h_{t+1} + \frac{\phi\psi\mu - 2\phi\mu - \psi\mu}{1+\phi^2}\right. \\ & \left. +\mu, \frac{\tau^2}{1+\phi^2}\right), t = 1, ..., n. \end{aligned} \tag{3.23}$$

- **Conditional distribution of h_t given $\boldsymbol{h}_{-t}$ in two independent AR(1) process**

  When there are two independent log-volatility processes going on simultaneously (model 4), the distributions of $h_t^{(1)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ and $h_t^{(2)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ are given by:

$$h_t^{(1)}|\boldsymbol{h}_{-t}, \phi, \tau \sim N\left(\frac{\phi(h_{t+1}^{(1)} + h_{t-1}^{(1)})}{1+\phi^2}, \frac{\tau^2\phi^2}{1+\phi^2}\right), t = 1, ..., n. \tag{3.24}$$

$$h_t^{(2)}|\boldsymbol{h}_{-t}, \phi_2, \tau_2 \sim N\left(\frac{\phi_2(h_{t+1}^{(2)} + h_{t-1}^{(2)})}{1+\phi_2^2}, \frac{\tau_2^2\phi_2^2}{1+\phi_2^2}\right), t = 1, ..., n. \tag{3.25}$$

- For **model 5**, however, the distribution of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ is not well-defined. Like all the other models, the probability density function of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ is proportional to the product of the probability density functions of $h_t|\boldsymbol{\theta}, h_{t-1}$ and $h_{t+1}|\boldsymbol{\theta}, h_t$:

$$\begin{aligned} g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) = & \exp\left\{\frac{-\left[h_t - \mu - \phi h_{t-1} + \phi\mu - \rho\phi\boldsymbol{y}_{t-1}\exp\left(\frac{-h_{t-1}}{2}\right)\right]^2}{2\tau^2(1-\rho^2)}\right. \\ & \left. - \frac{-\left[h_{t+1} - \mu - \phi h_t + \phi\mu - \rho\tau y_t \exp\left(\frac{-h_t}{2}\right)\right]^2}{2\tau^2(1-\rho^2)}\right\}, \end{aligned} \tag{3.26}$$

$$f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) \propto g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}), t = 1, ..., n. \tag{3.27}$$

Since $f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) = \int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) dh_t$, we have:

$$f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) = \frac{\int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) dh_t}{\int_{-\infty}^{\infty} g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) dh_t}, t = 1, ..., n, \tag{3.28}$$

where $g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ is proportional to $f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$, and the integral in the denominator,

$\int_{-\infty}^{\infty} g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})dh_t$ is the normalizing constant.

However, in practice, the direct calculation of the integral may give rise to a big error, and a safer way to diminish this error is the application of the following logarithms:

$$\begin{aligned}
\log f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) &= \log \left[\frac{\int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})dh_t}{\int_{-\infty}^{\infty} g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})dh_t} \right] && (3.29) \\
&= \log[\int_{-\infty}^{\infty} f(y_t^{\text{obs}}|\boldsymbol{\theta}, h_t) g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})dh_t] && (3.30) \\
&\quad - \log[\int_{-\infty}^{\infty} g(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})dh_t], t = 1, ..., n.
\end{aligned}$$

Therefore, in order to calculate the integrated likelihood for this model, we need to respectively evaluate the two integrals on the numerator and the denominator of the equation.

As we can see from the upper and lower limits, the integrals are both improper integrals. Therefore, we need to transform them to proper integrals in the first step. The transformation could be completed by substituting h_t with a function of $\boldsymbol{k}$: $\frac{h_{t-1}+h_{t+1}}{2} + \log(\frac{\boldsymbol{k}}{1-\boldsymbol{k}})$. This substitution changes the upper and lower limit of the integral from $-\infty$ and ∞ to 0 and 1. The result of the integrals can both be approximated by midpoint rule. To calculate an integral of a function f using midpoint rule, we equally divide n subintervals between the lower and upper limits, and calculate the function value at the midpoint of each subinterval. An approximated result can thus be obtained by multiplying the length of one subinterval by the summation of the function values at midpoints. Here, we used $n = 100$ for the total number of the subintervals, and the integrals on the numerator and denominator were calculated separately. The final result of the integrated log-likelihood can be subsequently obtained once we are able to evaluate the numerator and the denominator.

Chapter 4
Empirical Results

4.1 Simulation Studies

In our first study, the performances of the model selection criteria were tested by using a group of simulated data sets. First, we generated a data set from model 6, which means that the real model for the data is model 6. This data-generating process was repeated 100 times to generate 100 data sets. Second, each of the simulated data sets was individually fitted into all the candidate SV models listed in Chapter 2. Finally, the model-selection criteria, including DIC, nWAIC, nIS, iWAIC, and iIS, were used to choose the best model for simulated data sets.

In the first step, data sets were simulated by setting the parameters in model 6 to some specific values. In our particular case, the parameters used for the data-generations are $\mu = -10$, $\phi = 0.96$, $\tau = 0.345$, $\beta = 0.1$, $\kappa = 0.08$, and $\delta = 0.03$. Each simulated data set is a time series with 2000 observations.

Once the data sets were generated, we subsequently fitted the candidate SV models (see Chapter 2 for a list of the models) to the data. To fit the models, Markov chain Monte Carlo (MCMC) method was used to sample from the posteriors of the parameters in each model. A number of MCMC algorithms have been proposed to sample model parameter(s), such as the Metropolis-Hastings algorithm and Gibbs sampling. Based on these MCMC algorithms, many sampling software packages are developed, including WinBUGS, OpenBUGS and JAGS (Lunn et al., 2000; Spiegelhalter et al., 2007; Plummer, 2003). However, since these packages are mainly based on the Metropolis-Hastings algorithm, they may suffer from slow convergence problem due to random walk method used in the algorithm to propose a new state. To overcome this problem, stan package was developed (Carpenter et al., 2015;

Gelman et al., 2015). In stan, convergence could be much faster through the application of Hamiltonian Monte Carlo and no-U-turn sampling (Carpenter et al., 2015). Therefore, we decided to use stan sampler for our particular studies of SV models.

Before we can use the stan sampler to sample from the posterior distributions of model parameters, we need to assign priors to the parameters first. For all SV models in this study, the prior distribution of μ is normal with mean -10 and standard deviation of 5. In addition, the prior of $\tau^2 \sim \text{Inverse-Gamma}(2.5, 0.025)$ (Kim et al., 1998), and the prior of ϕ is uniformly distributed between zero and one for all the candidate models. For model 2, the prior of the parameter $\alpha \sim N(0, 10)$, and all the other parameter priors are the same as the basic SV model. The prior distribution of ψ in model 3 is identical with the prior distribution of ϕ in the basic SV model (uniformly distributed between 0 and 1). In model 4, the prior of the parameter ϕ_2 has the same distribution as ϕ in the basic SV model. For model 5, the prior of ρ is uniformly distributed between -1 and 1 with mean zero, which gives a non-informative prior distribution to the correlation parameter ρ. The β parameter in model 6 measures how much the current observation would affect the previous observation, and the parameter is generally considered as small. As a result, we imposed an informative prior of $\beta \sim N(0, 0.2)$ on this parameter. Also in model 6, the κ parameter that measures the probability of the occurrence of a jump (an additional upward or downward movement of y_t that may or may not occur) in an observation, is assigned with a $\text{Beta}(2, 100)$ prior (Chib et al., 2002). On the other hand, the prior of the jump-size parameter s_t follows a distribution of $\ln(1+s_t) \sim N(-\delta^2/2, \delta^2)$, and we assume that the prior distribution of $\log(\delta)$ is given by $\log(\delta) \sim N(-3.07, 0.149)$ (Chib et al., 2002). In model 8, the parameter ν has a uniform distribution on [2, 128] as its prior (Chib et al., 2002).

Once the priors of the model parameters were set, stan sampler read in the simulated observations (from model 6) and subsequently fitted the candidate models. In order to ensure the convergence of Markov chains, the number of sampling iterations were set to be $20,000$ for each individual Markov chain. Since the chains may take a while to converge, the first $10,000$ samples were dropped. In order to reduce autocorrelations between the neighboring samples, the final sample contains only every tenth sample of the remaining $10,000$ samples. Besides, to ensure the convergence of the Markov chains, two independent chains were run

Table 4.1: Average Parameter $\hat{R}$ for Simulated Data. The values in the table are the average $\hat{R}$ values of the model parameters over all the 100 data sets, with their corresponding standard deviations in the next line in the parentheses.

	Model							
Parameter	1	2	3	4	5	6	7	8
μ	1.0001	1.0000	1.0004	1.0918	1.0000	1.0003	1.0003	1.0006
	(0.0008)	(0.0007)	(0.0018)	(0.2289)	(0.007)	(0.0010)	(0.0011)	(0.0014)
ϕ	1.0024	1.0026	1.0174	53.8731	1.0024	1.0028	1.0026	1.0036
	(0.0024)	(0.0025)	(0.0156)	(70.0696)	(0.0033)	(0.0032)	(0.0037)	(0.0034)
ϕ_2	-	-	-	59.9186	-	-	-	-
	-	-	-	(90.8096)	-	-	-	-
τ	1.0056	1.0060	1.0181	2.8202	1.0062	1.0056	1.0048	1.0086
	(0.0044)	(0.0052)	(0.0161)	(1.8771)	(0.0060)	(0.0052)	(0.0054)	(0.0077)
τ_2	-	-	-	2.9484	-	-	-	-
	-	-	-	(1.9141)	-	-	-	-
α	-	1.0000	-	-	-	-	-	-
	-	(0.0010)	-	-	-	-	-	-
ψ	-	-	1.0168	-	-	-	-	-
	-	-	(0.0152)	-	-	-	-	-
ρ	-	-	-	-	0.9999	-	-	-
	-	-	-	-	(0.0007)	-	-	-
β	-	-	-	-	-	1.0002	-	-
	-	-	-	-	-	(0.0010)	-	-
κ	-	-	-	-	-	1.0034	1.0046	-
	-	-	-	-	-	(0.0056)	(0.0075)	-
δ	-	-	-	-	-	1.3777	1.3637	-
	-	-	-	-	-	(0.2737)	(0.2706)	-
ν	-	-	-	-	-	-	-	1.0465
	-	-	-	-	-	-	-	(0.0410)

simultaneously for each simulated data set. Comparison between the two chains on the same set of data confirmed that the Markov chains are converged before the first $10,000$ samples of the MCMC sampling (see Figure 4.1 for examples of the trace plots for model 6 parameters, and Table 4.1 for $\hat{R}$ values). The $\hat{R}$ is a relative measurement of cross-chain variances to within-chain variances, with values near 1.0 indicating good convergence (Gelman et al., 2011). In our studies, we ran two individual Markov chains for each posterior distribution (based on a data set for a given model), and if the Markov chains do converge, the two chains for the same set of data should exhibit similar patterns after the convergence point. An $\hat{R}$ value greater than 1 suggests imperfect convergence, and the bigger the $\hat{R}$ value, the worse the convergence. The parameter $\hat{R}$ values for the fitted models (using the simulated data) are mostly very close to one, indicating the Markov chains do converge for these models.

An exception, though, is the ϕ ($\hat{R} = 53.8731$), τ ($\hat{R} = 2.8202$), ϕ_2 ($\hat{R} = 59.9186$), and τ_2 ($\hat{R} = 2.9484$) parameters in model 4. These large values of $\hat{R}$ show that the Markov chains do not converge well in this model. However, the issue is not a big concern in this particular case. In model 4, we have two independent AR(1) processes that have the same formula format. As a result, the model contains two modes. If one mode contains $h_t^{(1)}$, ϕ, τ, $h_t^{(2)}$, ϕ_2, τ_2 and all the other parameters, then the other mode is formed by keeping all the other parameters unchanged while exchanging the values of $h_t^{(1)}$, ϕ, and τ altogether with $h_t^{(2)}$, ϕ_2, and τ_2. Therefore, the high values of $\hat{R}$ for model 4 are caused by the two chains converging to the two different modes (see Figure 4.2 for an example). Since the two modes are relatively far apart from each other, it is difficult for any existing sampler to explore the parameter space in this particular case. Since converging to different modes would keep the distribution of $h_t^{(1)} + h_t^{(2)}$ unchanged, and the distribution of $\hat{y}_t$ only depends on the summation of $h_t^{(1)}$ and $h_t^{(2)}$, the overall model is unaffected with respect to the predictions of y_t.

The values of the fitted parameters and their standard deviations are listed in Table 4.2. The results from the table show that the expected values of model parameters generally fits the profile of the data-generating parameters, which indicates a good fit.

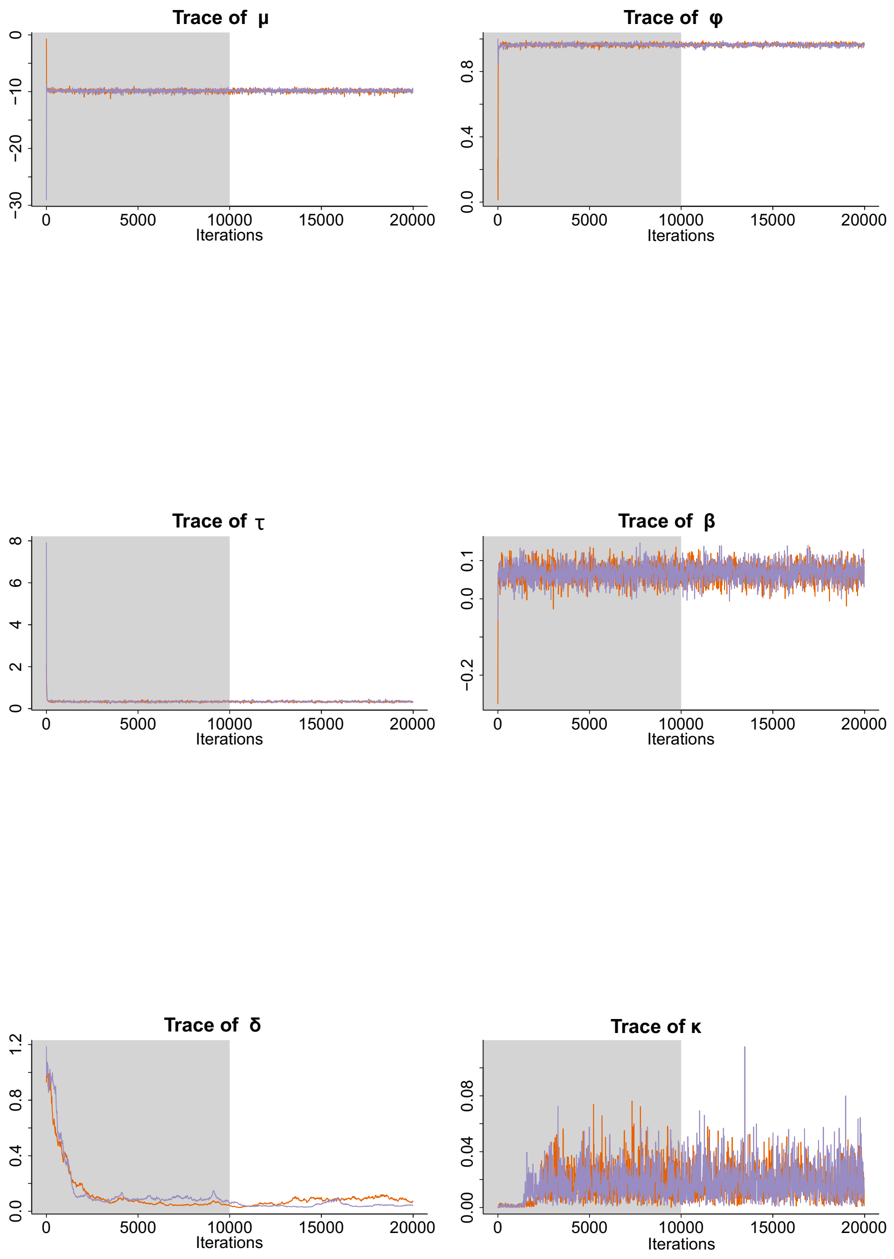
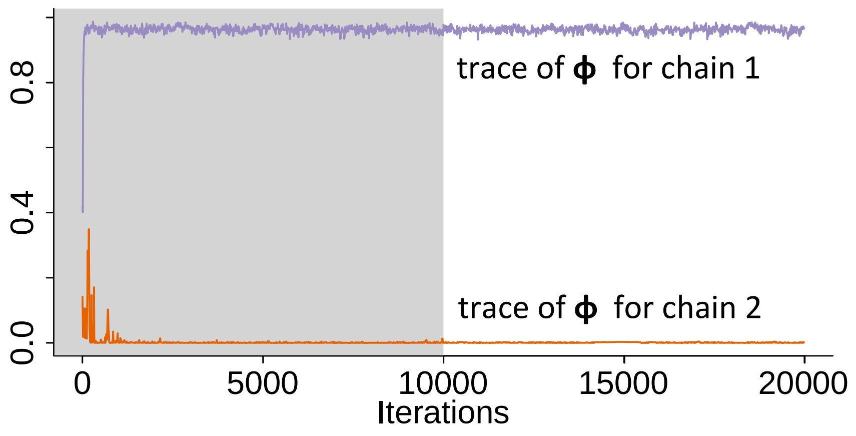


Figure 4.1: Example of trace plots of μ, ϕ, τ, β, δ, and κ in model 6. The two chains in each trace plot are from two individually simulated Markov chains based on model 6 and the same set of data. The cross-chain variances is relatively small comparing to the within-chain variances after the burn-in period, indicating good convergence of the Markov chains.



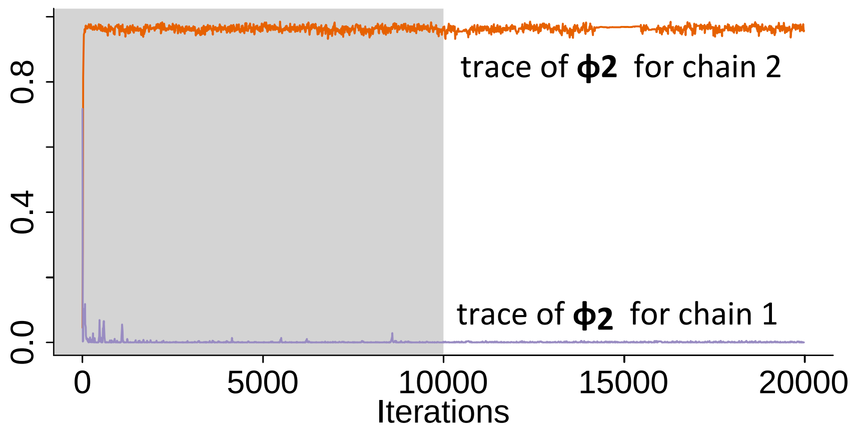


Figure 4.2: Example of trace plots of ϕ and ϕ_2 in model 4 when two Markov chains converge to different modes. The ϕ and ϕ_2 in the trace plots are from two individually simulated Markov chains based on model 4 and the same set of data. The cross-chain variances are huge comparing to the within-chain variances due to the fact that the two chain converge to two different modes.

Table 4.2: Average Parameter Estimates for Simulated Data. The values in the table are the expected values of the prior and the average values of the posterior model parameters, with their corresponding standard deviations in the next line in the parentheses.

		Model							
Parameter	Distribution	1	2	3	4	5	6	7	8
μ	Prior	-10	-10	-10	-10	-10	-10	-10	-10
		(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)
[-10]	Posterior	-9.964	-9.964	-9.971	-9.97	-9.961	-9.971	-9.964	-9.991
		(0.2020)	(0.2054)	(0.2051)	(0.1918)	(0.2027)	(0.2098)	(0.2085)	(0.2055)
ϕ	Prior	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)
[.96]	Posterior	0.9577	0.9585	0.7504	0.5304	0.958	0.9588	0.9585	0.9592
		(0.0093)	(0.0091)	(0.1423)	(0.2629)	(0.0092)	(0.0093)	(0.0093)	(0.0091)
τ	Prior	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
		(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)
[.345]	Posterior	0.3381	0.3367	0.3992	0.2364	0.3375	0.3377	0.3391	0.3301
		(0.0292)	(0.0290)	(0.0520)	(0.0766)	(0.0294)	(0.0295)	(0.0295)	(0.0296)
α	Prior	-	0	-	-	-	-	-	-
		-	(3.16)	-	-	-	-	-	-
	Posterior	-	$-1.6e^{-5}$	-	-	-	-	-	-
		-	(0.0001)	-	-	-	-	-	-
ψ	Prior	-	-	0.5	-	-	-	-	-
		-	-	(0.2887)	-	-	-	-	-
	Posterior	-	-	0.2004	-	-	-	-	-
		-	-	(0.1377)	-	-	-	-	-
ϕ_2	Prior	-	-	-	0.5	-	-	-	-
		-	-	-	(0.2887)	-	-	-	-
	Posterior	-	-	-	0.4591	-	-	-	-
		-	-	-	(0.0790)	-	-	-	-
τ_2	Prior	-	-	-	0.12	-	-	-	-
		-	-	-	(0.05)	-	-	-	-
	Posterior	-	-	-	0.22		-	-	-
		-	-	-	(0.2693)	-	-	-	-
ρ	Prior	-	-	-	-	0	-	-	-
		-	-	-	-	(0.58)	-	-	-

Table 4.2: Average Parameter Estimates for Simulated Data (continued)

		Model							
Parameter	Distribution	1	2	3	4	5	6	7	8
	Posterior	-	-	-	-	0.0602	-	-	-
		-	-	-	-	(0.0404)	-	-	-
β	Prior	-	-	-	-	-	0	-	-
		-	-	-	-	-	(0.45)	-	-
[0.1]	Posterior	-	-	-	-	-	0.0987	-	-
		-	-	-	-	-	(0.0238)	-	-
κ	Prior	-	-	-	-	-	0.02	0.02	-
		-	-	-	-	-	(0.01)	(0.01)	-
[0.08]	Posterior	-	-	-	-	-	0.0188	0.0187	-
		-	-	-	-	-	(0.0130)	(0.0130)	-
δ	Prior	-	-	-	-	-	0.05	0.05	-
		-	-	-	-	-	(0.02)	(0.02)	-
[0.03]	Posterior	-	-	-	-	-	0.0484	0.0496	-
		-	-	-	-	-	(0.0144)	(0.0152)	-
ν	Prior	-	-	-	-	-	-	-	65
		-	-	-	-	-	-	-	(36.4)
	Posterior	-	-	-	-	-	-	-	74.943
		-	-	-	-	-	-	-	(30.347)

When the sampling of the model parameters was completed by the stan sampler, the DIC, WAIC, IS, iWAIC, and iIS criteria were used to make the model selection. In order to calculate the integrated likelihood for the iIS and iWAIC, we sampled one hundred h_t for each time point t in each iteration. This random sampling process was completed according to the calculated distribution of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ (see Chapter 3 for details). When the samples from $f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ was obtained, the corresponding $\log f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ could be calculated. To calculate this integrated likelihood, we plugged in the the samples from $f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ into the probability function of y_t^{obs}, one at a time, in order to compute a total of one hundred log-scaled probability of y_t^{obs} for each time point in each iteration. Finally, the average of the one hundred log-likelihood of y_t^{obs} would provide a good estimate of the desired integrated log-likelihood $\log f(y_t^{\text{obs}}|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$. For model 5, however, the samples from $f(h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t})$ cannot be readily acquired from a well-defined distribution, and the approximated value of the integrated log-likelihood is calculated by the method of numerical quadrature (see Chapter 3 for details).

Since the model used to generate the data is model 6, a good model-selection criterion should be able to choose this model in most cases. For each data set under each criterion, the best model is the model with the lowest IC value. The average IC value results provided by Table 4.3b show that model 6 has the lowest average DIC and nWAIC values, suggesting the two criteria are able to correctly choose the real model in general. On the other hand, the average iWAIC value for model 6 (real model) is much higher than model 4, indicating the iWAIC method is not very reliable in selecting the correct model. Besides, the average nIS value for model 6 is also a little higher than model 4, although model 6 is more frequently selected than model 4 under the nIS criterion. The average iIS value of model 6, however, is much lower than that of model 4, indicating the integration step does improve the performance of the importance sampling information criterion. The model selection frequency results are listed in Table 4.3c. In this table, each information criterion decides which model to choose according to the IC values. According to Table 4.3c, the DIC criterion is able to choose the correct model (model 6) most of the time. In addition, although model 4 is not the true model, the DIC criterion seems to have a strong favor over it (28 out of 100). Finally, the DIC method manages to tell that the remaining candidate models (model 1, 2, 3, 5, 7, and 8) are not suitable to the data sets. These patterns of model-selection results shown in the DIC are very similar to the result patterns shown in the nIS. In nIS results, the true model is also selected most of the time (57 out of 100), and model 4 is frequently mis-selected as well (32 out of 100). The nWAIC method, on the other hand, has a moderate preference on model 8 (11 out of 100) besides model 6 (55 out of 100)and model 4 (30 out of 100). The iWAIC method is able to decrease the mis-selection of model 4 (24 out of 100) comparing to the nWAIC, but a two-fold increase in the selection of another incorrect model, model 8 (35 out of 100), makes the iWAIC the least likely method to choose the correct model. The iIS method, however, could successfully decrease the selection frequency of model 4 (24 out of 100) while not inflicting other significant mis-selections.

The CPU time table shows how much system time is used to calculate the information

Table 4.3: The Model Selection Results for Simulated Data

(a) The DIC, nWAIC, iWAIC, nWAIC, nIS, and iIS values for a randomly chosen set of simulation data.

Model	DIC	nWAIC	iWAIC	nIS	iIS
1	-13733.7	-13740.8	-13718.2	-13704.3	-13696.4
2	-13730.7	-13738.5	-13714.2	-13699.0	-13693.9
3	-13735.5	-13746.2	-13709.8	-13706.7	-13694.7
4	-13733.2	-13742.8	-13714.8	-13709.6	-13696.8
5	-13735.0	-13742.2	-13719.8	-13694.5	-13698.4
6	-13735.6	-13744.7	-13704.5	-13697.3	-13696.6
7	-13724.6	-13733.4	-13667.8	-13694.9	-13656.0
8	-13731.2	-13746.4	-13722.0	-13701.6	-13693.0

(b) The average DIC, nWAIC, iWAIC, nWAIC, nIS, and iIS values for the 100 sets of simulation data.

Model	DIC	nWAIC	iWAIC	nIS	iIS
1	-14040.8	-14048.7	-14020.7	-14007.2	-13997.3
2	-14040.3	-14047.9	-14020.0	-14006.7	-13997.1
3	-14041.7	-14053.5	-14011.4	-14008.4	-13993.5
4	-14049.6	-14060.4	-14023.8	-14022.2	-14005.5
5	-14040.3	-14047.9	-14022.0	-14007.1	-14000.5
6	-14053.1	-14061.1	-14024.9	-14020.5	-14017.6
7	-14037.2	-14046.6	-13994.2	-14006.0	-13983.8
8	-14039.7	-14056.1	-14027.0	-14009.3	-13997.7

(c) The model selection frequencies for all the 100 sets of simulated data.

	DIC	nWAIC	iWAIC	nIS	iIS
Model	Times Selected	Times Selected	Times Selected	Times Selected	Times Selected
1	1	0	0	3	2
2	1	1	2	1	0
3	3	3	0	2	0
4	28	30	24	32	21
5	0	0	3	1	1
6	**67**	**55**	**36**	**57**	**75**
7	0	0	0	0	0
8	0	11	35	4	1

criteria in each model. According to the results shown in Table 4.4, on average, the DIC method takes the least amount of time (from 10.4 seconds to 25.5 seconds), and nWAIC and nIS approaches both require slightly more time (about one second difference) than the DIC method. The iWAIC and iIS methods are the most time-consuming approaches. Due to the integration process, both iWAIC and iIS need 3 to 4.5 minutes of CPU time to compute. The CPU time also varies with models. In general, the calculation of information criteria tend to take less time for simple models (model 1, 2, 3, etc) than for complex models (model 6, 7, etc).

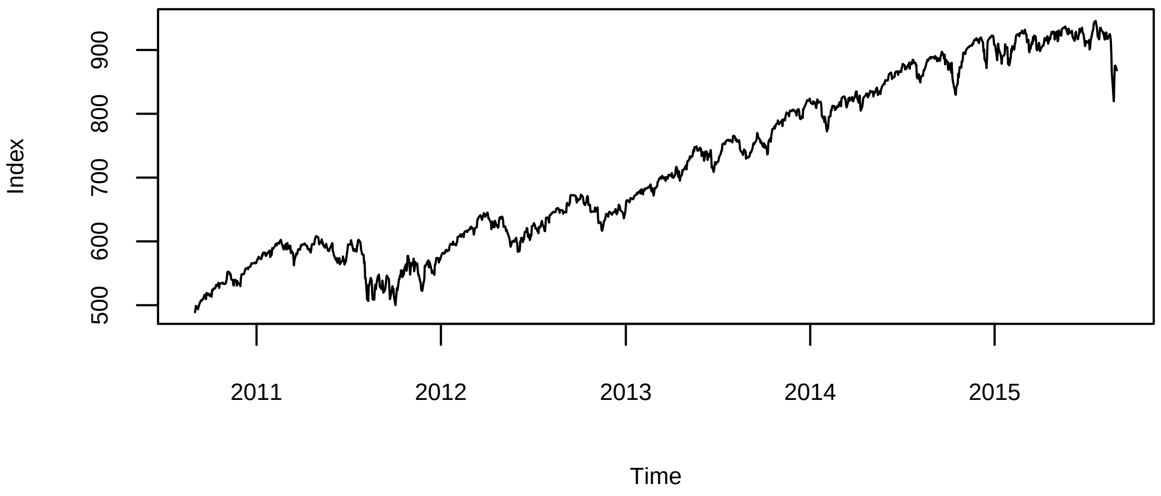
Table 4.4: The Average CPU Time Used to Compute the Criteria for the 100 Simulated Data Sets (in Seconds)

Model	DIC	nWAIC	iWAIC	nIS	iIS
1	10.403	11.240	181.519	11.470	181.672
2	10.457	11.292	194.370	11.555	194.412
3	10.452	11.290	182.253	11.525	182.401
4	15.571	16.407	262.114	16.707	262.200
5	10.212	11.048	277.157	11.260	277.367
6	25.448	26.282	273.858	26.525	273.995
7	25.506	26.328	274.639	26.562	274.785
8	15.591	16.427	213.800	16.678	213.964

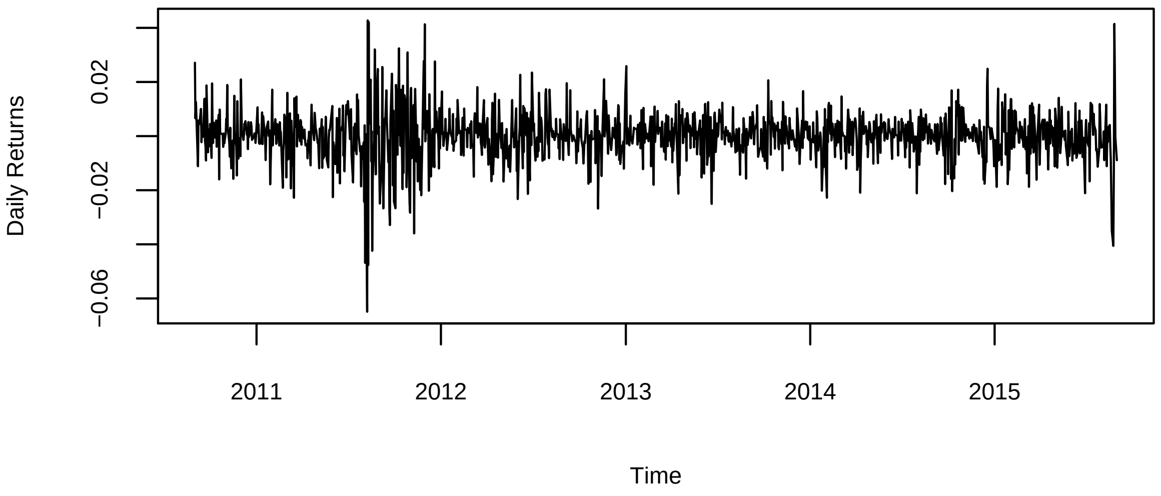
4.2 An Empirical Study on S&P 100 Data

Besides the simulation study, we also used a set of real-world stock market data (the S&P 100 stock index data from September 2010 to August 2015) to fit the SV models. The S&P 100 includes 100 leading U.S. stocks that comprise almost 45 percent of the market capitalization of the U.S. equity markets. This subset of stocks plays a major role in the capital market and is a good indicator for the overall strength of financial market. As a result, finding an appropriate way to model the S&P 100 index data is of great importance.

In this study, we used mean-corrected, continuously compounded daily returns of the S&P 100 index (exported from Yahoo Finance) from September 2010 to August 2015 (1,258 trading days). In general, as shown in Figure 4.3, the returns during this period went up, and is considered as a "recovery period" following the 2008 stock market falls. However, due to



(a) The Plot of S&P 100 Index from September 2010 to August 2015



(b) The Daily Return Plot of S&P 100 from September 2010 to August 2015

Figure 4.3: The plot of S&P 100 daily returns and index data from September 2010 to August 2015

frequent changes in economic conditions and monetary policies, the stock market volatilities are considerably different from time to time. Therefore, it makes sense to apply SV models to the stock market data.

Table 4.5: Parameter $\hat{R}$ for S&P 100 Index Data

	Model							
Parameter	1	2	3	4	5	6	7	8
μ	1.0018	1.0000	1.0003	1.1284	0.9993	1.0003	1.0008	1.0046
ϕ	1.0163	0.9998	1.0152	1.3141	1.0066	1.0022	1.0165	1.0169
τ	1.0199	0.9991	1.0176	1.0505	1.0141	1.0043	1.0226	1.0277
α	-	0.9991	-	-	-	-	-	-
ψ	-	-	1.0141	-	-	-	-	-
ϕ_2	-	-	-	1.0202	-	-	-	-
τ_2	-	-	-	1.0086	-	-	-	-
ρ	-	-	-	-	1.0004	-	-	-
β	-	-	-	-	-	0.9994	-	-
κ	-	-	-	-	-	1.0012	1.0002	-
δ	-	-	-	-	-	1.9959	1.3827	-
ν	-	-	-	-	-	-	-	1.0693

The model fitting process in the real data study is the same as our previous study on the simulated data. The rstan package was used to fit the model parameters with the stock market data. The total number of iterations for the Markov chain is 20,000, with a burn-in period of 10,000 iterations. That is, the first 10,000 samples were discarded. For the remaining 10,000 samples, we only retained every tenth sample in order to reduce autocorrelations. Two parallel Markov chains were run for the set of data in each model, and the $\hat{R}$ results (see Table 4.5 for details) show that the Markov chain converges after the burn-in period. The $\hat{R}$ values of the model parameters are generally close to one, indicating a good convergence of the Markov chains.

All the fitted parameters are listed in the model parameter table as shown in Table 4.6. From the results provided by the table, we can tell that some model parameters have very small absolute values and large variances, indicating the parameters are not significantly different from 0. If that's the case, the corresponding model may not be a good choice for the given data.

Table 4.6: Parameter Estimates for S&P 100 Index Data. The values in the table are the expected values of the prior and the posterior of the model parameters, with their corresponding standard deviations in the next line in the parentheses.

		Model							
Parameter	Distribution	1	2	3	4	5	6	7	8
μ	Prior	-10	-10	-10	-10	-10	-10	-10	-10
		(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)	(5.00)
	Posterior	-9.825	-9.836	-9.83	-9.791	-9.831	-9.732	-9.741	-9.871
		(0.1571)	(0.1632)	(0.1647)	(0.1343)	(0.1544)	(0.1708)	(0.1623)	(0.1626)
ϕ	Prior	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
		(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)	(0.2887)
	Posterior	0.9345	0.9363	0.5439	0.0074	0.9347	0.9401	0.9373	0.9383
		(0.0182)	(0.0179)	(0.2279)	(0.0066)	(0.0191)	(0.0185)	(0.0185)	(0.0186)
ϕ_2	Prior	-	-	-	0.5	-	-	-	-
		-	-	-	(0.2887)	-	-	-	-
	Posterior	-	-	-	0.9419	-	-	-	-
		-	-	-	(0.0139)	-	-	-	-
τ	Prior	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
		(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)	(0.05)
	Posterior	0.325	0.3283	0.4358	0.1313	0.3244	0.3139	0.3207	0.3121
		(0.0441)	(0.0457)	(0.0810)	(0.0551)	(0.0480)	(0.0455)	(0.0470)	(0.0475)
τ_2	Prior	-	-	-	0.12	-	-	-	-
		-	-	-	(0.05)	-	-	-	-
	Posterior	-	-	-	0.3067		-	-	-
		-	-	-	(0.0343)	-	-	-	-
α	Prior	-	0	-	-	-	-	-	-
		-	(3.16)	-	-	-	-	-	-
	Posterior	-	0.0004	-	-	-	-	-	-
		-	(0.0002)	-	-	-	-	-	-
ψ	Prior	-	-	0.5	-	-	-	-	-
		-	-	(0.2887)	-	-	-	-	-
	Posterior	-	-	0.3702	-	-	-	-	-
		-	-	(0.2173)	-	-	-	-	-
ρ	Prior	-	-	-	-	0	-	-	-
		-	-	-	-	(0.58)	-	-	-
	Posterior	-	-	-	-	0.0104	-	-	-

Table 4.6: Parameter Estimates for S&P 100 Index Data (continued)

		Model							
Parameter	Distribution	1	2	3	4	5	6	7	8
		-	-	-	-	(0.0104)	-	-	-
β	Prior	-	-	-	-	-	0	-	-
		-	-	-	-	-	(0.45)	-	-
	Posterior	-	-	-	-	-	-0.0371	-	-
		-	-	-	-	-	(0.0298)	-	-
κ	Prior	-	-	-	-	-	0.02	0.02	-
		-	-	-	-	-	(0.01)	(0.01)	-
	Posterior	-	-	-	-	-	0.0173	0.0182	-
		-	-	-	-	-	(0.0123)	(0.0129)	-
δ	Prior	-	-	-	-	-	0.05	0.05	-
		-	-	-	-	-	(0.02)	(0.02)	-
	Posterior	-	-	-	-	-	0.0553	0.0454	-
		-	-	-	-	-	(0.0242)	(0.0225)	-
ν	Prior	-	-	-	-	-	-	-	65
		-	-	-	-	-	-	-	(36.4)
	Posterior	-	-	-	-	-	-	-	58.354
		-	-	-	-	-	-	-	(33.324)

When we obtained the sample of model parameters from MCMC sampling process, we applied DIC, nWAIC, iWAIC, nWAIC, nIS, and iIS methods, respectively, to the models to make the selection (see the simulated study for details). The results listed in Table 4.7 show that all five model selection criteria except the iWAIC method select model 4 as the best model for the given stock market index data. Besides, the DIC, nWAIC, nIS, and iIS methods also provide very similar results on ranking the goodness of the models. The nWAIC method, however, selects model 8 as the best model. Also for the nWAIC method, the rest of the ranking result is very different from the other model-selection criteria as well.

The CPU time table for this empirical study exhibits a similar pattern as the simulated data study (see Table 4.8 for detailed results). The DIC method takes the least amount of time to compute (6 to 16 seconds), and the nWAIC and nIS methods needs a little bit more time than the DIC method (about half a second). The iWAIC and iIS criteria are the most

Table 4.7: The DIC, nWAIC, iWAIC, nWAIC, nIS, and iIS for S&P 100 Index Data

	DIC		nWAIC		iWAIC		nIS		iIS	
Model	Value	Ranking	Value	Ranking	Value	Ranking	Value	Ranking	Value	Ranking
4	-8674.47	1	-8673.28	1	-8649.779	3	-8655.53	1	-8639.73	1
2	-8672.7	2	-8669.39	4	-8651.002	2	-8646.8	2	-8636.47	2
1	-8665.9	5	-8661.55	5	-8644.613	4	-8631.92	5	-8629.95	3
8	-8668.75	4	-8670.61	3	-8652.337	1	-8632.02	4	-8629.88	4
3	-8670.91	3	-8671.88	2	-8638.254	6	-8640.62	3	-8629.76	5
5	-8663.63	6	-8659.54	6	-8643.669	5	-8631.54	6	-8629.19	6
7	-8661.08	7	-8656.85	7	-8629.491	7	-8630.55	8	-8623.23	7
6	-8659.46	8	-8654.25	8	-8623.167	8	-8630.58	7	-8616.41	8

expensive results, which requires 2 to 3 minutes CPU time on average.

Table 4.8: The CPU Time Used to Compute the Criteria for S&P 100 Index Data (in Seconds)

Model	DIC	nWAIC	iWAIC	nIS	iIS
1	6.356	6.878	114.088	7.015	114.222
2	6.446	6.970	118.802	7.112	118.931
3	6.497	7.023	115.261	7.162	115.394
4	9.558	10.081	161.369	10.237	161.498
5	6.431	6.957	179.323	7.095	179.470
6	15.776	16.302	180.280	16.439	180.412
7	15.722	16.236	176.942	16.375	177.083
8	9.480	10.003	129.989	10.134	130.119

Chapter 5
Conclusions and Discussions

In conclusion, according to the simulated data study, HMC method is successful in sampling from the posterior distributions of model parameters. Among the tested model-selection methods, the DIC approach works reasonably well. The good performance of the DIC method might be explained by the fact that in most of the fitted models, the parameters generally follow a multivariate normal distribution. Furthermore, the nIS is pretty consistent as well, which suggests the importance weight is an effective way to correct the optimistic bias. In addition, the iIS results show that the integration with respect to the current log-volatility, $\boldsymbol{h}_t$, is a good way to further address the bias issues. Therefore, the iIS method is able to make improvements over the nIS approach. However, the integrated method might not always be a good choice since it is computationally expensive. Finally, the performance of both nWAIC and iWAIC is the worst among all the tested methods, which makes their theoretical basis questionable. According to this study, we can tell that the two WAIC methods are probably not able to accurately quantify the model-complexity by their formulas.

Besides, the study on real stock market return data (S&P 100 index from September 2010 to August 2015) suggests that the best model is model 4 according to the model selection criteria, which indicates the data series follows an ARMA process. However, due to the fact that all the selection criteria have a strong preference on model 4 even when the real model is not model 4, selecting this model as the best model could be a mistake. As a result, the next-best model, model 2 (the nonzero expected return model), is also a good candidate for the real model.

In our studies, we used Markov chain Monte Carlo method to fit our stochastic volatility models, and evaluate the models using five different model-selection criteria (DIC, nWAIC, nIS, iWAIC, iIS) subsequently. To examine the reliability of the model-fitting algorithm and the consistency of the model-selection methods, a preliminary study was done on simulated data sets before using any real data. In the simulated study, a total number of 100 data sets were individually generated from model 6 with the following parameters: $\mu = -10$, $\phi = 0.96$, $\tau = 0.345$, $\beta = 0.1$, $\kappa = 0.08$, and $\delta = 0.03$. Through the data-generating process, we know both the real model and the true values for the model parameters. Thus we are able to evaluate the goodness of the model-fitting method, and the consistencies of the model-selection criteria as well.

Among the results, the parameter $\hat{R}$ values indicate that the Hamiltonian Monte Carlo (HMC) method does not give rise to any noticeable convergence issues in this study. Besides, the average values of the fitted parameters in model 6 are very close to the true values, indicating the HMC method works well in finding real parameter values for the model. It should be noted that the average values of both ϕ and ϕ_2 in model 4 are close to 0.5, and the standard deviations of the two parameters are much larger than the standard deviation of ϕ in the basic SV model (model 1). The average values and the large standard deviations of the two parameters (ϕ and ϕ_2) can be explained by the fact that the model 4 has two symmetric modes, and converging to different modes leads to bimodal distributions of both ϕ and ϕ_2. Since in both of the bimodal distributions, one mode is close to 1 and the other mode is close to 0, the average values of both ϕ and ϕ_2 are close to 0.5, and their corresponding standard deviations would be much larger than the unimodal ϕ in the basic SV model.

Although the primary goal of this study is not validating the HMC method, fitting the real model with correct parameter values is still a vital step. Without finding the correct values for the real model, the likelihood of the observations might be negatively affected, which can distort the model-selection results consequently. The model selection results, on the other hand, show that the DIC, nWAIC and nIS methods all perform reasonably well, while the iWAIC method's performance is worse than the nWAIC method. Among all the tested criteria, the iIS outperforms any other criterion, suggesting the integration may potentially improve the performance of the importance sampling information criterion.

The subsequent study on the S&P 100 index daily return data (from September 2010 to August 2015) shows that the Markov chains for all the candidate models are well-converged (all $\hat{R}$ values being very close to 1). This good convergence indicates the HMC method manages to fit all the candidate models with the stock market return data properly. It is noticed that unlike what we observed in the simulated data study, in the real data study, the $\hat{R}$ values for the ϕ and ϕ_2 parameters in model 4 are also very close to one, suggesting the two parallel Markov chains for the model happen to converge to the same mode in this particular case.

Once the the model fitting processes were completed, the DIC, nWAIC, iWAIC, nIS, and iIS criteria were used to select the best model for the data. Among all the information criteria (IC), the result provided by the iIS method seems to be very sensible and neat, since the top 4 models chosen by the iIS method (model 4, 2, 1, and 8) share many similarities. The first similarity is that none of the log-volatility processes in these models are AR(2). Secondly, none of the models contain the jump component in their observation equations. Thirdly, model 1 is nested with model 4, which explains why their IC values (rankings) are similar. In addition to the iIS result, all the other criteria except the iWAIC also picked model 4 as the best model, indicating model 4 is the best approximation for the real model. If that's the case, it would suggest that the volatility of stock market returns is the sum of two independent autoregressive processes during the given period of time. It should be noted, however, that according to the simulated data study, all the selection criteria have a strong favour on model 4, even when the real model is not model 4. As a result, model 4 could be mis-selection again in the real data study. Since the next-best model is model 2 (according to the DIC, iWAIC, nIS, and iIS results), it is possible that model 2 is the actual model for the stock market return data. Model 2 assumes a non-zero expected return (α), which is consistent with the stock market behavior between the selected period of time. According to the data, the annual returns of the S&P 100 index are all positive for six consecutive years from 2010 to 2015, which generally resembles a "recovery period" of the stock market following the 2008 stock market falls. Since the expected stock returns is actually nonzero for the selected period of time, it makes sense to consider model 2 as a good model for the given data. Finally, the model selection results provided by the information criteria can also be confirmed by the distributions of the fitted model parameters. From the MCMC samples of model parameters, we can tell that some of the tested models may not be good candidates for the real data set. For example, model 6 is not considered as a good model for the data set according to the IC results, which agrees with the fact that one of its unique model parameters, β, is not significantly different from 0. According to the MCMC samples, the average of β in model 6 is -0.0371 with a standard deviation of 0.0298, indicating the 95% confidence interval of the parameter contains 0. Therefore, the parameter β in model 6 is not significantly different from 0, suggesting the model may not be a good choice for the given set of data.

References

Alder, B. J. and Wainwright, T. (1959), “Studies in molecular dynamics. I. General method,” *The Journal of Chemical Physics*, 31, 459–466.

Andersen, H. C. (1980), “Molecular dynamics simulations at constant pressure and/or temperature,” *The Journal of Chemical Physics*, 72, 2384–2393.

Andersen, T. G., Chung, H.-J., and Sørensen, B. E. (1999), “Efficient method of moments estimation of a stochastic volatility model: A Monte Carlo study,” *Journal of Econometrics*, 91, 61–87.

Andrews, D. F. and Mallows, C. L. (1974), “Scale mixtures of normal distributions,” *Journal of the Royal Statistical Society. Series B (Methodological)*, 36, 99–102.

Berg, A., Meyer, R., and Yu, J. (2004), “Deviance information criterion for comparing stochastic volatility models,” *Journal of Business & Economic Statistics*, 22, 107–120.

Black, F. (1976), “Studies of Stock Market Volatility Changes,” in *Proceedings of the 1976 Meeting of the American Statistical Association, Business and Economic Statistics Section*, pp. 177–181.

Black, F. and Scholes, M. (1973), “The pricing of options and corporate liabilities,” *The Journal of Political Economy*, 81, 637–654.

Bollerslev, T. (1986), “Generalized autoregressive conditional heteroskedasticity,” *Journal of Econometrics*, 31, 307–327.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., and Riddell, A. (2015), “Stan: a probabilistic programming language,” *Journal of Statistical Software*, 10.

Celeux, G., Forbes, F., Robert, C. P., Titterington, D. M., et al. (2006), “Deviance information criteria for missing data models,” *Bayesian analysis*, 1, 651–673.

Cheng, B. (2005), “Yule–Walker Equations,” *Wiley StatsRef: Statistics Reference Online*.

Chernov, M., Gallant, A. R., Ghysels, E., and Tauchen, G. (2003), “Alternative models for stock price dynamics,” *Journal of Econometrics*, 116, 225–257.

Chib, S., Nardari, F., and Shephard, N. (2002), “Markov chain Monte Carlo methods for stochastic volatility models,” *Journal of Econometrics*, 108, 281–316.

Dempster, A. P. (1997), "The direct use of likelihood for significance testing," *Statistics and Computing*, 7, 247–252.

Engle, R. F. (1982), "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation," *Econometrica: Journal of the Econometric Society*, 50, 987–1007.

Engle, R. F. and Ng, V. K. (1993), "Measuring and testing the impact of news on volatility," *The Journal of Finance*, 48, 1749–1778.

Epifani, I., MacEachern, S. N., Peruggia, M., et al. (2008), "Case-deletion importance sampling estimators: Central limit theorems and related results," *Electronic Journal of Statistics*, 2, 774–806.

Gallant, A. R. and Tauchen, G. (1996), "Which moments to match?" *Econometric Theory*, 12, 657–681.

Gelfand, A. E. (1996), "Model determination using sampling-based methods," *Markov Chain Monte Carlo in Practice*, 145–161.

Gelfand, A. E., Dey, D. K., and Chang, H. (1992), "Model determination using predictive distributions with implementation via sampling-based methods," Tech. rep., DTIC Document.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2014), *Bayesian data analysis*, vol. 2, Chapman & Hall/CRC Boca Raton, FL, USA.

Gelman, A., Lee, D., and Guo, J. (2015), "Stan a probabilistic programming language for Bayesian inference and optimization," *Journal of Educational and Behavioral Statistics*, 40, 530–543.

Gelman, A., Shirley, K., et al. (2011), "Inference from simulations and monitoring convergence," *Handbook of Markov Chain Monte Carlo*, 163–174.

Gilks, W. R. (2005), *Markov chain monte carlo*, Wiley Online Library.

Giot, P. and Laurent, S. (2004), "Modelling daily value-at-risk using realized volatility and ARCH type models," *Journal of Empirical Finance*, 11, 379–398.

Guo, J., Lee, D., Sakrejda, K., Gabry, J., Goodrich, B., De Guzman, J., Niebler, E., Heller, T., and Fletcher, J. (2016), *rstan: R Interface to Stan*.

Harvey, A., Ruiz, E., and Shephard, N. (1994), "Multivariate stochastic variance models," *The Review of Economic Studies*, 61, 247–264.

Hull, J. and White, A. (1987), "The pricing of options on assets with stochastic volatilities," *The Journal of Finance*, 42, 281–300.

Jacquier, E., Polson, N. G., and Rossi, P. E. (2012), "Bayesian analysis of stochastic volatility models," *Journal of Business & Economic Statistics*, 20, 69–87.

Kim, S., Shephard, N., and Chib, S. (1998), "Stochastic volatility: likelihood inference and comparison with ARCH models," *The Review of Economic Studies*, 65, 361–393.

Li, L., Qiu, S., Zhang, B., and Feng, C. X. (2015), "Approximating cross-validatory predictive evaluation in Bayesian latent variable models with integrated IS and WAIC," *Statistics and Computing*, 26, 1–17.

Lundblad, C. (2007), "The risk return tradeoff in the long run: 1836–2003," *Journal of Financial Economics*, 85, 123–150.

Lunn, D. J., Thomas, A., Best, N., and Spiegelhalter, D. (2000), "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility," *Statistics and Computing*, 10, 325–337.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, 21, 1087–1092.

Neal, R. M. et al. (2011), "MCMC using Hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, 2, 113–162.

Peruggia, M. (1997), "On the variability of case-deletion Importance sampling Weights in the Bayesian linear model," *Journal of the American Statistical Association*, 92, 199–207.

Plummer, M. (2003), "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling," in *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, Technische Universit at Wien Wien, Austria.

Shephard, N. (1996), "Statistical aspects of ARCH and stochastic volatility," *Monographs on Statistics and Applied Probability*, 65, 1–68.

Spiegelhalter, D., Thomas, A., Best, N., and Lunn, D. (2007), "OpenBUGS user manual, version 3.0. 2," *MRC Biostatistics Unit, Cambridge.*

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and Linde, A. (2014), "The deviance information criterion: 12 years on," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76, 485–493.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and Van Der Linde, A. (2002), "Bayesian measures of model complexity and fit," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64, 583–639.

Taylor, S. J. (1982), "Financial returns modelled by the product of two stochastic processes, a study of daily sugar prices 1961-79," in *Financial Risk Measurement and Management*, Edward Elgar, pp. 441–464.

Vehtari, A. (2001), *Bayesian model assessment and selection using expected utilities*, Helsinki University of Technology.

Vehtari, A. and Lampinen, J. (2002), “Bayesian model assessment and comparison using cross-validation predictive densities,” *Neural Computation*, 14, 2439–2468.

Watanabe, S. (2010), “Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory,” *The Journal of Machine Learning Research*, 11, 3571–3594.

Appendix A

THE DEVIATION OF THE CONDITIONAL DISTRIBUTIONS OF LOG-VOLATILITIES IN THE AR-SV MODELS

A.1 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in AR(1) Process

When the log-volatilities follow an AR(1) process (model 1, 2, 6, 7, and 8), the state equations governing the log-volatility process are identical:

$$\begin{aligned} h_t|h_{t-1}, \mu, \phi, \tau &\sim N(\mu + \phi(h_{t-1} - \mu), \tau^2), t = 1, ..., n; && \text{(A.1)} \\ h_{t+1}|h_t, \mu, \phi, \tau &\sim N(\mu + \phi(h_t - \mu), \tau^2), t = 1, ..., n. && \text{(A.2)} \end{aligned}$$

In the equations above, all the parameters except h_t are treated as known. Based on these known parameters, the distribution of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ can be obtained. Using terminologies of Bayesian inference, the first equation could be considered as the prior distribution of h_t, and the second equation provides information for the posterior distribution of h_t. If we let $\alpha_t = \mu + \phi(h_t - \mu)$, the state equations above can be transformed into:

$$\begin{aligned} \alpha_t|h_{t-1}, \boldsymbol{\theta} &\sim N(\mu - \phi^2\mu + \phi^2 h_{t-1}, \tau^2\phi^2), t = 1, ..., n; && \text{(A.3)} \\ h_{t+1}|\alpha_t, \boldsymbol{\theta} &\sim N(\alpha_t, \tau^2), t = 1, ..., n. && \text{(A.4)} \end{aligned}$$

Therefore, the posterior distribution of α_t is proportional to their product (constant coefficients are neglected):

$$
\begin{aligned}
f(\alpha_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}) &\propto \exp\left[\frac{-0.5(h_{t+1}-\alpha_t)^2}{\tau^2} - \frac{0.5(\alpha_t - \mu + \phi^2\mu - \phi^2 h_{t-1})^2}{\tau^2\phi^2}\right] \qquad (A.5)\\
&\propto \exp\left[-0.5\left(\alpha_t - \frac{\frac{h_{t+1}}{\tau^2} + \frac{\mu - \phi^2\mu + \phi^2 h_{t-1}}{\tau^2\phi^2}}{\frac{1}{\tau^2} + \frac{1}{\tau^2\phi^2}}\right)^2 \left(\frac{1}{\tau^2} + \frac{1}{\tau^2\phi^2}\right)\right],\\
& t = 1, ..., n.
\end{aligned}
$$

From the formula above, we can tell that the posterior of $\alpha_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ follows the following normal distribution:

$$
\begin{aligned}
\alpha_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} &\sim N\left(\frac{\frac{h_{t+1}}{\tau^2} + \frac{\mu - \phi^2\mu + \phi^2 h_{t-1}}{\tau^2\phi^2}}{\frac{1}{\tau^2} + \frac{1}{\tau^2\phi^2}}, \frac{1}{\frac{1}{\tau^2} + \frac{1}{\tau^2\phi^2}}\right) \qquad (A.6)\\
&\sim N\left(\frac{\phi^2 h_{t+1} + \mu - \phi^2\mu + \phi^2 h_{t-1}}{\phi^2+1}, \frac{\tau^2\phi^2}{\phi^2+1}\right), t = 1, ..., n.
\end{aligned}
$$

Since $\alpha_t = \mu + \phi(h_t - \mu)$, we have $h_t = (\alpha_t - \mu + \phi\mu)/\phi$. As a result, for $t = 1, ..., n$, the posterior of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ follows the following normal distribution:

$$
\begin{aligned}
h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} &\sim N\left(\frac{\left(\frac{\phi^2 h_{t+1} + \mu - \phi^2\mu + \phi^2 h_{t-1}}{\phi^2+1} - \mu + \phi\mu\right)}{\phi}, \frac{\tau^2}{\phi^2+1}\right) \qquad (A.7)\\
&\sim N\left(\frac{\phi}{\phi^2+1}h_{t+1} + \frac{\phi}{\phi^2+1}h_{t-1} + \frac{\mu}{\phi(\phi^2+1)} - \frac{\phi\mu}{\phi^2+1} - \frac{\mu}{\phi} + \mu,\right.\\
&\left.\frac{\tau^2\phi^2}{\phi^2+1}\right).
\end{aligned}
$$

A.2 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in AR(2) Process

When the log-volatilities follow an AR(2) process (model 3), the current h_t depends on both h_{t-1} and h_{t-2}. Therefore, for $t = 1, ..., n$, we have the following state equations related to h_t:

$$\begin{aligned} h_t|h_{t-1}, h_{t-2}, \mu, \phi, \tau &\sim N(\mu + \phi(h_{t-1} - \mu) + \psi(h_{t-2} - \mu), \tau^2); && \text{(A.8)} \\ h_{t+1}|h_t, h_{t-1}, \mu, \phi, \tau &\sim N(\mu + \phi(h_t - \mu) + \psi(h_{t-1} - \mu), \tau^2); && \text{(A.9)} \\ h_{t+2}|h_t, h_{t+1}, \mu, \phi, \tau &\sim N(\mu + \phi(h_{t+1} - \mu) + \psi(h_t - \mu), \tau^2). && \text{(A.10)} \end{aligned}$$

The distribution of $h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t}$ for model 3 can be similarly calculated. Let $\alpha_t = \psi\phi(h_t - \mu), \beta_t = \psi h_{t+1}$, and $\gamma_t = \phi h_{t+2}$, then for $t = 1, ..., n$, the above state equations can be written as follows:

$$\begin{aligned} \alpha_t|h_{t-1}, h_{t-2}, \boldsymbol{\theta} &\sim N(\phi\psi[\phi(h_{t-1} - \mu) + \psi(h_{t-2} - \mu)], \tau\phi\psi); && \text{(A.11)} \\ \beta_t|\alpha_t, h_{t-1}, \boldsymbol{\theta} &\sim N(\psi\mu + \alpha_t + \psi^2(h_{t-1} - \mu), \tau\psi); && \text{(A.12)} \\ \gamma_t|\alpha_t, \beta_t, \boldsymbol{\theta} &\sim N\left(\phi\mu + \alpha_t + \phi^2\left(\frac{\beta_t}{\psi} - \mu\right), \tau\phi\right). && \text{(A.13)} \end{aligned}$$

The first equation can be treated as a prior distribution of α_t, and the second and the third equations both provide information on the posterior distribution of α_t. Therefore, the posterior distribution of α_t is proportional to the product of the three (neglecting constant coefficients):

$$\begin{aligned} f(\alpha_t|h_{t-1}, h_{t-2}, \beta_t, \gamma_t, \boldsymbol{\theta}) &\propto f(\alpha_t|h_{t-1}, h_{t-2}, \boldsymbol{\theta})f(\beta_t|\alpha_t, h_{t-1}, \boldsymbol{\theta})f(\gamma_t|\alpha_t, \beta_t, \boldsymbol{\theta}) && \text{(A.14)} \\ &\propto \exp\left\{-\frac{[\alpha_t - \phi\psi(\phi h_{t-1} - \phi\mu + \psi h_{t-2} - \psi\mu)]^2}{2\tau^2\phi^2\psi^2}\right. \\ &\quad -\frac{[\beta_t - (\psi\mu + \alpha_t + \psi^2 h_{t-1} - \psi^2\mu)]^2}{2\tau^2\psi^2} \\ &\quad \left.-\frac{\left[\gamma_t - \left(\phi\mu + \alpha_t + \frac{\phi^2\beta_t}{\psi} - \phi^2\mu\right)\right]^2}{2\tau^2\phi^2}\right\}, t = 1, ..., n. \end{aligned}$$

If we let $a_t = \mu + \phi(h_{t-1} - \mu) + \psi(h_{t-2} - \mu), b_t = \beta_t - \psi\mu - \psi^2(h_{t-1} - \mu)$, and $c_t = \gamma_t - \phi\mu - \phi^2(\beta_t/\psi - \mu)$, then the desired density function $f(\alpha_t|h_{t-1}, h_{t-2}, \beta_t, \gamma_t, \boldsymbol{\theta})$ can be written as:

$$
\begin{aligned}
f(\alpha_t|h_{t-1}, h_{t-2}, \beta_t, \gamma_t, \boldsymbol{\theta}) &\propto \exp\left[\frac{-0.5(\alpha_t - a)^2}{\tau^2\phi^2\psi^2} - \frac{-0.5(\alpha_t - b)^2}{\tau^2\psi^2} - \right. \\
&\qquad \left. \frac{-0.5(\alpha_t - c)^2}{\tau^2\phi^2}\right] \\
&\propto \exp\left[-\frac{1}{2}\left(\alpha_t - \frac{\frac{a_t}{\tau^2\phi^2\psi^2} + \frac{b_t}{\tau^2\psi^2} + \frac{c_t}{\tau^2\phi^2}}{\frac{1}{\tau^2\phi^2\psi^2} + \frac{1}{\tau^2\psi^2} + \frac{1}{\tau^2\phi^2}}\right)^2 \right. \\
&\qquad \left. \left(\frac{1}{\tau^2\phi^2\psi^2} + \frac{1}{\tau^2\psi^2} + \frac{1}{\tau^2\phi^2}\right)\right], t = 1, ..., n.
\end{aligned}
\tag{A.15}
$$

The equation above shows that the posterior distribution of α_t is normal:

$$
\begin{aligned}
\alpha_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} &\sim N\left(\frac{\frac{a_t}{\tau^2\phi^2\psi^2} + \frac{b_t}{\tau^2\psi^2} + \frac{c_t}{\tau^2\phi^2}}{\frac{1}{\tau^2\phi^2\psi^2} + \frac{1}{\tau^2\psi^2} + \frac{1}{\tau^2\phi^2}}, \sqrt{\frac{1}{\frac{1}{\tau^2\phi^2\psi^2} + \frac{1}{\tau^2\psi^2} + \frac{1}{\tau^2\phi^2}}}\right) \\
&\sim N\left(\frac{a_t + \phi^2 b_t + \psi^2 c_t}{1 + \phi^2 + \psi^2}, \frac{\tau\phi\psi}{\sqrt{1 + \phi^2 + \psi^2}}\right), t = 1, ..., n.
\end{aligned}
\tag{A.16}
$$

Since $\alpha_t = \psi\phi(\boldsymbol{h}_t - \mu)$, we have $h_t = \alpha_t/(\psi\phi) + \mu$. Therefore, the distribution of $h_t|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ is normal as well:

$$
\begin{aligned}
h_t|\boldsymbol{\theta}, \boldsymbol{h}_{-t} &\sim N\left(\frac{a_t + \phi^2 b_t + \psi^2 c_t}{\phi\psi(1 + \phi^2 + \psi^2)} + \mu, \frac{\tau}{\sqrt{1 + \phi^2 + \psi^2}}\right) \\
&\sim N\left(\frac{\phi + \psi - \phi\psi}{1 + \phi^2} h_{t-1} + \frac{\phi}{1 + \phi^2} h_{t+1} + \frac{\phi\psi\mu - 2\phi\mu - \psi\mu}{1 + \phi^2} + \mu, \right. \\
&\qquad \left. \frac{\tau^2}{1 + \phi^2}\right), t = 1, ..., n.
\end{aligned}
\tag{A.17}
$$

A.3 Conditional Distribution of h_t Given $\boldsymbol{h}_{-t}$ in Two Independent AR(1) Processes

When there are two independent log-volatility processes going on simultaneously (model 4), the state equations are given by:

$$h_t^{(1)}|h_{t-1}^{(1)}, \phi, \tau \sim N(\phi \boldsymbol{h}_{t-1}^{(1)}, \tau^2), \tag{A.18}$$

$$h_{t+1}^{(1)}|h_t^{(1)}, \phi, \tau \sim N(\phi h_t^{(1)}, \tau^2), \tag{A.19}$$

$$h_t^{(2)}|h_{t-1}^{(2)}, \phi_2, \tau_2 \sim N(\phi_2 h_{t-1}^{(2)}, \tau_2^2), \tag{A.20}$$

$$h_{t+1}^{(2)}|h_t^{(2)}, \phi_2, \tau_2 \sim N(\phi_2 h_t^{(2)}, \tau_2^2), \tag{A.21}$$

where $t = 1, ..., n$. In this model, each of the log-volatility processes can be treated as a special case of the log-volatility process in the basic SV model (model 1) with $\mu = 0$. As a result, the calculations of the distributions of $h_t^{(1)}|\boldsymbol{h}_{-t}^{(1)}, \boldsymbol{\theta}$ and $h_t^{(2)}|\boldsymbol{h}_{-t}^{(2)}, \boldsymbol{\theta}$ are very similar to the basic SV model. If we start from the conditional distribution of $h_t|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ for the basic SV model, then by setting $\mu = 0$ and replacing h_i with $h_i^{(1)}$, the distribution of $h_t^{(1)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ can be obtained:

$$h_t^{(1)}|\boldsymbol{h}_{-t}, \phi, \tau \sim N\left(\frac{\phi(h_{t+1}^{(1)} + h_{t-1}^{(1)})}{1+\phi^2}, \frac{\tau^2\phi^2}{1+\phi^2}\right), t = 1, ..., n. \tag{A.22}$$

The calculation of the distribution of $h_t^{(2)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ is identical to $h_t^{(1)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$. The final result for the calculated distribution of $h_t^{(2)}|\boldsymbol{h}_{-t}, \boldsymbol{\theta}$ is given by:

$$h_t^{(2)}|\boldsymbol{h}_{-t}, \phi_2, \tau_2 \sim N\left(\frac{\phi_2(h_{t+1}^{(2)} + h_{t-1}^{(2)})}{1+\phi_2^2}, \frac{\tau_2^2\phi_2^2}{1+\phi_2^2}\right), t = 1, ..., n. \tag{A.23}$$

Appendix B

R CODE FOR THE STOCHASTIC VOLATILITY MODELS

B.1 Generating Simulated Data Sets From Model 6

```
##########################################
## The following R code generates 100 sets of simulated data from model 6.
## The data sets are stored in the current folder once they are generated.
##########################################
## y --- The simulated data set.
## mu, phi, tau, beta, kappa, delta, and h -- The true values of model parameters.

for (ifold in 1:100){
  D <- 1900
  T <- 3900
  mu <- -10
  phi <- 0.96
  tau <- 0.345
  beta <- 0.1
  kappa <- 0.08
  delta <- 0.03
  h0 <- 2
  s <- lss <- y<- h <- qq <- rep (0, T)
  h[1] <- rnorm (1, mu + phi * (h0 - mu), tau)
  for (t in 1:T) {lss[t] <- rnorm(1, -(delta^2)/2,delta^2); s[t] <- exp (lss[t]) -
       1}
  for (t in 1:T) qq[t] <- rbinom(1, 1, kappa)
  for (i in 2:T) h[i] <- rnorm (1, mu + phi* (h[i-1] - mu), tau)
  y[1] <- rnorm (1, s[1]*qq[1], exp (h[1]/2) )
  for (t in 2:T) y[t] <- rnorm (1, beta*y[t-1]+s[t]*qq[t] , exp (h[t]/2) )
  x <- rnorm (T)

  x <- x[-(1:D)]
  y <- y[-(1:D)]
  h <- h[-(1:D)]
  T <- T - D

  rm (.Random.seed)

  save.image (file = sprintf ("sample_data%d.RData", ifold))
}
```

B.2 Model Specifications and Model Fittings

The entire Section B.2 is written by Zhouji Zheng.

- **Model 1**

  ```
  ###########################################
  ## The following R code defines model 1 in rstan language.
  ###########################################

  library(rstan)
  model1 <- '
    data {
      int<lower=1> T;
      vector[T] y;
    }

    parameters {
      real mu;
      real<lower=0,upper=1> phi;
      real<lower=0.01> tausq;
      real h0;
      vector[T] h;
    }

    transformed parameters {
      real<lower=0> tau;
      tau <- sqrt(tausq);
    }

    model {
      mu ~ normal(-10,5);
      phi ~ beta(1, 1);
      tausq ~ inv_gamma(2.5, 0.025);
      h0 ~ normal(0, 10000);
      h[1] ~ normal(mu + phi*(h0-mu), tau);
      for (t in 2:T)
        h[t] ~ normal(mu + phi*(h[t-1]-mu), tau);
      for (t in 1:T)
        y[t] ~ normal(0,exp(h[t]/2));
    }
  ```

```
  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t],0,exp (h[t]/2) );

    }
  }
'

###########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
###########################################

load ("sample_data.RData")
fit <- stan(model_code = model1, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 2**

```
###########################################
## The following R code defines model 2 in rstan language.
###########################################

library (rstan)
model2 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0,upper=1> phi;
    real<lower=0.01> tausq;
    real alpha;
    real h0;
    vector[T] h;
  }

  transformed parameters {
    real<lower=0> tau;
    tau <- sqrt(tausq);
  }
  model {

    mu ~ normal(-10,5);
    phi ~ beta(10, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    alpha ~ normal(0, sqrt(10));
    h0 ~ normal(0, 10000);
    h[1] ~ normal(mu + phi*(h0-mu), tau);
    for (t in 2:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu), tau);
    for (t in 1:T)
      y[t] ~ normal(alpha, exp(h[t]/2));
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t],alpha,exp (h[t]/2) );
    }
  }
'

###########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
###########################################

load ("sample_data.RData")
fit <- stan(model_code = model2, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 3**

```
###########################################
## The following R code defines model 3 in rstan language.
###########################################

library (rstan)
model3 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0.01> tausq;
    real h0;
    vector[T] h;
    real<lower=-1,upper=1> r1;
    real<lower=-1,upper=1> r2;

  }

  transformed parameters {
    real phi;
    real psi;
    real<lower=0> tau;
    phi <- r1;
    psi <- r2;
    tau <- sqrt(tausq);
  }

  model {
    r1 ~ uniform(0,1);
    r2 ~ uniform(0, 1);
    mu ~ normal(-10,5);
    tausq ~ inv_gamma(2.5, 0.025);
    h0 ~ normal(0, 10000);
    h[1] ~ normal(mu + phi*(h0-mu), tau);
    h[2] ~ normal(mu + phi*(h[1]-mu)+ psi*(h0-mu), tau);
    for (t in 3:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu)+ psi*(h[t-2]-mu), tau);
    for (t in 1:T)
      y[t] ~ normal(0,exp(h[t]/2));
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t], 0 ,exp (h[t]/2) );
    }
  }
'

###########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
###########################################

load ("sample_data.RData")
fit <- stan(model_code = model3, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
```

```
save (fit, file = "sample_fit")
```

- **Model 4**

```
##########################################
## The following R code defines model 4 in rstan language.
##########################################

library (rstan)
model4 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0,upper=1> phi1;
    real<lower=0,upper=1> phi2;
    real<lower=0.0001> tausq;
    real<lower=0.0001> tau2sq;
    real h10;
    real h20;
    vector[T] h1;
    vector[T] h2;
  }

  transformed parameters {
    real<lower=0> tau;
    real<lower=0> tau2;
    tau <- sqrt(tausq);
    tau2 <- sqrt(tau2sq);
  }

  model {

    mu ~ normal(-10,5);
    phi1 ~ beta(1, 1);
    phi2 ~ beta(1, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    tau2sq ~ inv_gamma(2.5, 0.025);
    h10 ~ normal(0, 10000);
    h20 ~ normal(0, 10000);
    h1[1] ~ normal(phi1*h10, tau);
    h2[1] ~ normal(phi2*h20, tau2);
    for (t in 2:T)
      h1[t] ~ normal(phi1*h1[t-1], tau);
    for (t in 2:T)
```

```
      h2[t] ~ normal(phi2*h2[t-1], tau2);
    for (t in 1:T)
      y[t] ~ normal(0,exp (mu/2 + h1[t]/2+ h2[t]/2 ));
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t], 0 ,exp (mu/2 + h1[t]/2+ h2[t]/2 ));
    }
  }
'

##########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
##########################################

load ("sample_data.RData")
fit <- stan(model_code = model4, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 5**

```
##########################################
## The following R code defines model 5 in rstan language.
##########################################

library (rstan)
model5 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=-1,upper=1> phis;
    real<lower=0.0001> tausq;
    real<lower=0,upper=1> rho;
    real h0;
    real y0;
    vector[T] h;
  }

  transformed parameters {
    real phi;
    real<lower=0> tau;
    phi <- 2*phis - 1;
    tau <- sqrt(tausq);
  }

  model {
    mu ~ normal(-10,5);
    phis ~ uniform(-1, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    rho ~ uniform(-1,1);
    h0 ~ normal(0, 10000);
    h[1] ~ normal(mu + phi*(h0-mu) + rho*tau*exp(-0.5*h0)*y0, tau*sqrt(1-
        square(rho)));
    for (t in 2:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu) + rho*tau*exp(-0.5*h[t-1])*y[t-1],
          tau*sqrt(1-square(rho)));

    for (t in 2:T)
      y[t] ~ normal(0,exp(h[t]/2));
    y0 ~ normal(0,exp(h0/2));
    y[1] ~ normal(0,exp(h[1]/2));

  }

  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t], 0 ,exp (h[t]/2) );
    }
  }
'

###########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
###########################################

load ("sample_data.RData")
fit <- stan(model_code = model5, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 6**

```
###########################################
## The following R code defines model 6 in rstan language.
```

```
##########################################

library ("rstan")
model6 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0,upper=1> phi;
    real<lower=0.01> tausq;
    real beta;
    real<lower=0,upper=1> kappa;
    real h0;
    real y0;
    real ldelta;
    vector[T] h;
    real l[T];
    vector[T] q;
  }

  transformed parameters {
    real<lower=0> tau;
    real delta;
    vector[T] s;
    tau <- sqrt(tausq);
    delta <- exp(ldelta);
    for (t in 1:T)
    s[t] <- exp(l[t])-1;
  }

  model {
    matrix[T,2] logpy;
    mu ~ normal(-10,5);
    phi ~ beta(1, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    kappa ~ beta(2,100);
    beta ~ normal(0,sqrt(0.2));
    ldelta ~ normal(-3.07, sqrt(0.149));
    h0 ~ normal(mu, tau);
    h[1] ~ normal(mu + phi*(h0-mu), tau);
    for (t in 1:T)
      l[t] ~ normal(-square(delta)/2,square(delta));
    for (t in 2:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu), tau);
    for (t in 2:T)
    {
        increment_log_prob (
                log_sum_exp(
                  log (1-kappa) + normal_log(y[t], beta*y[t-1],exp(h[t]/2)),
                  log (kappa) + normal_log(y[t], beta*y[t-1]+s[t],exp(h[t]/2))
                )
        );
    }
    y0 ~ normal(0, exp(h0/2));

    increment_log_prob (
                log_sum_exp(
                  log (1-kappa) + normal_log(y[1], beta*y0,exp(h[1]/2)),
                  log (kappa) + normal_log(y[1], beta*y0+s[1],exp(h[1]/2))
                )
    );
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 2:T){
      log_lik[t] <- log_sum_exp(
                      log (1-kappa) + normal_log(y[t], beta*y[t-1],exp(h[t
                         ]/2)),
                       log (kappa) + normal_log(y[t], beta*y[t-1]+s[t],exp(h[
                         t]/2))
                    );
    }
    log_lik[1] <-  log_sum_exp(
                      log (1-kappa) + normal_log(y[1], beta*y0,exp(h[1]/2)),
                      log (kappa) + normal_log(y[1], beta*y0+s[1],exp(h[1]/2)
                         )
    );
  }
'

##########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
##########################################

load ("sample_data.RData")
fit <- stan(model_code = model6, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 7**

```
##########################################
## The following R code defines model 7 in rstan language.
##########################################

library (rstan)
model7 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0, upper=1> phis;
    real<lower=0.01> tausq;
    real<lower=0,upper=1> kappa;
    real h0;
    real y0;
    real ldelta;
    vector[T] h;
    real l[T];
    vector[T] q;
  }

  transformed parameters {
    real<lower=0> tau;
    real phi;
    real delta;
    vector[T] s;
    tau <- sqrt(tausq);
    delta <- exp(ldelta);
    phi <- phis;
    for (t in 1:T)
    s[t] <- exp(l[t])-1;
  }

  model {
    mu ~ normal(-10,5);
    phis ~ beta(1, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    kappa ~ beta(2,100);
    ldelta ~ normal(-3.07, sqrt(0.149));
    h0 ~ normal(mu, tau);
    h[1] ~ normal(mu + phi*(h0-mu), tau);
    for (t in 1:T)
```

```
      l[t] ~ normal(-square(delta)/2,square(delta));
    for (t in 2:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu), tau);
    for (t in 2:T)
    {
        increment_log_prob (
                log_sum_exp(
                  log (1-kappa) + normal_log(y[t], 0,exp(h[t]/2)),
                  log (kappa) + normal_log(y[t], s[t],exp(h[t]/2))
                )
        );
    }
    y0 ~ normal(0, exp(h0/2));

    increment_log_prob (
                log_sum_exp(
                  log (1-kappa) + normal_log(y[1], 0,exp(h[1]/2)),
                  log (kappa) + normal_log(y[1], s[1],exp(h[1]/2))
                )
    );
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 2:T){
      log_lik[t] <- log_sum_exp(
                      log (1-kappa) + normal_log(y[t], 0,exp(h[t]/2)),
                       log (kappa) + normal_log(y[t], s[t],exp(h[t]/2))
                   );
    }
    log_lik[1] <-  log_sum_exp(
                      log (1-kappa) + normal_log(y[1], 0,exp(h[1]/2)),
                      log (kappa) + normal_log(y[1], s[1],exp(h[1]/2))
    );
  }
'

##########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
##########################################

load ("sample_data.RData")
fit <- stan(model_code = model7, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

- **Model 8**

```
###########################################
## The following R code defines model 8 in rstan language.
###########################################

library (rstan)
model8 <- '
  data {
    int<lower=1> T;
    vector[T] y;
  }

  parameters {
    real mu;
    real<lower=0, upper=1> phis;
    real<lower=0.01> tausq;
    real<lower=0> nu;
    real h0;
    vector[T] h;
    real<lower=0> w[T];
  }

  transformed parameters {
    real phi;
    real<lower=0> tau;
    tau <- sqrt(tausq);
    phi <- 2*phis - 1;
  }

  model {
    mu ~ normal(-10,5);
    phis ~ uniform(0.5, 1);
    tausq ~ inv_gamma(2.5, 0.025);
    h0 ~ normal(0, 10000);
    h[1] ~ normal(mu + phi*(h0-mu), tau);
    nu ~ uniform(2,128);
    for (t in 1:T)
      w[t] ~ gamma(nu/2,nu/2);
    for (t in 2:T)
      h[t] ~ normal(mu + phi*(h[t-1]-mu), tau);
    for (t in 1:T)
      y[t] ~ normal(0,exp (h[t]/2)/sqrt(w[t]));
  }

  generated quantities {
    vector[T] log_lik;
    for (t in 1:T){
      log_lik[t] <- normal_log (y[t], 0 ,exp (h[t]/2)/sqrt(w[t]) );
    }
  }
'

###########################################
## The following R code fits the model with a given set of data using HMC
   method.
## Two independent Markov chains are generated from each set of tested data.
###########################################

load ("sample_data.RData")
fit <- stan(model_code = model8, data = list(y = y, T = T), iter = 20000,
   chains = 2, thin = 10)
save (fit, file = "sample_fit")
```

Appendix C
R CODE FOR CALCULATING THE INFORMATION CRITERIA

C.1 Utility Functions

C.1.1 Sums and Means

```
##########################################
## The following functions are used in the calculations of the information
   criteria.
## log_sum_exp --- This function reads in a set of log-scaled data, and outputs
   the logarithm of the sum of the data.
## log_mean_exp --- This function reads in a set of log-scaled data, and outputs
   the logarithm of the mean of the data.
## log_hmean_exp --- This function reads in a set of log-scaled data, and outputs
   the logarithm of the harmonic mean of the data.
## log_sum_exp_mat --- This function reads in a matrix of log-scaled data, and
   outputs the logarithm of the sums of the data by the rows.
## log_mean_exp_mat --- This function reads in a matrix of log-scaled data, and
   outputs the logarithm of the means of the data by the rows.
##########################################

log_sum_exp <- function (lx)
{
  mlx <- max (lx)
  log (sum (exp (lx - mlx))) + mlx
}

log_mean_exp <- function (lx)
{
  log_sum_exp (lx) - log(length (lx))
}

log_hmean_exp <- function (lx)
{
  - log_mean_exp (-lx)
}

log_sum_exp_mat <- function (lx)
{
  mlx <- apply(lx, 1, max)
  log (rowSums (exp (lx - mlx))) + mlx
}

log_mean_exp_mat <- function (lx)
{
  log_sum_exp_mat (lx) - log(length (lx[1, ]))
}
```

C.1.2 WAIC and IS-IC

```
##########################################
## logp.tr is a matrix storing log P (y_i|theta_j) with cols for cases (y_i) and
   rows for theta_j (mc samples).
## is_function --- This function calculates the IS-IC value for a model with
   respect to its fitted data.
## waic_function --- This function calculates the WAIC value for a model with
   respect to its fitted data.
##########################################

is_function <- function (logp.tr){
  logp_is <- apply (logp.tr, 2, log_hmean_exp)
  logp_within_sample <- apply (logp.tr,2, log_mean_exp)
  p_is <- sum(logp_within_sample - logp_is)
  is <- -2 * sum (logp_is)

  list (logp_is = logp_is, is = -2 * sum (logp_is), p_is = p_is,
        total = c(is=is, p_is = p_is))
}

waic_function <- function (logp.tr){
  mlogp <- apply (logp.tr, 2, log_mean_exp)
  vlogp <- apply (logp.tr, 2, var)
  logp_waic <- mlogp - vlogp
  p_waic <- sum(vlogp)
  waic <- -2 * sum (logp_waic)

  list (logp_waic = logp_waic, waic = -2 * sum (logp_waic), p_waic = p_waic,
        total = c(waic=waic, p_waic=p_waic))
}
```

C.1.3 Functions Related to the Calculation of the Integrals in Model 5

```
##########################################
## log_f --- a function computing the logarithm of the integrant function
## range --- the range of integral varaible, a vector of two elements
## n     --- the number of points at which the integrant is evaluated
## ...   --- other parameters needed by log_f
## log_int_mid --- This function reads in the log of a function and outputs the
   log of the integral result.
##########################################

log_int_mid <- function(log_f, range, n,...)
{  if(range[1] >= range[2])
  stop("Wrong ranges")
   h <- (range[2]-range[1]) / n
   v_log_f <- sapply(range[1] + (1:n - 0.5) * h, log_f,...)
   log_sum_exp_mat(v_log_f) + log(h)
}

##########################################
## log_f_num --- This function gives the logarithm of the numerator value before
   the integration in model 5.
## log_f_denum --- This function gives the logarithm of the denominator value
   before the integration in model 5.
## log_f_num_transformed --- This function substitutes the "hcondi" in the
   log_f_num function with "mu0+log(k/(1-k))".
## log_f_denum_transformed --- This function substitutes the "hcondi" in the
   log_f_denum function with "mu0+log(k/(1-k))".
##########################################

log_f_num <- function (mu, phi, tau, rho, hcondi, htplus, htminus, yt, ytminus){

  dnorm (yt, 0, exp(hcondi/2), log = T) +
  -.5*(hcondi-mu-phi*htminus+phi*mu-rho*tau*ytminus*exp(-.5*htminus))^2 / (tau
     ^2*(1-rho^2)) +
  -.5*(htplus-mu-phi*hcondi+phi*mu-rho*tau*yt*exp(-.5*hcondi))^2 / (tau^2*(1-rho
     ^2))
}

log_f_num_transformed <- function (mu, phi, tau, rho, k, htplus, htminus, yt,
   ytminus){
  mu0 <- (htplus + htminus)/2
  log_f_num(mu = mu, phi = phi, tau = tau, rho=rho, hcondi=mu0+log(k/(1-k)),
     htplus=htplus, htminus=htminus, yt=yt, ytminus=ytminus)-log(k-k^2)
}

log_f_denum <- function (mu, phi, tau, rho, hcondi, htplus, htminus, yt, ytminus){

   -.5*(hcondi-mu-phi*htminus+phi*mu-rho*tau*ytminus*exp(-.5*htminus))^2 / (tau
      ^2*(1-rho^2)) +
    -.5*(htplus-mu-phi*hcondi+phi*mu-rho*tau*yt*exp(-.5*hcondi))^2 / (tau^2*(1-rho
        ^2))
}


log_f_denum_transformed <- function (mu, phi, tau, rho, k, htplus, htminus, yt,
    ytminus){
  mu0 <- (htplus + htminus)/2
  log_f_denum(mu = mu, phi = phi, tau = tau, rho=rho, hcondi=mu0+log(k/(1-k)),
      htplus=htplus, htminus=htminus, yt=yt, ytminus=ytminus)-log(k-k^2)
}
```

C.2 Computing the DIC, WAIC, and IS for the SV Models

- **Model 1**

```
##########################################
## Sourcing functions, loading original data observations and the fitted
    chains
##########################################

source ("iswaic.r")
source ("model1.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
```

```
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

##########################################
## Computing log integrated likelihood
##########################################
time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  y <- raw_data$y
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  h <- fit_ss$h
  J <- 100
  hcondi <- array(0, dim = c(I, T, J))
  a <- phi / (phi^2 + 1)
  b <- phi / (phi^2 + 1)
  int <- ((mu - phi^2 * mu) / (phi^2 + 1) - mu + phi * mu) / phi
  sd <- tau / (phi^2 + 1)^0.5
  hcondi[, 1, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 2] + int, sd)
  hcondi[, T, ] <- rnorm(J * I, mu - phi * mu + phi * h[, T - 1], tau)

  for (t in 2:(T - 1)) {
    hcondi[,t,] <- rnorm(J * I, a * h[, t - 1] + b * h[, t +1 ] + int, sd)
  }
  thcondi <- aperm(hcondi, c(2, 1, 3))
  log_py <- dnorm(y, mean = 0, sd = exp(thcondi /2 ), log = T)
  log_ipy<-log_mean_exp_arr(log_py)
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

##########################################
## Creating files for the calculated results
##########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 2**

```
##########################################
## Sourcing functions, loading original data observations and the fitted
    chains
##########################################

source ("iswaic.r")
source ("model2.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

##########################################
```

```
## Computing log integrated likelihood
###########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  y <- raw_data$y
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  h <- fit_ss$h
  J <- 100
  hcondi <- array(0, dim = c(I, T, J))
  a <- phi / (phi^2 + 1)
  b <- phi / (phi^2 + 1)
  int <- ((mu - phi^2 * mu) / (phi^2 + 1) - mu + phi * mu) / phi
  sd <- tau / (phi^2 + 1)^0.5
  hcondi[, 1, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 2] + int, sd)
  hcondi[, T, ] <- rnorm(J * I, mu - phi * mu + phi * h[, T - 1], tau)

  for (t in 2:(T - 1)) {
    hcondi[,t,] <- rnorm(J * I, a * h[, t - 1] + b * h[, t +1 ] + int, sd)
  }

  thcondi <- aperm(hcondi, c(2, 1, 3))
  log_py <- dnorm(y, mean = 0, sd = exp(thcondi /2 ), log = T)
  dim(log_py)
  log_ipy<-log_mean_exp_arr(log_py)
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
```

```
iistime <- time5 +time7

##########################################
## Creating files for the calculated results
##########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 3**

```
##########################################
## Sourcing functions, loading original data observations and the fitted
    chains
##########################################

source ("iswaic.r")
source ("model3.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

##########################################
## Computing log integrated likelihood
##########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  psi <- fit_ss$psi
  tau <- fit_ss$tau
  Y <- raw_data$y
  h <- fit_ss$h
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  J <- 100
  hcondi <- array(0, dim = c(I, T, J))
  a <- psi / (1 + phi^2 + psi^2)
  b <- (phi - psi * phi) / (1 + phi^2 + psi^2)
  c <- (phi - psi * phi) / (1 + phi^2 + psi^2)
  d <- psi / (1 + phi^2 + psi^2)
  int <- (-2 * phi * mu - 2 * psi * mu + 2 * phi * psi * mu) / (1 + phi^2 +
      psi^2) + mu
  sd <- tau / sqrt(phi^2 + psi^2 + 1)
  hcondi[, 1, ] <- rnorm(J * I, (phi * (fit_ss$h0 - mu) + phi * (h[, 2] - mu
      - psi * (fit_ss$h0 - mu)) + psi * (h[, 3] - mu - phi * (h[, 2] - mu)))
      /(1 + phi^2 + psi^2) + mu, sd)
  hcondi[, 2, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 1] + c * h[, 3] + d *
       h[, 4] + int, sd)
  hcondi[, T - 1, ] <- rnorm(J * I, (phi * (h[, T - 2] - mu) + psi * (h[, T -
       3] - mu) + phi * (h[, T] - mu - psi * (h[, T - 2] - mu))) / (1 + phi
      ^2) + mu, tau / sqrt(1 + phi^2))
  hcondi[, T, ] <- rnorm (J * I, mu + phi * (h[, T - 1] - mu) + psi * (h[, T
      - 2] - mu), tau)

  for (t in 3:(T - 2)) {
    hcondi[, t, ] <- rnorm(J * I, a * h[, t - 2] + b * h[, t - 1] + c * h[, t
         + 1] + d * h[, t + 2] + int, sd)
  }


  thcondi <- aperm(hcondi, c(2, 1, 3))
  log_py <- dnorm(y, mean = 0, sd = exp(thcondi /2 ), log = T)
  log_ipy<-log_mean_exp_arr(log_py)
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
  )

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

###########################################
## Creating files for the calculated results
###########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")
```

```
system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 4**

```
##########################################
## Sourcing functions, loading original data observations and the fitted
    chains
##########################################

source ("iswaic.r")
source ("model4.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

##########################################
## Computing log integrated likelihood
##########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  phi2 <- fit_ss$phi2
  tau <- fit_ss$tau
  tau2 <- fit_ss$tau2
  h1 <- fit_ss$h1
  h2 <- fit_ss$h2
  y <- raw_data$y
  I <- length(fit_ss$h1[, 1])
  T <- length(fit_ss$h1[1, ])
  J <- 100
  h1condi <- array (0, dim = c(I, T, J))
  h2condi <- h1condi
  log_py <- array(0, dim = c(I, T, J))
  log_ipy <- matrix(0, nrow = I, ncol = T)
  a1 <- phi / (phi^2 + 1)
  b1 <- phi / (phi^2 + 1)
  sd1 <- tau / sqrt(phi^2 + 1)
  a2 <- phi2 / (phi2^2 + 1)
  b2 <- phi2 / (phi2^2 + 1)
  sd2 <- tau2 / sqrt(phi2^2 + 1)
  h1condi[, 1, ] <- rnorm(J * I, a1 * fit_ss$h10 + b1 * h1[, 2], sd1)
  h2condi[, 1, ] <- rnorm(J * I, a2 * fit_ss$h20 + b2 * h2[, 2], sd2)
  h1condi[, T, ] <- rnorm(J * I, phi * h1[, T - 1], tau)
  h1condi[, T, ] <- rnorm(J * I, phi2 * h2[, T - 1], tau2)
  for (t in 2:(T - 1)) {
    h1condi[, t, ] <- rnorm(J * I, a1 * h1[, t - 1] + b1 * h1[, t + 1], sd1)
    h2condi[, t, ] <- rnorm(J * I, a2 * h2[, t - 1] + b2 * h2[, t + 1], sd2)
  }

  th1condi <- aperm(h1condi, c(2, 1, 3))
  th2condi <- aperm(h2condi, c(2, 1, 3))
  mu_array <- array(mu, dim = c(I, T, J))
  tmu_array <- aperm(mu_array, c(2, 1, 3))
  log_py <- dnorm(y, mean = 0, sd = exp(tmu_array /2 + th1condi /2 + th2condi
      /2 ), log = T)
  log_ipy<-log_mean_exp_arr(log_py)
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

##########################################
## Creating files for the calculated results
##########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 5**

```
##########################################
## Sourcing functions, loading original data observations and the fitted
   chains
##########################################

source ("model5.r")
source ("iswaic.r")
source ("log_int_mid.r")
source ("log_f_num.R")
source ("log_f_denum.R")
source ("log_f_num_transformed.R")
source ("log_f_denum_transformed.R")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)
```

```
istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

###########################################
## Computing log integrated likelihood
###########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  rho <- fit_ss$rho
  y <- raw_data$y
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  J <- 100
  h <- fit_ss$h
  y0 <- fit_ss$y0
  h0 <- fit_ss$h0
  log_ipy <- matrix(0, nrow = I, ncol = T)
  hcondiT <- array (0, dim = c (1, I, J))
  hcondiT[1, , ] <- rnorm(J * I, mu + phi * (h[, T - 1] - mu) + rho * tau * y
      [T-1] * exp(-.5 * h[, T-1]), tau * sqrt (1-rho^2))
  log_pyT <- dnorm(y[T], mean = 0, sd = exp(hcondiT /2 ), log = T)
  log_ipy[, T] <- log_mean_exp_arr(log_pyT)

  log_ipy_num_1 <- log_int_mid (log_f_num_transformed, c(0, 1), n = 100, mu =
       mu, phi = phi, tau = tau, rho = rho, htplus = h[, 2], htminus = h0, yt
       = rep (y[1], I), ytminus = y0)
  log_ipy_denum_1 <- log_int_mid (log_f_denum_transformed, c(0, 1), n = 100,
      mu = mu, phi = phi, tau = tau, rho = rho, htplus = h[, 2], htminus = h0
      , yt = rep (y[1], I), ytminus = y0)
  log_ipy[, 1] <- log_ipy_num_1 - log_ipy_denum_1


for (t in 2:(T-1)) {
  log_ipy_num_t <- log_int_mid (log_f_num_transformed, c(0, 1), n = 100, mu =
     mu, phi = phi, tau = tau, rho = rho, htplus = h[, t+1], htminus = h[,
     t-1], yt = rep (y[t], I), ytminus = rep (y[t-1], I))
  log_ipy_denum_t <- log_int_mid (log_f_denum_transformed, c(0, 1), n = 100,
     mu = mu, phi = phi, tau = tau, rho = rho, htplus = h[, t+1], htminus =
     h[, t-1], yt = rep (y[t], I), ytminus = rep (y[t-1], I))
  log_ipy[, t] <- log_ipy_num_t - log_ipy_denum_t
}

}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

##########################################
## Creating files for the calculated results
##########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
   iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
    iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
   p_iis)
save (p_list, file = "p_list")
```

- **Model 6**

```
##########################################
```

```
## Sourcing functions, loading original data observations and the fitted
    chains
##########################################

source ("iswaic.r")
source ("model6.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

##########################################
## Computing DIC/WAIC/IS
##########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  Dthetabar <- -2* sum(dnorm (raw_data$y, 0, exp (hbar/2), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
```

```
)

waictime <- time1 + time4

##########################################
## Computing log integrated likelihood
##########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  kappa <- fit_ss$kappa
  beta <- fit_ss$beta
  y <- raw_data$y
  s <- fit_ss$s
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  y0 <- fit_ss$y0
  h <- fit_ss$h
  J <- 100
  hcondi <- array(0, dim = c(I, T, J))
  a <- phi / (phi^2 + 1)
  b <- phi / (phi^2 + 1)
  int <- ((mu - phi^2 * mu) / (phi^2 + 1) - mu + phi * mu) / phi
  sd <- tau / (phi^2 + 1)^0.5
  hcondi[, 1, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 2] + int, sd)
  hcondi[, T, ] <- rnorm(J * I, mu - phi * mu + phi * h[, T - 1], tau)
  for (t in 2:(T - 1)) {
    hcondi[,t,] <- rnorm(J * I, a * h[, t - 1] + b * h[, t +1 ] + int, sd)
  }

  thcondi <- aperm(hcondi, c(2, 1, 3))
  ytminus1_matrix <- rbind(y0, matrix(y[1:T-1], nrow = T-1, ncol = I))
  beta_matrix <- t(matrix(beta, nrow = I, ncol = T))
  kappa_array <- aperm(array(kappa, dim = c(I, T, J)), c(2, 1, 3))
  log_py <- log ((1 - kappa_array) * dnorm(y, mean = beta_matrix *
      ytminus1_matrix, sd = exp(thcondi /2 )) + kappa_array * dnorm(y, mean =
       beta_matrix * ytminus1_matrix + t(s), sd = exp(thcondi /2 )))
  log_ipy<-log_mean_exp_arr(log_py)
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

###########################################
## Creating files for the calculated results
###########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 7**

```
###########################################
## Sourcing functions, loading original data observations and the fitted
    chains
###########################################

source ("iswaic.r")
source ("model7.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

###########################################
## Computing DIC/WAIC/IS
###########################################
```

```
time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
}
)

time2 <- system.time (
{
  Dbar <- sum(colMeans(-2*log_lik))
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  kappabar <- sfit$summary["kappa", "mean"]
  srow <- grep("s\\[",rownames(sfit$summary))
  sbar <- sfit$summary[srow, "mean"]
  Dthetabar <- -2* sum(log((1 - kappabar) * dnorm(y, 0, exp(hbar /2 )) +
      kappabar * dnorm(y, sbar, exp(hbar/2 ))))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

##########################################
## Computing log integrated likelihood
##########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  kappa <- fit_ss$kappa
  delta <- fit_ss$delta
  s <- fit_ss$s
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  h <- fit_ss$h
  J <- 100
  hcondi <- log_py <- array(0, dim = c(I, T, J))

  a <- phi / (phi^2 + 1)
  b <- phi / (phi^2 + 1)
  int <- ((mu - phi^2 * mu) / (phi^2 + 1) - mu + phi * mu) / phi
  sd <- tau / (phi^2 + 1)^0.5
  hcondi[, 1, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 2] + int, sd)
  hcondi[, T, ] <- rnorm(J * I, mu - phi * mu + phi * h[, T - 1], tau)

  for (t in 2:(T - 1)) {
    hcondi[,t,] <- rnorm(J * I, a * h[, t - 1] + b * h[, t +1 ] + int, sd)
  }

  log1ps <- matrix (rnorm (J * I, -(delta^2/2), delta), nrow = I, ncol = J)
  s <- exp(log1ps) - 1
  kappa_matrix <- matrix(rep(kappa, J), nrow = I, ncol = J)

  for (t in 1:T){
    log_py[, t, ] <- log ((1 - kappa_matrix) * dnorm(y[t], 0, exp(hcondi[, t,
         ]/2 )) + kappa_matrix * dnorm(y[t], s, exp(hcondi[, t, ]/2 )))
  }

  log_ipy <- log_mean_exp_arr(aperm(log_py,c(2,1,3)))
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
```

```
)

iwaictime <- time5 + time6
iistime <- time5 +time7

###########################################
## Creating files for the calculated results
###########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```

- **Model 8**

```
###########################################
## Sourcing functions, loading original data observations and the fitted
    chains
###########################################

source ("iswaic.r")
source ("model8.r")
source ("log_sum_exp_arr.R")

raw_data <- load ("sample_data.RData")
load ("sample_fit")

###########################################
## Computing DIC/WAIC/IS
###########################################

time1 <- system.time (
{
  sfit<-summary(fit)
  log_lik <- extract (fit, "log_lik")$log_lik
  Dbar <- sum(colMeans(-2*log_lik))
}
)

time2 <- system.time (
{
  hrow <- grep("h\\[",rownames(sfit$summary))
  hbar <- sfit$summary[hrow, "mean"]
  wrow <- grep("w\\[",rownames(sfit$summary))
  wbar <- sfit$summary[wrow, "mean"]
  Dthetabar <- -2* sum(dnorm (y, 0, exp (hbar/2)/sqrt(wbar), log = TRUE))
  dic <- 2*Dbar - Dthetabar
}
)

dictime <- time1 + time2

time3 <- system.time (
{
  is <- is_function (log_lik)$is
  p_is <- is_function (log_lik)$p_is
}
)

istime <- time1 + time3

time4 <- system.time (
{
  waic <- waic_function (log_lik)$waic
  p_waic <- waic_function (log_lik)$p_waic
}
)

waictime <- time1 + time4

###########################################
## Computing log integrated likelihood
###########################################

time5 <- system.time (
{
  fit_ss <- extract(fit, permuted = TRUE)
  mu <- fit_ss$mu
  phi <- fit_ss$phi
  tau <- fit_ss$tau
  w <- fit_ss$w
  I <- length(fit_ss$h[, 1])
  T <- length(fit_ss$h[1, ])
  h <- fit_ss$h
  J <- 100
  log_py <- array(0, dim=c(I, T, J))
  hcondi <- array(0, dim = c(I, T, J))
  a <- phi / (phi^2 + 1)
  b <- phi / (phi^2 + 1)
  int <- ((mu - phi^2 * mu) / (phi^2 + 1) - mu + phi * mu) / phi
  sd <- tau / (phi^2 + 1)^0.5
  hcondi[, 1, ] <- rnorm(J * I, a * fit_ss$h0 + b * h[, 2] + int, sd)
  hcondi[, T, ] <- rnorm(J * I, mu - phi * mu + phi * h[, T - 1], tau)

  for (t in 2:(T - 1)) {
    hcondi[,t,] <- rnorm(J * I, a * h[, t - 1] + b * h[, t +1 ] + int, sd)
  }

  for (t in 1:T){
  log_py[, t, ] <- dnorm(y[t], mean = 0, sd = exp(hcondi[, t, ]/2) / sqrt(
      matrix(w[, t], nrow = I, ncol = J)), log = T)
  }
  log_ipy <- log_mean_exp_arr(aperm(log_py,c(2,1,3)))
}
)

time6 <- system.time(
{
  iwaic <- waic_function (log_ipy) $waic
  p_iwaic <- waic_function (log_ipy) $p_waic
}
)

time7 <- system.time(
{
  iis <- is_function (log_ipy) $is
  p_iis <- is_function (log_ipy) $p_is
}
)

iwaictime <- time5 + time6
iistime <- time5 +time7

##########################################
## Creating files for the calculated results
##########################################

dicwaicis_list <- list (dic = dic, waic = waic, iwaic =iwaic, is = is, iis =
    iis)
save (dicwaicis_list, file = "dicwaic_list")

system_time_list <- list (dictime = dictime, waictime = waictime, iwaictime =
     iwaictime, istime = istime, iistime = iistime)
save (system_time_list, file = "system_time_list")

p_list <- list (p_waic = p_waic, p_iwaic = p_iwaic, p_is = p_is, p_iis =
    p_iis)
save (p_list, file = "p_list")
```